

Makefile

Page 1/1

```

1: # fiwix/Makefile
2: #
3: # Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
4: # Distributed under the terms of the Fiwix License.
5: #
6:
7: TOPDIR := $(shell if [ "$$PWD" != "" ] ; then echo $$PWD ; else pwd ; fi)
8: INCLUDE = $(TOPDIR)/include
9:
10: ARCH = -m32
11: CPU = -march=i386
12:
13: CC = $(CROSS_COMPILE)gcc $(ARCH) $(CPU) -D__KERNEL__ #-D__DEBUG__
14: LD = $(CROSS_COMPILE)ld
15:
16: CFLAGS = -I$(INCLUDE) -O2 -ffreestanding -Wall -Wstrict-prototypes #-Wextra
17: LDFLAGS = -m elf_i386 -nostartfiles -nostdlib -nodefaultlibs -nostdinc
18:
19: DIRS = kernel kernel/syscalls mm fs drivers/block drivers/char drivers/video \
20:       lib
21: OBJS = kernel/kernel.o kernel/syscalls/syscalls.o mm/mm.o fs/fs.o \
22:       drivers/block/block.o drivers/char/char.o drivers/video/video.o \
23:       lib/lib.o
24:
25: export CC LD CFLAGS LDFLAGS INCLUDE
26:
27: all:
28:     @echo "#define UTS_VERSION \"`date`\"" > include/fiwix/version.h
29:     @for n in $(DIRS) ; do (cd $$n ; $(MAKE)) ; done
30:     $(LD) -N -T fiwix.ld $(LDFLAGS) $(OBJS) -o fiwix
31:     nm fiwix | sort | gzip -9c > System.map.gz
32:
33: clean:
34:     @for n in $(DIRS) ; do (cd $$n ; $(MAKE) clean) ; done
35:     rm -f *.o fiwix System.map.gz
36:
37: floppy:
38:     mkfs.ext2 -m 0 /dev/fd0
39:     mount -t ext2 /dev/fd0 /mnt/floppy
40:     @mkdir -p /mnt/floppy/boot/grub
41:     @echo "(fd0) /dev/fd0" > /mnt/floppy/boot/grub/device.map
42:     @grub-install --root-directory=/mnt/floppy /dev/fd0
43:     @tools/MAKEBOOTDISK
44:     @cp -prf tools/etc/* /mnt/floppy/etc
45:     @cp fiwix /mnt/floppy/boot
46:     @cp System.map.gz /mnt/floppy/boot
47:     @cp tools/install.sh /mnt/floppy/sbin
48:     umount /mnt/floppy
49:
50: floppy_update:
51:     mount -t ext2 /dev/fd0 /mnt/floppy
52:     cp fiwix /mnt/floppy/boot
53:     cp System.map.gz /mnt/floppy/boot
54:     umount /mnt/floppy
55:
56:

```

fiwix.ld

Page 1/1

```
1: /*
2:  * Linker script for the Fiwix kernel (3GB user / 1GB kernel).
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: OUTPUT_FORMAT("elf32-i386")
9: OUTPUT_ARCH(i386)
10: ENTRY(start)          /* entry point */
11: vaddr = 0xC0000000;    /* virtual base address at 3GB */
12: paddr = 0x100000;     /* physical address at 1MB */
13:
14: /* define output sections */
15: SECTIONS
16: {
17:     . = paddr;
18:
19:     /* kernel setup code */
20:     .setup ALIGN(4096) :
21:     {
22:         *(.setup)
23:     }
24:
25:     . += vaddr;
26:
27:     /* kernel code */
28:     .text : AT(ADDR(.text) - vaddr)
29:     {
30:         *(.text)
31:     }
32:     _etext = .;
33:
34:     /* initialized data */
35:     .data ALIGN(4096) : AT(ADDR(.data) - vaddr)
36:     {
37:         *(.data)
38:         *(.rodata*)
39:     }
40:     _edata = .;
41:
42:     /* uninitialized data */
43:     .bss ALIGN(4096) : AT(ADDR(.bss) - vaddr)
44:     {
45:         *(COMMON*)
46:         *(.bss*)
47:     }
48:     _end = .;
49:
50:     /* remove information not needed */
51:     /DISCARD/ :
52:     {
53:         *(.eh_frame)
54:     }
55: }
```

kernel/boot.S

```

1: /*
2:  * fiwix/kernel/boot.S
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #define ASM_FILE      1
9:
10: #include <fiwix/segments.h>
11: #include <fiwix/multiboot1.h>
12:
13: #define MULTIBOOT_HEADER_FLAGS MULTIBOOT_PAGE_ALIGN | MULTIBOOT_MEMORY_INFO
14:
15: /* flags for CR0 (control register) */
16: #define CR0_MP 0x00000002 /* bit 01 -> enable monitor coprocessor */
17: #define CR0_NE 0x00000020 /* bit 05 -> enable native x87 FPU mode */
18: #define CR0_WP 0x00010000 /* bit 16 -> enable write protect (for CoW) */
19: #define CR0_AM 0x00040000 /* bit 18 -> enable alignment checking */
20: #define CR0_PG 0x80000000 /* bit 31 -> enable paging */
21:
22: .section .setup, "a" /* "a" attribute means Allocatable section */
23:
24: .align 4
25: tmp_gdtr:
26:     .word ((3 * 8) - 1)
27:     .long tmp_gdt
28:
29: .align 4
30: tmp_gdt:
31:     /* NULL DESCRIPTOR */
32:     .word 0x0000
33:     .word 0x0000
34:     .word 0x0000
35:     .word 0x0000
36:
37:     /* KERNEL CODE */
38:     .word 0xFFFF /* segment limit 15-00 */
39:     .word 0x0000 /* base address 15-00 */
40:     .byte 0x00 /* base address 23-16 */
41:     .byte 0x9A /* P=1 DPL=00 S=1 TYPE=1010 (exec/read) */
42:     .byte 0xCF /* G=1 DB=1 0=0 AVL=0 SEGLIM=1111 */
43:     .byte 0x40 /* base address 31-24 */
44:
45:     /* KERNEL DATA */
46:     .word 0xFFFF /* segment limit 15-00 */
47:     .word 0x0000 /* base address 15-00 */
48:     .byte 0x00 /* base address 23-16 */
49:     .byte 0x92 /* P=1 DPL=00 S=1 TYPE=0010 (read/write) */
50:     .byte 0xCF /* G=1 DB=1 0=0 AVL=0 SEGLIM=1111 */
51:     .byte 0x40 /* base address 31-24 */
52:
53:
54: .text
55:
56: .align 4
57: multiboot_header: /* multiboot header */
58:     .long MULTIBOOT_HEADER_MAGIC /* magic */
59:     .long MULTIBOOT_HEADER_FLAGS /* flags */
60:     /* checksum */
61:     .long -(MULTIBOOT_HEADER_MAGIC + MULTIBOOT_HEADER_FLAGS)
62:
63:     /* no used */
64:     .long 0 /* header_addr */
65:     .long 0 /* load_addr */
66:     .long 0 /* load_end_addr */
67:     .long 0 /* bss_end_addr */

```

kernel/boot.S

Page 2/3

```

68:      .long    0                /* entry_addr */
69:
70:      /* valid only with GRUB2 */
71:      .long    0                /* mode_type */
72:      .long    0                /* width */
73:      .long    0                /* height */
74:      .long    0                /* depth */
75:
76: /*
77:  * We use the CX register in order to keep intact the values in AX and BX
78:  * registers, since they are holding the Multiboot values 'magic' and 'info'
79:  * respectively.
80:  */
81: .globl start; start:
82:     cli
83:     lgdt     tmp_gdtr          /* load GDTR with the temporary GDT */
84:     movw    $KERNEL_DS, %cx
85:     movw    %cx, %ds
86:     movw    %cx, %es
87:     movw    %cx, %fs
88:     movw    %cx, %gs
89:     movw    %cx, %ss
90:     ljmp    $KERNEL_CS, $1f
91: 1:
92:
93:
94: /*
95:  * WARNING WARNING WARNING WARNING WARNING WARNING WARNING WARNING WARNING !!!
96:  * -----
97:  * The minimal page directory of 4MB only works if the in-memory size of the
98:  * kernel is smaller than 3MB (which I presume will be its size for a very long
99:  * time). If you need more space go to the setup_minmem() function and set the
100:  * 'mb4' variable accordingly.
101:  *
102:  * In order to know the current size of the Fiwix kernel, just follow this:
103:  *
104:  * $ readelf -l fiwix
105:  * Elf file type is EXEC (Executable file)
106:  * Entry point 0xc0100050
107:  * There are 2 program headers, starting at offset 52
108:  *
109:  * Program Headers:
110:  *   Type           Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
111:  *   LOAD            0x000074 0x00100000 0x00100000 0x000020 0x000020 R  0x4
112:  *   LOAD            0x0000a0 0xc0100020 0x00100020 0x3996c 0x41cc0 RWE 0x20
113:  *                                     check this value --> ^^^^^^^
114:  *   ...
115:  */
116:     movl    $0xC0010000, %esp    /* default stack address */
117:     pushl   $0                  /* reset EFLAGS */
118:     popf
119:
120:     pushl   %eax                /* save Multiboot magic value */
121:     call    setup_minmem        /* setup a minimal page directory */
122:     movl    %eax, %cr3
123:
124:     movl    %cr0, %eax
125:     andl    $0x00000011, %eax    /* disable all, preserve ET & PE (GRUB)
*/
126:     orl    $CR0_PG, %eax        /* enable PG */
127:     orl    $CR0_AM, %eax        /* enable AM */
128:     orl    $CR0_WP, %eax        /* enable WP */
129:     orl    $CR0_NE, %eax        /* enable NE */
130:     orl    $CR0_MP, %eax        /* enable MP */
131:     movl    %eax, %cr0
132:
133:     call    bss_init            /* initialize BSS segment */

```

kernel/boot.S

Page 3/3

```
134:      call    gdt_init          /* setup and load the definitive GDT */
135:
136:      pushl  %ebx              /* save Multiboot info structure */
137:      call  get_last_boot_addr
138:      add    $4, %esp
139:      popl   %ecx              /* restore Multiboot magic value */
140:      andl  $0xFFFFF000, %eax /* page aligned */
141:      addl  $0x3000, %eax     /* 2 whole pages for kernel stack */
142:      subl  $4, %eax
143:      movl  %eax, %esp        /* set kernel stack */
144:
145:      pushl  %esp              /* save kernel stack address */
146:      pushl  %ebx              /* save Multiboot info structure */
147:      pushl  %ecx              /* save Multiboot magic value */
148:      call  start_kernel
149:
150:      /* not reached */
151:      jmp   cpu_idle
152:
```

kernel/cmos.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/cmos.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/cmos.h>
10:
11: int cmos_update_in_progress(void)
12: {
13:     return(cmos_read(CMOS_STATA) & CMOS_STATA_UIP);
14: }
15:
16: unsigned char cmos_read_date(unsigned char addr)
17: {
18:     /* make sure an update isn't in progress */
19:     while(cmos_update_in_progress());
20:
21:     if(!(cmos_read(CMOS_STATB) & CMOS_STATB_DM) {
22:         return BCD2BIN(cmos_read(addr));
23:     }
24:     return cmos_read(addr);
25: }
26:
27: void cmos_write_date(unsigned char addr, unsigned char value)
28: {
29:     /* make sure an update isn't in progress */
30:     while(cmos_update_in_progress());
31:
32:     if(!(cmos_read(CMOS_STATB) & CMOS_STATB_DM) {
33:         cmos_write(addr, BIN2BCD(value));
34:     }
35:     cmos_write(addr, value);
36: }
37:
38: unsigned char cmos_read(unsigned char addr)
39: {
40:     unsigned long int flags;
41:
42:     SAVE_FLAGS(flags); CLI();
43:     outport_b(CMOS_INDEX, addr);
44:     RESTORE_FLAGS(flags);
45:
46:     return inport_b(CMOS_DATA);
47: }
48:
49: void cmos_write(unsigned char addr, unsigned char value)
50: {
51:     unsigned long int flags;
52:
53:     SAVE_FLAGS(flags); CLI();
54:     outport_b(CMOS_INDEX, addr);
55:     outport_b(CMOS_DATA, value);
56:     RESTORE_FLAGS(flags);
57: }
```

kernel/core386.S

Page 1/10

```

1: /*
2:  * fiwix/kernel/core386.S
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #define ASM_FILE      1
9:
10: #include <fiwix/segments.h>
11: #include <fiwix/unistd.h>
12:
13: #define CRO_MP      ~ (0x00000002)    /* CRO bit-01 MP (Monitor Coprocessor) */
14: #define CRO_EM      0x00000004      /* CRO bit-02 EM (Emulation) */
15:
16: #define SS_RPL3      0x03          /* Request Privilege Level 3 */
17:
18: #define GS           0x00
19: #define FS           0x04
20: #define ES           0x08
21: #define DS           0x0C
22: #define EDI          0x10          /* \ */
23: #define ESI          0x14          /* | */
24: #define EBP          0x18          /* | */
25: #define ESP          0x1C          /* | saved by */
26: #define EBX          0x20          /* | 'pusha' */
27: #define EDX          0x24          /* | */
28: #define ECX          0x28          /* | */
29: #define EAX          0x2C          /* / */
30: #define ERR          0x30          /* error code (or padding) */
31: #define EIP          0x34          /* \ */
32: #define CS           0x38          /* | saved by processor */
33: #define FLAGS        0x3C          /* / */
34: #define OLDESP       0x40          /* \ saved by processor on */
35: #define OLDSS        0x44          /* / privilege level change */
36:
37: #define SAVE_ALL
38:     pushal                \
39:     pushl   %ds           ;\
40:     pushl   %es           ;\
41:     pushl   %fs           ;\
42:     pushl   %gs           ;\
43:
44: #define EXCEPTION(exception) \
45:     pushl   $exception   ;\
46:     call    trap_handler ;\
47:     addl    $4, %esp
48:
49: #define IRQ(irq) \
50:     pushl   $irq         ;\
51:     call    irq_handler  ;\
52:     addl    $4, %esp
53:
54: #define BOTTOM_HALVES \
55:     sti                ;\
56:     call    do_bh
57:
58: #define CHECK_IF_NESTED_INTERRUPT \
59:     testl   $$SS_RPL3, CS(%esp) ;\
60:     jz      2f
61:
62: #define CHECK_IF_SIGNALS \
63:     call    issig        ;\
64:     testl   $0xFFFFFFFF, %eax ;\
65:     jz      1f           ;\
66:     movl   %esp, %eax    ;\
67:     pushl  %eax          ;\

```

kernel/core386.S

Page 2/10

```

68:      call    psig                                ;\
69:      addl   $4, %esp                             ;\
70: 1:
71:
72: #define CHECK_IF_NEED_SCHEDULE                   \
73:      movl   need_resched, %eax                   ;\
74:      testl  $0xFFFFFFFF, %eax                   ;\
75:      jz     2f                                    ;\
76:      call   do_sched                             ;\
77: 2:
78:
79: #define RESTORE_ALL                             \
80:      popl   %gs                                  ;\
81:      popl   %fs                                  ;\
82:      popl   %es                                  ;\
83:      popl   %ds                                  ;\
84:      popal                                     ;\
85:      addl   $4, %esp      /* suppress error code from stack */
86:
87:
88: .text
89:
90: #define BUILD_EXCEPTION_SIMUL_ERR(num, name)    \
91: .align 4                                        ;\
92: .globl name; name:                             ;\
93:      pushl  $0      /* save simulated error code to stack */;\
94:      SAVE_ALL                                           ;\
95:      EXCEPTION(num)                                     ;\
96:      BOTTOM_HALVES                                       ;\
97:      CHECK_IF_NESTED_INTERRUPT                           ;\
98:      CHECK_IF_SIGNALS                                    ;\
99:      CHECK_IF_NEED_SCHEDULE                              ;\
100:     RESTORE_ALL                                           ;\
101:     iret
102:
103: #define BUILD_EXCEPTION(num, name)              \
104: .align 4                                        ;\
105: .globl name; name:                             ;\
106:     SAVE_ALL                                           ;\
107:     EXCEPTION(num)                                     ;\
108:     BOTTOM_HALVES                                       ;\
109:     CHECK_IF_NESTED_INTERRUPT                           ;\
110:     CHECK_IF_SIGNALS                                    ;\
111:     CHECK_IF_NEED_SCHEDULE                              ;\
112:     RESTORE_ALL                                           ;\
113:     iret
114:
115: BUILD_EXCEPTION_SIMUL_ERR(0, except0) /* DIVIDE ERROR */
116: BUILD_EXCEPTION_SIMUL_ERR(1, except1) /* DEBUG */
117: BUILD_EXCEPTION_SIMUL_ERR(2, except2) /* NMI INTERRUPT */
118: BUILD_EXCEPTION_SIMUL_ERR(3, except3) /* BREAKPOINT INT3 */
119: BUILD_EXCEPTION_SIMUL_ERR(4, except4) /* OVERFLOW */
120: BUILD_EXCEPTION_SIMUL_ERR(5, except5) /* BOUND */
121: BUILD_EXCEPTION_SIMUL_ERR(6, except6) /* INVALID OPCODE */
122:
123: .align 4
124: .globl except7; except7:      # NO MATH COPROCESSOR
125:     pushl  $0      # save simulated error code to stack
126:     SAVE_ALL
127:     EXCEPTION(0x7)
128:     clls      # floating-opcode cached!
129:     BOTTOM_HALVES
130:     CHECK_IF_NESTED_INTERRUPT
131:     CHECK_IF_SIGNALS
132:     CHECK_IF_NEED_SCHEDULE
133:     RESTORE_ALL
134:     iret

```


kernel/core386.S

Page 3/10

```

135:
136: BUILD_EXCEPTION(8, except8) /* DOUBLE FAULT */
137: BUILD_EXCEPTION_SIMUL_ERR(9, except9) /* COPROCESSOR SEGMENT OVERRUN */
138: BUILD_EXCEPTION(10, except10) /* INVALID TSS */
139: BUILD_EXCEPTION(11, except11) /* SEGMENT NOT PRESENT */
140: BUILD_EXCEPTION(12, except12) /* STACK SEGMENT FAULT */
141: BUILD_EXCEPTION(13, except13) /* GENERAL PROTECTION FAULT */
142: BUILD_EXCEPTION(14, except14) /* PAGE FAULT */
143: BUILD_EXCEPTION_SIMUL_ERR(15, except15) /* INTEL RESERVED */
144: BUILD_EXCEPTION_SIMUL_ERR(16, except16) /* FLOATING POINT ERROR */
145: BUILD_EXCEPTION(17, except17) /* ALIGNMENT CHECK */
146: BUILD_EXCEPTION_SIMUL_ERR(18, except18) /* MACHINE CHECK */
147: BUILD_EXCEPTION_SIMUL_ERR(19, except19) /* SIMD FLOATING POINT */
148: BUILD_EXCEPTION_SIMUL_ERR(20, except20) /* INTEL RESERVED */
149: BUILD_EXCEPTION_SIMUL_ERR(21, except21) /* INTEL RESERVED */
150: BUILD_EXCEPTION_SIMUL_ERR(22, except22) /* INTEL RESERVED */
151: BUILD_EXCEPTION_SIMUL_ERR(23, except23) /* INTEL RESERVED */
152: BUILD_EXCEPTION_SIMUL_ERR(24, except24) /* INTEL RESERVED */
153: BUILD_EXCEPTION_SIMUL_ERR(25, except25) /* INTEL RESERVED */
154: BUILD_EXCEPTION_SIMUL_ERR(26, except26) /* INTEL RESERVED */
155: BUILD_EXCEPTION_SIMUL_ERR(27, except27) /* INTEL RESERVED */
156: BUILD_EXCEPTION_SIMUL_ERR(28, except28) /* INTEL RESERVED */
157: BUILD_EXCEPTION_SIMUL_ERR(29, except29) /* INTEL RESERVED */
158: BUILD_EXCEPTION_SIMUL_ERR(30, except30) /* INTEL RESERVED */
159: BUILD_EXCEPTION_SIMUL_ERR(31, except31) /* INTEL RESERVED */
160:
161:
162: #define BUILD_IRQ(num, name) \
163: .align 4 ;\
164: .globl name; name: ;\
165: pushl $0 /* save simulated error code to stack */;\
166: SAVE_ALL ;\
167: IRQ(num) ;\
168: BOTTOM_HALVES ;\
169: CHECK_IF_NESTED_INTERRUPT ;\
170: CHECK_IF_SIGNALS ;\
171: CHECK_IF_NEED_SCHEDULE ;\
172: RESTORE_ALL ;\
173: iret ;\
174:
175: BUILD_IRQ(0, irq0)
176: BUILD_IRQ(1, irq1)
177: BUILD_IRQ(2, irq2)
178: BUILD_IRQ(3, irq3)
179: BUILD_IRQ(4, irq4)
180: BUILD_IRQ(5, irq5)
181: BUILD_IRQ(6, irq6)
182: BUILD_IRQ(7, irq7)
183: BUILD_IRQ(8, irq8)
184: BUILD_IRQ(9, irq9)
185: BUILD_IRQ(10, irq10)
186: BUILD_IRQ(11, irq11)
187: BUILD_IRQ(12, irq12)
188: BUILD_IRQ(13, irq13)
189: BUILD_IRQ(14, irq14)
190: BUILD_IRQ(15, irq15)
191:
192: .align 4
193: .globl unknown_irq; unknown_irq:
194: pushl $0 # save simulated error code to stack
195: SAVE_ALL
196: call unknown_irq_handler
197: RESTORE_ALL
198: iret
199:
200: .align 4
201: .globl switch_to_user_mode; switch_to_user_mode:

```

kernel/core386.S

Page 4/10

```

202:      cli
203:      xorl    %eax, %eax           # initialize %eax
204:      movl   %eax, %ebx           # initialize %ebx
205:      movl   %eax, %ecx           # initialize %ecx
206:      movl   %eax, %edx           # initialize %edx
207:      movl   %eax, %esi           # initialize %esi
208:      movl   %eax, %edi           # initialize %edi
209:      movl   %eax, %ebp           # initialize %ebp
210:      movl   $(USER_DS | SS_RPL3), %eax
211:      movw   %ax, %ds
212:      movw   %ax, %es
213:      movw   %ax, %fs
214:      movw   %ax, %gs
215:      pushl  %eax
216:      pushl  $KERNEL_BASE_ADDR - 4 # user stack address
217:      pushl  $0x202                # initialize eflags (Linux 2.2 = 0x292)
218:      popfl
219:      pushfl
220:      movl   $(USER_CS | SS_RPL3), %eax
221:      pushl  %eax
222:      pushl  $KERNEL_BASE_ADDR - 0x1000 # go to init_trampoline() in use
r mode
223:      iret
224:
225:      .align 4
226:      .globl sighandler_trampoline; sighandler_trampoline:
227:      /*
228:       * This pushes twice the %eax register in order to save the 'signum'
229:       * value from being clobbered by functions like printf(), during the
230:       * signal handler calling.
231:       * FIXME: is this a bug in Newlib or in my side?
232:       */
233:      pushl  %eax                 # save 'signum'
234:      pushl  %eax                 # 'signum' is the argument of the handle
r
235:      call   *%ecx
236:      addl   $4, %esp             # discard the first push
237:      popl   %ebx                 # restore 'signum'
238:
239:      movl   $SYS_sigreturn, %eax
240:      int   $0x80
241:
242:      # never reached, otherwise call sys_exit()
243:      movl   $SYS_exit, %eax
244:      int   $0x80
245:      ret
246:      .align 4
247:      .globl end_sighandler_trampoline; end_sighandler_trampoline:
248:      nop
249:
250:      .align 4
251:      .globl syscall; syscall:    # SYSTEM CALL ENTRY
252:      pushl  %eax                 # save the system call number
253:      SAVE_ALL
254:
255:      pushl  %edi                 # \ 5th argument
256:      pushl  %esi                 # | 4th argument
257:      pushl  %edx                 # | 3rd argument
258:      pushl  %ecx                 # | 2nd argument
259:      pushl  %ebx                 # / 1st argument
260:      pushl  %eax                 # system call number
261:      call   do_syscall
262:      addl   $24, %esp            # suppress all 6 pushl from the stack
263:      movl   %eax, EAX(%esp)      # save the return value
264:
265:      BOTTOM_HALVES
266:      CHECK_IF_SIGNALS

```

kernel/core386.S

Page 5/10

```

267:         CHECK_IF_NEED_SCHEDULE
268: .align 4
269: .globl return_from_syscall; return_from_syscall:
270:         RESTORE_ALL
271:         iret
272:
273: .align 4
274: .globl do_switch; do_switch:
275:         movl    %esp, %ebx
276:         pushal
277:         pushfl
278:         movl    0x4(%ebx), %eax           # save ESP to 'prev->tss.esp'
279:         movl    %esp, (%eax)
280:         movl    0x8(%ebx), %eax           # save EIP to 'prev->tss.eip'
281:         movl    $1f, (%eax)
282:         movl    0xC(%ebx), %esp           # load 'next->tss.esp' into ESP
283:         pushl   0x10(%ebx)                # push 'next->tss.eip' into ESP
284:         movl    0x14(%ebx), %eax          # load 'next->tss.cr3' into CR3
285:         ltr     0x18(%ebx)                # load TSS
286:         movl    %eax, %cr3
287:         ret
288: 1:
289:         popfl
290:         popal
291:
292: .align 4
293: .globl cpuid; cpuid:
294:         pushl   %ebp
295:         movl    %esp, %ebp
296:         pushl   %edi
297:         pushl   %esi
298:         pushl   %ebx
299:
300:         pushf
301:         pop     %eax                       # put original EFLAGS in EAX
302:         mov     %eax, %ecx                 # save original EFLAGS in ECX
303:         xor     $0x200000, %eax           # change bit 21 (ID) in EFLAGS
304:         push   %eax                       # save new EFLAGS on stack
305:         popf
306:         pushf
307:         pop     %eax                       # put EFLAGS in EAX
308:         cmp     %ecx, %eax                 # compare if both EFLAGS are equal
309:
310:         je     test386                    # can't toggle ID bit, no CPUID
311:         xor     %ebx, %ebx                 # CPUID available, will return 0
312:         jmp    end_cpuid
313:
314: test386:
315:         mov     %ecx, %eax                 # get original EFLAGS
316:         xor     $0x40000, %eax            # change bit 18 (AC) in EFLAGS
317:         push   %eax                       # save new EFLAGS on stack
318:         popf
319:         pushf
320:         pop     %eax                       # replace current EFLAGS
321:         cmp     %ecx, %eax                 # compare if both EFLAGS are equal
322:         movb   $3, %bl                     # looks like an i386, return 3
323:         je     end_cpuid
324:         movb   $4, %bl                     # otherwise is an old i486, return 4
325:
326: end_cpuid:
327:         push   %ecx                       # push original EFLAGS
328:         popf
329:         xor     %eax, %eax                 # restore original EFLAGS
330:         movb   %bl, %al                    # put return value to AL
331:
332:         popl   %ebx
333:         popl   %esi

```

kernel/core386.S

Page 6/10

```

334:     popl    %edi
335:     popl    %ebp
336:     ret
337:
338: .align 4
339: .globl getfpu; getfpu:
340:     pushl   %ebp
341:     movl    %esp, %ebp
342:     pushl   %edi
343:     pushl   %esi
344:     pushl   %ebx
345:
346:     fninit
347:     movl    $0x5a5a, _fpstatus
348:     fnstsw _fpstatus
349:     movl    _fpstatus, %eax
350:     cmp     $0, %al
351:     movl    $0, _fpstatus
352:     jne     end_getfpu
353:
354: check_control_word:
355:     fnstcw _fpstatus
356:     movl    _fpstatus, %eax
357:     andl    $0x103f, %eax
358:     cmp     $0x3f, %ax
359:     movl    $0, _fpstatus
360:     jne     end_getfpu
361:     movl    $1, _fpstatus
362:
363: end_getfpu:
364:     movl    _fpstatus, %eax
365:     cmp     $0, %al
366:     jne     1f                               # return if there is a coprocessor
367:     movl    %cr0, %eax                       # otherwise (no math processor):
368:     orl    $CR0_EM, %eax                     # - set EM (Emulation)
369:     andl    $CR0_MP, %eax                    # - clear MP (Monitor Coprocessor)
370:     movl    %eax, %cr0
371:     movl    $0, %eax
372: 1:
373:     popl    %ebx
374:     popl    %esi
375:     popl    %edi
376:     popl    %ebp
377:     ret
378:
379: .align 4
380: .globl vendor_id; vendor_id:
381:     pushl   %ebp
382:     movl    %esp, %ebp
383:     pushl   %ebx
384:     pushl   %edx
385:     pushl   %ecx
386:
387:     mov     $0, %eax
388:     cpuid
389:     movl    %ebx, _vendorid                  # save the 12 bytes of vendor ID string
390:     movl    %edx, _vendorid+4
391:     movl    %ecx, _vendorid+8
392:
393:     popl    %ecx
394:     popl    %edx
395:     popl    %ebx
396:     popl    %ebp
397:     ret                                       # EAX returns the highest CPUID value
398:
399: .align 4
400: .globl signature_flags; signature_flags:

```

kernel/core386.S

Page 7/10

```

401:    pushl   %ebp
402:    movl    %esp, %ebp
403:    pushl   %edi
404:    pushl   %esi
405:    pushl   %ebx
406:    pushl   %edx
407:
408:    mov     $1, %eax
409:    cpuid
410:    movl    %eax, _cpusignature    # signature (model and stepping)
411:    movl    %ebx, _brandid        # misc. information
412:    movl    %edx, _cpuflags      # feature flags
413:    shrl   $8, %eax
414:    andl   $0xF, %eax
415:    movl    %eax, _cputype       # family
416:
417:    popl   %edx
418:    popl   %ebx
419:    popl   %esi
420:    popl   %edi
421:    popl   %ebp
422:    ret
423:
424:    .align 4
425:    .globl brand_str; brand_str:
426:    pushl   %ebp
427:    movl    %esp, %ebp
428:    pushl   %edi
429:    pushl   %esi
430:    pushl   %ebx
431:
432:    movl    $0x80000000, %eax
433:    cpuid
434:    cmp     $0x80000000, %eax    # check if brand string is supported
435:    jbe    no_brand_str
436:    movl    $0x80000002, %eax    # get first 16 bytes of brand string
437:    cpuid
438:    movl    %eax, _brandstr
439:    movl    %ebx, _brandstr+4
440:    movl    %ecx, _brandstr+8
441:    movl    %edx, _brandstr+12
442:    movl    $0x80000003, %eax    # get more 16 bytes of brand string
443:    cpuid
444:    movl    %eax, _brandstr+16
445:    movl    %ebx, _brandstr+20
446:    movl    %ecx, _brandstr+24
447:    movl    %edx, _brandstr+28
448:    movl    $0x80000004, %eax    # get last 16 bytes of brand string
449:    cpuid
450:    movl    %eax, _brandstr+32
451:    movl    %ebx, _brandstr+36
452:    movl    %ecx, _brandstr+40
453:    movl    %edx, _brandstr+44
454:    jmp     end_brand_str
455:
456:    no_brand_str:
457:    movl    $1, %eax
458:
459:    end_brand_str:
460:    movl    $0, %eax
461:    popl   %ebx
462:    popl   %esi
463:    popl   %edi
464:    popl   %ebp
465:    ret
466:
467:    .align 4

```

kernel/core386.S

Page 8/10

```

468: .globl tlbinfo; tlbinfo:
469:     pushl   %edx
470:     pushl   %ecx
471:     mov     $2, %eax
472:     cpuid
473:     movl    %eax, _tlbinfo_eax    # store cache information
474:     movl    %ebx, _tlbinfo_ebx
475:     movl    %edx, _tlbinfo_ecx
476:     movl    %ecx, _tlbinfo_edx
477:     popl    %ecx
478:     popl    %edx
479:     ret
480:
481: .align 4
482: .globl inport_b; inport_b:
483:     pushl   %ebp
484:     movl    %esp, %ebp
485:
486:     movw    0x08(%ebp), %dx      # port addr
487:     inb     %dx, %al
488:
489:     jmp     1f                  # recovery time
490: 1:     jmp     1f                  # recovery time
491: 1:     popl   %ebp
492:     ret
493:
494: .align 4
495: .globl inport_w; inport_w:
496:     pushl   %ebp
497:     movl    %esp, %ebp
498:
499:     movw    0x08(%ebp), %dx      # port addr
500:     inw     %dx, %ax
501:
502:     jmp     1f                  # recovery time
503: 1:     jmp     1f                  # recovery time
504: 1:     popl   %ebp
505:     ret
506:
507: .align 4
508: .globl inport_sw; inport_sw:
509:     pushl   %ebp
510:     movl    %esp, %ebp
511:     pushl   %edx
512:     pushl   %edi
513:     pushl   %ecx
514:
515:     cld
516:     mov     0x8(%ebp), %edx      # port addr
517:     mov     0xC(%ebp), %edi      # dest
518:     mov     0x10(%ebp), %ecx     # count
519:     rep     insw
520:
521:     popl    %ecx
522:     popl    %edi
523:     popl    %edx
524:     popl    %ebp
525:     ret
526:
527: .align 4
528: .globl outport_b; outport_b:
529:     pushl   %ebp
530:     movl    %esp, %ebp
531:
532:     movw    0x8(%ebp), %dx      # port addr
533:     movb    0xC(%ebp), %al      # data
534:     outb    %al, %dx

```

kernel/core386.S

Page 9/10

```

535:
536:         jmp     1f                # recovery time
537: 1:      jmp     1f                # recovery time
538: 1:      popl   %ebp
539:         ret
540:
541: .align 4
542: .globl outport_w; outport_w:
543:         pushl  %ebp
544:         movl   %esp, %ebp
545:
546:         movw   0x8(%ebp), %dx     # port addr
547:         movw   0xC(%ebp), %ax    # data
548:         outw   %ax, %dx
549:
550:         jmp     1f                # recovery time
551: 1:      jmp     1f                # recovery time
552: 1:      popl   %ebp
553:         ret
554:
555: .align 4
556: .globl outport_sw; outport_sw:
557:         pushl  %ebp
558:         movl   %esp, %ebp
559:         pushl  %edx
560:         pushl  %esi
561:         pushl  %ecx
562:
563:         cld
564:         mov    0x8(%ebp), %edx    # port addr
565:         mov    0xC(%ebp), %esi    # src
566:         mov    0x10(%ebp), %ecx   # count
567:         rep    outsw
568:
569:         popl   %ecx
570:         popl   %esi
571:         popl   %edx
572:         popl   %ebp
573:         ret
574:
575: .align 4
576: .globl load_gdt; load_gdt:
577:         movl   0x4(%esp), %eax
578:         lgdt   (%eax)
579:         movw   $KERNEL_DS, %ax
580:         movw   %ax, %ds
581:         movw   %ax, %es
582:         movw   %ax, %fs
583:         movw   %ax, %gs
584:         movw   %ax, %ss
585:         ljmp  $KERNEL_CS, $1f
586: 1:      ret
587:
588:
589: .align 4
590: .globl load_idt; load_idt:
591:         movl   0x4(%esp), %eax
592:         lidt   (%eax)
593:         ret
594:
595: .align 4
596: .globl activate_kpage_dir; activate_kpage_dir:
597:         movl   kpage_dir, %eax
598:         movl   %eax, %cr3
599:         ret
600:
601: .align 4

```

kernel/core386.S

Page 10/10

```
602: .globl load_tr; load_tr:
603:     mov     0x4(%esp), %ax
604:     ltr    %ax
605:     ret
606:
607: .align 4
608: .globl get_rdtsc; get_rdtsc:
609:     cpuid
610:     rdtsc
611:     ret
612:
613: .align 4
614: .globl invalidate_tlb; invalidate_tlb:
615:     movl   %cr3, %eax
616:     movl   %eax, %cr3
617:     ret
618:
619:
620: .data
621:
622: .globl  _cputype
623: .globl  _cpusignature
624: .globl  _cpuflags
625: .globl  _fpstatus
626: .globl  _brandid
627: .globl  _vendorid
628: .globl  _brandstr
629: .globl  _tlbinfo_eax
630: .globl  _tlbinfo_ebx
631: .globl  _tlbinfo_ecx
632: .globl  _tlbinfo_edx
633:
634: _cputype:      .int    0
635: _cpusignature: .int    0
636: _cpuflags:     .int    0
637: _fpstatus:     .int    0
638: _brandid:      .int    0
639: _vendorid:     .fill   13,1,0
640: _brandstr:     .fill   49,1,0
641: _tlbinfo_eax:  .int    0
642: _tlbinfo_ebx:  .int    0
643: _tlbinfo_ecx:  .int    0
644: _tlbinfo_edx:  .int    0
```


kernel/cpu.c

Page 1/5

```

1: /*
2:  * fiwix/kernel/cpu.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/kernel.h>
10: #include <fiwix/utsname.h>
11: #include <fiwix/pic.h>
12: #include <fiwix/pit.h>
13: #include <fiwix/cpu.h>
14: #include <fiwix/timer.h>
15: #include <fiwix/stdio.h>
16: #include <fiwix/string.h>
17:
18: char UTS_MACHINE[_UTSNAME_LENGTH];
19:
20: static struct cpu_type intel[] = {
21:     { 4,
22:       { "i486 DX", "i486 DX", "i486 SX", "i486 DX/2",
23:         "i486 SL", "i486 SX/2", NULL, "i486 DX/2 WBE",
24:         "i486 DX/4", NULL, NULL, NULL, NULL, NULL, NULL, NULL }
25:     },
26:     { 5,
27:       { NULL, "Pentium 60/66", "Pentium 75-200", "Pentium ODfor486",
28:         "PentiumMMX", NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
29:         NULL, NULL, NULL }
30:     },
31:     { 6,
32:       { NULL, "Pentium Pro", NULL, "Pentium II", NULL, "Pentium II",
33:         "Intel Celeron", "Pentium III", "Pentium III", NULL,
34:         "Pentium III Xeon", "Pentium III", NULL, NULL, NULL, NULL }
35:     }
36: };
37:
38: static const char *cpu_flags[] = {
39:     "FPU", "VME", "DE", "PSE", "TSC", "MSR", "PAE", "MCE", "CX8", "APIC",
40:     "10", "SEP", "MTRR", "PGE", "MCA", "CMOV", "PAT", "PSE-36", "PSN",
41:     "CLFSH", "20", "DS", "ACPI", "MMX", "FXSR", "SSE", "SSE2", "SS",
42:     "HTT", "TM", "30", "PBE"
43: };
44:
45: static unsigned long int detect_cpuspeed(void)
46: {
47:     unsigned long long int tsc1, tsc2;
48:
49:     outport_b(MODEREG, SEL_CHAN2 | LSB_MSB | TERM_COUNT | BINARY_CTR);
50:     outport_b(CHANNEL2, (OSCIL / HZ) & 0xFF);
51:     outport_b(CHANNEL2, (OSCIL / HZ) >> 8);
52:     outport_b(PS2_SYSCtrl_B, inport_b(PS2_SYSCtrl_B) | ENABLE_SDATA | ENABLE
_TMR2G);
53:
54:     tsc1 = 0;
55:     tsc1 = get_rdtsc();
56:
57:     while(!(inport_b(PS2_SYSCtrl_B) & 0x20));
58:
59:     tsc2 = 0;
60:     tsc2 = get_rdtsc();
61:
62:     outport_b(PS2_SYSCtrl_B, inport_b(PS2_SYSCtrl_B) & ~(ENABLE_SDATA | ENAB
LE_TMR2G));
63:
64:     return (tsc2 - tsc1) * HZ;
65: }

```

kernel/cpu.c

Page 2/5

```

66:
67: /*
68:  * These are the 2nd and 3rd level cache values according to Intel Processor
69:  * Identification and the CPUID Instruction.
70:  * Application Note 485. Document Number: 241618-031. September 2006.
71:  */
72: static void show_cache(int value)
73: {
74:     switch(value) {
75:         /* 2nd level cache */
76:         case 0x39:
77:         case 0x3B:
78:         case 0x41:
79:         case 0x79:
80:             cpu_table.cache = "128KB L2";
81:             break;
82:         case 0x3A:
83:             cpu_table.cache = "192KB L2";
84:             break;
85:         case 0x3C:
86:         case 0x42:
87:         case 0x7A:
88:         case 0x82:
89:             cpu_table.cache = "256KB L2";
90:             break;
91:         case 0x3D:
92:             cpu_table.cache = "384KB L2";
93:             break;
94:         case 0x3E:
95:         case 0x43:
96:         case 0x7B:
97:         case 0x7F:
98:         case 0x83:
99:         case 0x86:
100:            cpu_table.cache = "512KB L2";
101:            break;
102:         case 0x44:
103:         case 0x78:
104:         case 0x7C:
105:         case 0x84:
106:         case 0x87:
107:            cpu_table.cache = "1MB L2";
108:            break;
109:         case 0x45:
110:         case 0x7D:
111:         case 0x85:
112:            cpu_table.cache = "2MB L2";
113:            break;
114:
115:         /* 3rd level cache */
116:         case 0x22:
117:             cpu_table.cache = "512KB L3";
118:             break;
119:         case 0x23:
120:             cpu_table.cache = "1MB L3";
121:             break;
122:         case 0x25:
123:             cpu_table.cache = "2MB L3";
124:             break;
125:         case 0x29:
126:         case 0x46:
127:             cpu_table.cache = "4MB L3";
128:             break;
129:         case 0x49:
130:             cpu_table.cache = "4MB L3 & L2";
131:             break;
132:         case 0x4A:

```

kernel/cpu.c

Page 3/5

```

133:         cpu_table.cache = "6MB L3";
134:         break;
135:     case 0x47:
136:     case 0x4B:
137:         cpu_table.cache = "8MB L3";
138:         break;
139:     case 0x4C:
140:         cpu_table.cache = "12MB L3";
141:         break;
142:     case 0x4D:
143:         cpu_table.cache = "16MB L3";
144:         break;
145:     default:
146:         break;
147:     }
148: }
149:
150: static void check_cache(int maxcpuid)
151: {
152:     int n, maxcpuids;
153:
154:     maxcpuids = 1;
155:     if(maxcpuid >= 2) {
156:         for(n = 0; n < maxcpuids; n++) {
157:             tlbinfo();
158:             maxcpuids = _tlbinfo_eax & 0xFF;
159:             show_cache((_tlbinfo_eax >> 8) & 0xFF);
160:             show_cache((_tlbinfo_eax >> 16) & 0xFF);
161:             show_cache((_tlbinfo_eax >> 24) & 0xFF);
162:             if(!_tlbinfo_ebx & RESERVED_DESC) {
163:                 show_cache(_tlbinfo_ebx & 0xFF);
164:                 show_cache((_tlbinfo_ebx >> 8) & 0xFF);
165:                 show_cache((_tlbinfo_ebx >> 16) & 0xFF);
166:                 show_cache((_tlbinfo_ebx >> 24) & 0xFF);
167:             }
168:             if(!_tlbinfo_ecx & RESERVED_DESC) {
169:                 show_cache(_tlbinfo_ecx & 0xFF);
170:                 show_cache((_tlbinfo_ecx >> 8) & 0xFF);
171:                 show_cache((_tlbinfo_ecx >> 16) & 0xFF);
172:                 show_cache((_tlbinfo_ecx >> 24) & 0xFF);
173:             }
174:             if(!_tlbinfo_edx & RESERVED_DESC) {
175:                 show_cache(_tlbinfo_edx & 0xFF);
176:                 show_cache((_tlbinfo_edx >> 8) & 0xFF);
177:                 show_cache((_tlbinfo_edx >> 16) & 0xFF);
178:                 show_cache((_tlbinfo_edx >> 24) & 0xFF);
179:             }
180:         }
181:     }
182: }
183:
184: int get_cpu_flags(char *buffer, int offset)
185: {
186:     int n, size;
187:     unsigned int mask;
188:
189:     size = sprintf(buffer + offset, "flags          :");
190:     for(n = 0, mask = 1; n < 32; n++, mask <<= 1) {
191:         if(_cpuflags & mask) {
192:             size += sprintf(buffer + offset + size, " %s", cpu_flags
[n]);
193:         }
194:     }
195:     size += sprintf(buffer + offset + size, "\n");
196:     return size;
197: }
198:

```


kernel/cpu.c

Page 5/5

```
259:                printk("80%d86\n", cpu_table.family);
260:                cpu_table.has_cpuid = 0;
261:            }
262:            cpu_table.has_fpu = getfpu();
263: }
```

kernel/gdt.c

Page 1/1

```

1:  /*
2:  *  fiwix/kernel/gdt.c
3:  *
4:  *  Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  *  Distributed under the terms of the Fiwix License.
6:  */
7:
8:  #include <fiwix/asm.h>
9:  #include <fiwix/types.h>
10: #include <fiwix/segments.h>
11: #include <fiwix/process.h>
12: #include <fiwix/limits.h>
13: #include <fiwix/string.h>
14:
15: struct seg_desc gdt[NR_GDT_ENTRIES];
16:
17: struct desc_r gdtr = {
18:     sizeof(gdt) - 1,
19:     (unsigned int)&gdt
20: };
21:
22: static void gdt_set_entry(int num, unsigned int base_addr, unsigned int limit, c
har loflags, char hiflags)
23: {
24:     num /= sizeof(struct seg_desc);
25:     gdt[num].sd_lolimit = limit & 0xFFFF;
26:     gdt[num].sd_lobase = base_addr & 0xFFFFFFFF;
27:     gdt[num].sd_loflags = loflags;
28:     gdt[num].sd_hilimit = (limit >> 16) & 0x0F;
29:     gdt[num].sd_hiflags = hiflags;
30:     gdt[num].sd_hibase = (base_addr >> 24) & 0xFF;
31: }
32:
33: void gdt_init(void)
34: {
35:     unsigned char loflags;
36:
37:     gdt_set_entry(0, 0, 0, 0, 0); /* null descriptor */
38:
39:     loflags = SD_CODE | SD_CD | SD_DPL0 | SD_PRESENT;
40:     gdt_set_entry(KERNEL_CS, 0, 0xFFFFFFFF, loflags, SD_OPSIZE32 | SD_PAGE4K
B);
41:     loflags = SD_DATA | SD_CD | SD_DPL0 | SD_PRESENT;
42:     gdt_set_entry(KERNEL_DS, 0, 0xFFFFFFFF, loflags, SD_OPSIZE32 | SD_PAGE4K
B);
43:
44:     loflags = SD_CODE | SD_CD | SD_DPL3 | SD_PRESENT;
45:     gdt_set_entry(USER_CS, 0, 0xFFFFFFFF, loflags, SD_OPSIZE32 | SD_PAGE4KB)
;
46:     loflags = SD_DATA | SD_CD | SD_DPL3 | SD_PRESENT;
47:     gdt_set_entry(USER_DS, 0, 0xFFFFFFFF, loflags, SD_OPSIZE32 | SD_PAGE4KB)
;
48:
49:     loflags = SD_TSSPRESENT;
50:     gdt_set_entry(TSS, 0, sizeof(struct proc) - 1, loflags, SD_OPSIZE32);
51:
52:     load_gdt((unsigned int)&gdtr);
53: }

```

kernel/idt.c

Page 1/2

```

1: /*
2:  * fiwix/kernel/idt.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/types.h>
10: #include <fiwix/segments.h>
11: #include <fiwix/string.h>
12:
13: struct gate_desc idt[NR_IDT_ENTRIES];
14:
15: struct desc_r idtr = {
16:     sizeof(idt) - 1,
17:     (unsigned int)idt
18: };
19:
20: static void *except_handlers[] = {
21:     &except0, &except1, &except2, &except3, &except4, &except5, &except6, &e
xcept7,
22:     &except8, &except9, &except10, &except11, &except12, &except13, &except1
4, &except15,
23:     &except16, &except17, &except18, &except19, &except20, &except21, &excep
t22, &except23,
24:     &except24, &except25, &except26, &except27, &except28, &except29, &excep
t30, &except31
25: };
26:
27: static void *irq_handlers[] = {
28:     &irq0, &irq1, &irq2, &irq3, &irq4, &irq5, &irq6, &irq7,
29:     &irq8, &irq9, &irq10, &irq11, &irq12, &irq13, &irq14, &irq15
30: };
31:
32: static void set_idt_entry(int num, __off_t handler, unsigned int flags)
33: {
34:     idt[num].gd_looffset = handler & 0x0000FFFF;
35:     idt[num].gd_selector = KERNEL_CS;
36:     idt[num].gd_flags = flags << 8;
37:     idt[num].gd_hioffset = handler >> 16;
38: }
39:
40: void idt_init(void)
41: {
42:     int n;
43:
44:     memset_b(idt, NULL, sizeof(idt));
45:     for(n = 0; n < NR_IDT_ENTRIES; n++) {
46:         if(n < 0x20) {
47:             set_idt_entry(n, (__off_t)except_handlers[n], SD_32TRAPG
ATE | SD_PRESENT);
48:             continue;
49:         }
50:         if(n < 0x30) {
51:             set_idt_entry(n, (__off_t)irq_handlers[n - 0x20], SD_32I
NTRGATE | SD_PRESENT);
52:             continue;
53:         }
54:         set_idt_entry(n, (__off_t)&unknown_irq, SD_32INTRGATE | SD_PRESE
NT);
55:     }
56:
57:     set_idt_entry(0x80, (__off_t)&syscall, SD_32TRAPGATE | SD_DPL3 | SD_PRES
ENT);
58:
59:     load_idt((unsigned int)&idtr);

```

```
60: }
```


kernel/init.c

Page 1/2

```

1: /*
2:  * fiwix/kernel/init.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/kernel.h>
10: #include <fiwix/system.h>
11: #include <fiwix/mm.h>
12: #include <fiwix/timer.h>
13: #include <fiwix/sched.h>
14: #include <fiwix/sleep.h>
15: #include <fiwix/fcntl.h>
16: #include <fiwix/stat.h>
17: #include <fiwix/process.h>
18: #include <fiwix/syscalls.h>
19: #include <fiwix/unistd.h>
20: #include <fiwix/stdio.h>
21: #include <fiwix/string.h>
22:
23: #define INIT_TRAMPOLINE_SIZE    128    /* max. size of init_trampoline() */
24:
25: char *init_argv[] = { INIT_PROGRAM, NULL, NULL };
26: char *init_envp[] = { "HOME=/", "TERM=linux", NULL };
27:
28: static void init_trampoline(void)
29: {
30:     USER_SYSCALL(SYS_open, "/dev/console", O_RDWR, 0);    /* stdin */
31:     USER_SYSCALL(SYS_dup, 0, NULL, NULL);    /* stdout */
32:     USER_SYSCALL(SYS_dup, 0, NULL, NULL);    /* stderr */
33:     USER_SYSCALL(SYS_execve, INIT_PROGRAM, init_argv, init_envp);
34:
35:     /* only reached in case of error in sys_execve() */
36:     USER_SYSCALL(SYS_exit, NULL, NULL, NULL);
37: }
38:
39: void init_init(void)
40: {
41:     int n;
42:     unsigned int page;
43:     struct inode *i;
44:     unsigned int *pgdir;
45:     struct proc *init;
46:
47:     if(namei(INIT_PROGRAM, &i, NULL, FOLLOW_LINKS)) {
48:         PANIC("can't find %s.\n", INIT_PROGRAM);
49:     }
50:     if(!S_ISREG(i->i_mode)) {
51:         PANIC("%s is not a regular file.\n", INIT_PROGRAM);
52:     }
53:     iput(i);
54:
55:     /* INIT slot was already created in main.c */
56:     init = &proc_table[INIT];
57:
58:     /* INIT process starts with the current (kernel) Page Directory */
59:     if(!(pgdir = (void *)kmalloc())) {
60:         goto init_init__die;
61:     }
62:     init->rss++;
63:     memcpy_b(pgdir, kpage_dir, PAGE_SIZE);
64:     init->tss.cr3 = V2P((unsigned int)pgdir);
65:
66:     memset_b(init->vma, NULL, sizeof(init->vma));
67:     init->ppid = 0;

```

kernel/init.c

Page 2/2

```

68:         init->pgid = 0;
69:         init->sid = 0;
70:         init->flags = 0;
71:         init->children = 0;
72:         init->priority = DEF_PRIORITY;
73:         init->start_time = CURRENT_TICKS;
74:         init->sleep_address = NULL;
75:         init->uid = init->gid = 0;
76:         init->euid = init->egid = 0;
77:         init->suid = init->sgid = 0;
78:         memset_b(init->fd, NULL, sizeof(init->fd));
79:         memset_b(init->fd_flags, NULL, sizeof(init->fd_flags));
80:         init->root = current->root;
81:         init->pwd = current->pwd;
82:         strcpy(init->argv0, init_argv[0]);
83:         init_argv[1] = init_args;
84:         sprintfk(init->pidstr, "%d", init->pid);
85:         init->sigpending = 0;
86:         init->sigblocked = 0;
87:         init->sigexecuting = 0;
88:         memset_b(init->sigaction, NULL, sizeof(init->sigaction));
89:         memset_b(&init->usage, NULL, sizeof(struct rusage));
90:         memset_b(&init->cusage, NULL, sizeof(struct rusage));
91:         init->timeout = 0;
92:         for(n = 0; n < RLIM_NLIMITS; n++) {
93:             init->rlim[n].rlim_cur = init->rlim[n].rlim_max = RLIM_INFINITY;
94:         }
95:         init->rlim[RLIMIT_NOFILE].rlim_cur = OPEN_MAX;
96:         init->rlim[RLIMIT_NOFILE].rlim_max = NR_OPENS;
97:         init->rlim[RLIMIT_NPROC].rlim_cur = CHILD_MAX;
98:         init->rlim[RLIMIT_NPROC].rlim_max = NR_PROCS;
99:         init->umask = 0022;
100:
101:         /* setup the stack */
102:         if(!(init->tss.esp0 = kmalloc())) {
103:             goto init_init__die;
104:         }
105:         init->tss.esp0 += PAGE_SIZE - 4;
106:         init->rss++;
107:         init->tss.ss0 = KERNEL_DS;
108:
109:         /* setup the init_trampoline */
110:         page = map_page(init, KERNEL_BASE_ADDR - PAGE_SIZE, 0, PROT_READ | PROT_
WRITE);
111:         memcpy_b((void *)page, init_trampoline, INIT_TRAMPOLINE_SIZE);
112:
113:         init->tss.eip = (unsigned int)switch_to_user_mode;
114:         init->tss.esp = page + PAGE_SIZE - 4;
115:
116:         runnable(init);
117:         nr_processes++;
118:         return;
119:
120: init_init__die:
121:         PANIC("unable to run init process.\n");
122:     }

```

kernel/main.c

Page 1/3

```

1: /*
2:  * fiwix/kernel/main.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/kernel.h>
10: #include <fiwix/limits.h>
11: #include <fiwix/fs.h>
12: #include <fiwix/system.h>
13: #include <fiwix/version.h>
14: #include <fiwix/utsname.h>
15: #include <fiwix/stdio.h>
16: #include <fiwix/string.h>
17: #include <fiwix/video.h>
18: #include <fiwix/console.h>
19: #include <fiwix/pic.h>
20: #include <fiwix/segments.h>
21: #include <fiwix/devices.h>
22: #include <fiwix/cpu.h>
23: #include <fiwix/timer.h>
24: #include <fiwix/sleep.h>
25: #include <fiwix/keyboard.h>
26: #include <fiwix/sched.h>
27: #include <fiwix/mm.h>
28:
29: unsigned int _last_data_addr;
30: int _memsize;
31: int _extmemsize;
32: int _rootdev;
33: int _noramdisk;
34: int _ramdisksize;
35: char _rootfstype[10];
36: char _rootdevname[DEVNAME_MAX + 1];
37: char _initrd[DEVNAME_MAX + 1];
38: int _syscondev;
39: char *init_args;
40:
41: char cmdline[NAME_MAX + 1];
42:
43: struct new_utsname sys_utsname = {
44:     UTS_SYSNAME,
45:     UTS_NODENAME,
46:     UTS_RELEASE,
47:     UTS_VERSION,
48:     "",
49:     UTS_DOMAINNAME,
50: };
51:
52: struct kernel_stat kstat;
53:
54: void start_kernel(unsigned long magic, unsigned long info, unsigned int stack)
55: {
56:     struct proc *init;
57:
58:     _last_data_addr = stack - KERNEL_BASE_ADDR;
59:     memset_b(&kstat, NULL, sizeof(kstat));
60:
61:     /* default kernel values */
62:     strcpy(_rootfstype, "ext2"); /* filesystem is ext2 */
63:     _syscondev = MKDEV(VCONSOLES_MAJOR, 0); /* console is /dev/tty0 */
64:
65:     pic_init();
66:     idt_init();
67:     dev_init();

```

kernel/main.c

Page 2/3

```

68:         tty_init();
69:
70:         printk("                                Welcome to %s\n", UTS_SYSNAME);
71:         printk("                                Copyright (c) 2018-2021, Jordi Sanfeliu\n")
;
72:         printk("\n");
73:         printk("                                kernel v%s for i386 architecture\n", UTS_R
RELEASE);
74:         printk("                                (GCC %s, built on %s)\n", __VERSION__, UTS_VERSIO
N);
75:         printk("\n");
76:         printk("DEVICE          ADDRESS          IRQ          COMMENT\n");
77:         printk("-----\n");
-----\n");
78:
79:         cpu_init();
80:         multiboot(magic, info);
81:         mem_init();
82:         video_init();
83:         console_init();
84:         timer_init();
85:         keyboard_init();
86:         proc_init();
87:
88:         /*
89:          * IDLE is now the current process (created manually as PID 0),
90:          * it won't be placed in the running queue.
91:          */
92:         current = get_proc_free();
93:         proc_slot_init(current);
94:         set_tss(current);
95:         load_tr(TSS);
96:         current->tss.cr3 = (unsigned int)kpage_dir;
97:         current->flags |= PF_KPROC;
98:         sprintf(current->argv0, "%s", "idle");
99:
100:        /* PID 1 is for the INIT process */
101:        init = get_proc_free();
102:        proc_slot_init(init);
103:        init->pid = get_unused_pid();
104:
105:        /* PID 2 is for the kswapd process */
106:        kernel_process("kswapd", kswapd);
107:
108:        /* kswapd will take over the rest of the kernel initialization */
109:        need_resched = 1;
110:
111:        STI();          /* let's rock! */
112:        cpu_idle();
113: }
114:
115: void stop_kernel(void)
116: {
117:     struct proc *p;
118:     int n;
119:
120:     /* stop and disable all interrupts! */
121:     CLI();
122:     for(n = 0; n < NR_IRQS; n++) {
123:         disable_irq(n);
124:     }
125:
126:     printk("\n");
127:     printk("***      Safe to Power Off      **\n");
128:     printk("                -or-\n");
129:     printk("*** Press Any Key to Reboot **\n");
130:     any_key_to_reboot = 1;

```

kernel/main.c

Page 3/3

```
131:
132:     /* put all processes to sleep and reset all pending signals */
133:     FOR_EACH_PROCESS_RUNNING(p) {
134:         not_runnable(p, PROC_SLEEPING);
135:         p->sigpending = 0;
136:         p = p->next_run;
137:     }
138:
139:     /* enable keyboard only */
140:     STI();
141:     enable_irq(KEYBOARD_IRQ);
142:
143:     /* switch to IDLE process */
144:     if(current) {
145:         do_sched();
146:     }
147:
148:     cpu_idle();
149: }
150:
151: void cpu_idle()
152: {
153:     for(;;) {
154:         if(need_resched) {
155:             do_sched();
156:         }
157:         HLT();
158:     }
159: }
```

kernel/Makefile

Page 1/1

```
1: # fiwix/kernel/Makefile
2: #
3: # Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
4: # Distributed under the terms of the Fiwix License.
5: #
6:
7: .S.o:
8:     $(CC) -traditional -I$(INCLUDE) -c -o $@ $<
9: .c.o:
10:     $(CC) $(CFLAGS) -c -o $@ $<
11:
12: OBJS = boot.o core386.o main.o init.o gdt.o idt.o syscalls.o pic.o pit.o \
13:     traps.o cpu.o cmos.o timer.o sched.o sleep.o signal.o process.o \
14:     multiboot.o
15:
16: kernel: $(OBJS)
17:     $(LD) $(LDFLAGS) -r $(OBJS) -o kernel.o
18:
19: clean:
20:     rm -f *.o
21:
```

kernel/multiboot.c

Page 1/5

```

1: /*
2:  * fiwix/kernel/multiboot.c
3:  *
4:  * Copyright 2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/config.h>
9: #include <fiwix/kernel.h>
10: #include <fiwix/multiboot1.h>
11: #include <fiwix/stdio.h>
12: #include <fiwix/string.h>
13: #include <fiwix/limits.h>
14: #include <fiwix/kparms.h>
15: #include <fiwix/i386elf.h>
16: #include <fiwix/ramdisk.h>
17: #include <fiwix/mm.h>
18: #include <fiwix/bios.h>
19: #include <fiwix/vgacon.h>
20: #include <fiwix/fb.h>
21: #include <fiwix/fbcon.h>
22:
23: Elf32_Shdr *symtab, *strtab;
24:
25: /* check the validity of a command line parameter */
26: static int check_parm(struct kparms *parm, const char *value)
27: {
28:     int n;
29:
30:     if(!strcmp(parm->name, "root=")) {
31:         for(n = 0; parm->value[n]; n++) {
32:             if(!strcmp(parm->value[n], value)) {
33:                 _rootdev = parm->sysval[n];
34:                 strncpy(_rootdevname, value, DEVNAME_MAX);
35:                 return 0;
36:             }
37:         }
38:         return 1;
39:     }
40:     if(!strcmp(parm->name, "noramdisk")) {
41:         _noramdisk = 1;
42:         return 0;
43:     }
44:     if(!strcmp(parm->name, "ramdisksize=")) {
45:         int size = atoi(value);
46:         if(!size || size > RAMDISK_MAXSIZE) {
47:             printk("WARNING: 'ramdisksize' value is out of limits, d
efaulting to 4096KB.\n");
48:             _ramdisksize = 0;
49:         } else {
50:             _ramdisksize = size;
51:         }
52:         return 0;
53:     }
54:     if(!strcmp(parm->name, "initrd=")) {
55:         if(value[0]) {
56:             strncpy(_initrd, value, DEVNAME_MAX);
57:             return 0;
58:         }
59:     }
60:     if(!strcmp(parm->name, "rootfstype=")) {
61:         for(n = 0; parm->value[n]; n++) {
62:             if(!strcmp(parm->value[n], value)) {
63:                 strncpy(_rootfstype, value, sizeof(_rootfstype))
;
64:                 return 0;
65:             }

```

kernel/multiboot.c

Page 2/5

```

66:         }
67:         return 1;
68:     }
69:     if(!strcmp(parm->name, "console=")) {
70:         for(n = 0; parm->value[n]; n++) {
71:             if(!strcmp(parm->value[n], value)) {
72:                 if(parm->sysval[n]) {
73:                     _syscondev = parm->sysval[n];
74:                     return 0;
75:                 }
76:                 printk("WARNING: device name for '%s' is not def
ined!\n", parm->name);
77:             }
78:         }
79:         return 1;
80:     }
81:     printk("WARNING: the parameter '%s' looks valid but it's not defined!\n"
, parm->name);
82:     return 0;
83: }
84:
85: static int parse_arg(const char *arg)
86: {
87:     int n;
88:
89:     /* '--' marks the beginning of the init arguments */
90:     if(!strcmp(arg, "--")) {
91:         return 1;
92:     }
93:
94:     for(n = 0; parm_table[n].name; n++) {
95:         if(!strncmp(arg, parm_table[n].name, strlen(parm_table[n].name))
) {
96:             arg += strlen(parm_table[n].name);
97:             if(check_parm(&parm_table[n], arg)) {
98:                 printk("WARNING: invalid value '%s' in the '%s'
parameter.\n", arg, parm_table[n].name);
99:             }
100:            return 0;
101:        }
102:    }
103:    printk("WARNING: invalid cmdline parameter: '%s'.\n", arg);
104:    return 0;
105: }
106:
107: static char * parse_cmdline(const char *str)
108: {
109:     char *from, *to;
110:     char arg[CMDL_ARG_LEN];
111:     char c;
112:
113:     from = to = (char *)str;
114:     for(;;) {
115:         c = *(str++);
116:         if(c == ' ' || !c) {
117:             if(to - from < CMDL_ARG_LEN) {
118:                 memcpy_b(arg, from, to - from);
119:                 arg[to - from] = NULL;
120:                 if(arg[0] != NULL) {
121:                     if(parse_arg(arg)) {
122:                         while(*(from++)) {
123:                             if(*from != '-' && *from
!= ' ') {
124:                                 break;
125:                             }
126:                         }
127:                     return from;

```


kernel/multiboot.c

Page 3/5

```

128:                                     }
129:                                     }
130:                                     } else {
131:                                     memcopy_b(arg, from, CMDL_ARG_LEN);
132:                                     arg[CMDL_ARG_LEN - 1] = NULL;
133:                                     printk("WARNING: invalid length of the cmdline p
parameter '%s'.\n", arg);
134:                                     }
135:                                     from = ++to;
136:                                     if(!c) {
137:                                     break;
138:                                     }
139:                                     continue;
140:                                     }
141:                                     to++;
142:                                     }
143:
144:                                     return NULL;
145: }
146:
147: /*
148:  * This function returns the last address used by kernel symbols or the value
149:  * of 'mod_end' (in the module structure) of the last module loaded by GRUB.
150:  *
151:  * This is intended to setup the kernel stack beyond all these addresses.
152:  */
153: unsigned int get_last_boot_addr(unsigned int info)
154: {
155:     struct multiboot_info *mbi;
156:     Elf32_Shdr *shdr;
157:     struct multiboot_elf_section_header_table *hdr;
158:     struct multiboot_mod_list *mod;
159:     unsigned short int n;
160:     unsigned int addr;
161:
162:     symtab = strtab = NULL;
163:     mbi = (struct multiboot_info *)info;
164:     hdr = &(mbi->u.elf_sec);
165:     for(n = 0; n < hdr->num; n++) {
166:         shdr = (Elf32_Shdr *) (hdr->addr + (n * hdr->size));
167:         if(shdr->sh_type == SHT_SYMTAB) {
168:             symtab = shdr;
169:         }
170:         if(shdr->sh_type == SHT_STRTAB) {
171:             strtab = shdr;
172:         }
173:     }
174:
175:     addr = strtab->sh_addr + strtab->sh_size;
176:
177:     /*
178:      * https://www.gnu.org/software/grub/manual/multiboot/multiboot.html
179:      *
180:      * Check if GRUB has loaded some modules and, if so, get the last
181:      * address used by the last one.
182:      */
183:     if(mbi->flags & MULTIBOOT_INFO_MODS) {
184:         mod = (struct multiboot_mod_list *)mbi->mods_addr;
185:         for(n = 0; n < mbi->mods_count; n++, mod++) {
186:             addr = mod->mod_end;
187:         }
188:     }
189:
190:     return P2V(addr);
191: }
192:
193: void multiboot(unsigned long magic, unsigned long info)

```

kernel/multiboot.c

Page 4/5

```

194: {
195:     struct multiboot_info mbi;
196:
197:     memset_b(&video, 0, sizeof(struct video_parms));
198:
199:     if(magic != MULTIBOOT_BOOTLOADER_MAGIC) {
200:         printk("WARNING: invalid multiboot magic number: 0x%x. Assuming
4MB of RAM.\n", (unsigned long int)magic);
201:         memset_b(&mbi, NULL, sizeof(struct multiboot_info));
202:         _memsize = 640;
203:         _extmemsize = 3072;
204:         bios_map_init(NULL, 0);
205:         video.columns = 80;
206:         video.lines = 25;
207:         video.flags = VPF_VGA;
208:         video.memsize = 384 * 1024;
209:         return;
210:     }
211:
212:     memcpy_b(&mbi, (void *)info, sizeof(struct multiboot_info));
213:
214:     if(mbi.flags & MULTIBOOT_INFO_BOOT_LOADER_NAME) {
215:         printk("bootloader           -   %s\n", mbi.boot_loader_n
ame);
216:     }
217:
218:     if(!(mbi.flags & MULTIBOOT_INFO_MEMORY)) {
219:         printk("WARNING: values in mem_lower and mem_upper are not valid
!\n");
220:     }
221:     _memsize = (unsigned int)mbi.mem_lower;
222:     _extmemsize = (unsigned int)mbi.mem_upper;
223:
224:
225:     if(mbi.flags & MULTIBOOT_INFO_CMDLINE) {
226:         int n, len;
227:         char c;
228:         char *p;
229:
230:         p = (char *)mbi.cmdline;
231:         len = strlen(p);
232:         /* suppress 'fiwix' */
233:         for(n = 0; n < len; n++) {
234:             c = *(p++);
235:             if(c == ' ') {
236:                 break;
237:             }
238:         }
239:         strcpy(cmdline, p);
240:         init_args = parse_cmdline(cmdline);
241:     } else {
242:         printk("WARNING: no cmdline detected!\n");
243:     }
244:     printk("kernel      0x%08X      -   cmdline=' %s' \n", KERNEL_ENTRY_ADDR,
cmdline);
245:
246:
247:     if(mbi.flags & MULTIBOOT_INFO_MODS) {
248:         int n;
249:         struct multiboot_mod_list *mod;
250:
251:         mod = (struct multiboot_mod_list *)mbi.mods_addr;
252:         for(n = 0; n < mbi.mods_count; n++, mod++) {
253:             if(!strcmp((char *)mod->cmdline, _initrd)) {
254:                 printk("initrd      0x%08X-0x%08X file=' %s' size=%
dKB\n", mod->mod_start, mod->mod_end, mod->cmdline, (mod->mod_end - mod->mod_start) / 1
024);

```

kernel/multiboot.c

Page 5/5

```

255:                                ramdisk_table[0].addr = (char *)mod->mod_start;
256:                                }
257:                                }
258:                                }
259:
260:
261:    if(!(mbi.flags & MULTIBOOT_INFO_ELF_SHDR)) {
262:        printk("WARNING: ELF section header table is not valid!\n");
263:    }
264:
265:    if(mbi.flags & MULTIBOOT_INFO_MEM_MAP) {
266:        bios_map_init((struct multiboot_mmap_entry *)mbi.mmap_addr, mbi.
mmap_length);
267:    } else {
268:        bios_map_init(NULL, 0);
269:    }
270:
271:    if(mbi.flags & MULTIBOOT_INFO_VBE_INFO) {
272:        struct vbe_controller *vbec;
273:        struct vbe_mode *vbem;
274:        unsigned long int from, to;
275:
276:        vbec = (struct vbe_controller *)mbi.vbe_control_info;
277:        vbem = (struct vbe_mode *)mbi.vbe_mode_info;
278:
279:        video.flags = VPF_VESAFB;
280:        video.address = (unsigned int *)vbem->phys_base;
281:        video.port = 0;
282:        video.memsize = vbec->total_memory * vbem->win_size * 1024;
283:        strcpy((char *)video.signature, (char *)vbec->signature);
284:        video.columns = vbem->x_resolution / vbem->x_char_size;
285:        video.lines = vbem->y_resolution / vbem->y_char_size;
286:        video.fb_version = vbec->version;
287:        video.fb_width = vbem->x_resolution;
288:        video.fb_height = vbem->y_resolution;
289:        video.fb_char_width = vbem->x_char_size;
290:        video.fb_char_height = vbem->y_char_size;
291:        video.fb_bpp = vbem->bits_per_pixel;
292:        video.fb_pixelwidth = vbem->bits_per_pixel / 8;
293:        video.fb_pitch = vbem->bytes_per_scanline;
294:        video.fb_linesize = video.fb_pitch * video.fb_char_height;
295:        video.fb_size = vbem->x_resolution * vbem->y_resolution * video.
fb_pixelwidth;
296:        video.fb_vsize = video.lines * video.fb_pitch * video.fb_char_he
ight;
297:
298:        from = (unsigned long int)video.address;
299:        to = from + video.memsize;
300:        bios_map_add(from, to, MULTIBOOT_MEMORY_AVAILABLE, MULTIBOOT_MEM
ORY_AVAILABLE);
301:        from = (unsigned long int)video.address - KERNEL_BASE_ADDR;
302:        to = (from + video.memsize);
303:        bios_map_add(from, to, MULTIBOOT_MEMORY_AVAILABLE, MULTIBOOT_MEM
ORY_RESERVED);
304:    } else {
305:        video.columns = 80;
306:        video.lines = 25;
307:        video.flags = VPF_VGA;
308:        video.memsize = 384 * 1024;
309:    }
310: }

```

kernel/pic.c

Page 1/4

```

1: /*
2:  * fiwix/kernel/pic.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/kernel.h>
10: #include <fiwix/config.h>
11: #include <fiwix/limits.h>
12: #include <fiwix/errno.h>
13: #include <fiwix/pic.h>
14: #include <fiwix/stdio.h>
15: #include <fiwix/string.h>
16: #include <fiwix/sigcontext.h>
17: #include <fiwix/sleep.h>
18:
19: /* interrupt vector base addresses */
20: #define IRQ0_ADDR      0x20
21: #define IRQ8_ADDR     0x28
22:
23: struct interrupt *irq_table[NR_IRQS];
24: static struct bh *bh_table = NULL;
25:
26: /*
27:  * This sends the command OCW3 to PIC (master or slave) to obtain the register
28:  * values. Slave is chained and represents IRQs 8-15. Master represents IRQs
29:  * 0-7, with IRQ 2 being the chain.
30:  */
31: static unsigned short int pic_get_irq_reg(int ocw3)
32: {
33:     outport_b(PIC_MASTER, ocw3);
34:     outport_b(PIC_SLAVE, ocw3);
35:     return (inport_b(PIC_SLAVE) << 8) | inport_b(PIC_MASTER);
36: }
37:
38: int register_irq(int num, struct interrupt *new_irq)
39: {
40:     struct interrupt **irq;
41:
42:     if(num < 0 || num >= NR_IRQS) {
43:         return -EINVAL;
44:     }
45:
46:     irq = &irq_table[num];
47:
48:     while(*irq) {
49:         if(*irq == new_irq) {
50:             printk("WARNING: %s(): interrupt %d already registered!\n",
n", __FUNCTION__, num);
51:             return -EINVAL;
52:         }
53:         irq = &(*irq)->next;
54:     }
55:     *irq = new_irq;
56:     new_irq->ticks = 0;
57:     return 0;
58: }
59:
60: int unregister_irq(int num, struct interrupt *old_irq)
61: {
62:     struct interrupt **irq, *prev_irq;
63:
64:     if(num < 0 || num >= NR_IRQS) {
65:         return -EINVAL;
66:     }

```

kernel/pic.c

Page 2/4

```

67:
68:     irq = &irq_table[num];
69:     prev_irq = NULL;
70:
71:     while(*irq) {
72:         if(*irq == old_irq) {
73:             if((*irq)->next) {
74:                 printk("WARNING: %s(): cannot unregister interrupt
pt %d.\n", __FUNCTION__, num);
75:                 return -EINVAL;
76:             }
77:             *irq = NULL;
78:             if(prev_irq) {
79:                 prev_irq->next = NULL;
80:             }
81:             break;
82:         }
83:         prev_irq = *irq;
84:         irq = &(*irq)->next;
85:     }
86:     return 0;
87: }
88:
89: void add_bh(struct bh *new)
90: {
91:     unsigned long int flags;
92:     struct bh **b;
93:
94:     SAVE_FLAGS(flags); CLI();
95:
96:     b = &bh_table;
97:     while(*b) {
98:         b = &(*b)->next;
99:     }
100:    *b = new;
101:
102:    RESTORE_FLAGS(flags);
103: }
104:
105: void enable_irq(int irq)
106: {
107:     int addr;
108:
109:     addr = (irq > 7) ? PIC_SLAVE + DATA : PIC_MASTER + DATA;
110:     irq &= 0x0007;
111:
112:     outport_b(addr, inport_b(addr) & ~(1 << irq));
113: }
114:
115: void disable_irq(int irq)
116: {
117:     int addr;
118:
119:     addr = (irq > 7) ? PIC_SLAVE + DATA : PIC_MASTER + DATA;
120:     irq &= 0x0007;
121:
122:     outport_b(addr, inport_b(addr) | (1 << irq));
123: }
124:
125: /* each ISR points to this function (interrupts are disabled) */
126: void irq_handler(int num, struct sigcontext sc)
127: {
128:     struct interrupt *irq;
129:     int real;
130:
131:     irq = irq_table[num];
132:

```

```

133:      /* spurious interrupt treatment */
134:      if(!irq) {
135:          real = pic_get_irq_reg(PIC_READ_ISR);
136:          if(!real) {
137:              /*
138:               * If IRQ was not real and came from slave, then send
139:               * an EOI to master because it doesn't know if the IRQ
140:               * was a spurious interrupt from slave.
141:               */
142:              if(num > 7) {
143:                  outport_b(PIC_MASTER, EOI);
144:              }
145:              if(kstat.sirqs < MAX_SPU_NOTICES) {
146:                  printk("WARNING: spurious interrupt detected (un
registered IRQ %d).\n", num);
147:              } else if(kstat.sirqs == MAX_SPU_NOTICES) {
148:                  printk("WARNING: too many spurious interrupts; n
ot logging any more.\n");
149:              }
150:              kstat.sirqs++;
151:              return;
152:          }
153:          if(num > 7) {
154:              outport_b(PIC_SLAVE, EOI);
155:          }
156:          outport_b(PIC_MASTER, EOI);
157:          return;
158:      }
159:
160:      if(num > 7) {
161:          outport_b(PIC_SLAVE, EOI);
162:      }
163:      outport_b(PIC_MASTER, EOI);
164:
165:      kstat.irqs++;
166:      irq->ticks++;
167:      do {
168:          irq->handler(num, &sc);
169:          irq = irq->next;
170:      } while(irq);
171: }
172:
173: void unknown_irq_handler(void)
174: {
175:     printk("Unknown IRQ received!\n");
176:     return;
177: }
178:
179: /* do bottom halves (interrupts are enabled) */
180: void do_bh(void)
181: {
182:     struct bh *b;
183:     void (*fn)(void);
184:
185:     if(!lock_area(AREA_BH)) {
186:         b = bh_table;
187:         while(b) {
188:             if(b->flags & BH_ACTIVE) {
189:                 b->flags &= ~BH_ACTIVE;
190:                 fn = b->fn;
191:                 (*fn)();
192:             }
193:             b = b->next;
194:         }
195:         unlock_area(AREA_BH);
196:     }
197: }

```

```
198:
199: void pic_init(void)
200: {
201:     memset_b(irq_table, NULL, sizeof(irq_table));
202:
203:     /* remap interrupts for PIC1 */
204:     outport_b(PIC_MASTER, ICW1_RESET);
205:     outport_b(PIC_MASTER + DATA, IRQ0_ADDR);          /* ICW2 */
206:     outport_b(PIC_MASTER + DATA, 1 << CASCADE_IRQ); /* ICW3 */
207:     outport_b(PIC_MASTER + DATA, ICW4_8086EOI);
208:
209:     /* remap interrupts for PIC2 */
210:     outport_b(PIC_SLAVE, ICW1_RESET);
211:     outport_b(PIC_SLAVE + DATA, IRQ8_ADDR);          /* ICW2 */
212:     outport_b(PIC_SLAVE + DATA, CASCADE_IRQ);        /* ICW3 */
213:     outport_b(PIC_SLAVE + DATA, ICW4_8086EOI);
214:
215:     /* mask all IRQs except cascade */
216:     outport_b(PIC_MASTER + DATA, ~(1 << CASCADE_IRQ));
217:
218:     /* mask all IRQs */
219:     outport_b(PIC_SLAVE + DATA, OCW1);
220: }
```

kernel/pit.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/pit.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/pit.h>
10:
11: void pit_beep_on(void)
12: {
13:     outport_b(MODEREG, SEL_CHAN2 | LSB_MSB | SQUARE_WAVE | BINARY_CTR);
14:     outport_b(CHANNEL2, (OSCIL / BEEP_FREQ) & 0xFF);      /* LSB */
15:     outport_b(CHANNEL2, (OSCIL / BEEP_FREQ) >> 8);      /* MSB */
16:     outport_b(PS2_SYSCTRL_B, inport_b(PS2_SYSCTRL_B) | ENABLE_SDATA | ENABLE
_LE_TMR2G);
17: }
18:
19: void pit_beep_off(unsigned int unused)
20: {
21:     outport_b(PS2_SYSCTRL_B, inport_b(PS2_SYSCTRL_B) & ~(ENABLE_SDATA | ENAB
LE_TMR2G));
22: }
23:
24: void pit_init(unsigned short int hertz)
25: {
26:     outport_b(MODEREG, SEL_CHAN0 | LSB_MSB | RATE_GEN | BINARY_CTR);
27:     outport_b(CHANNEL0, (OSCIL / hertz) & 0xFF);      /* LSB */
28:     outport_b(CHANNEL0, (OSCIL / hertz) >> 8);      /* MSB */
29: }
```


kernel/process.c

Page 1/7

```

1: /*
2:  * fiwix/kernel/process.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/mm.h>
10: #include <fiwix/errno.h>
11: #include <fiwix/process.h>
12: #include <fiwix/timer.h>
13: #include <fiwix/sched.h>
14: #include <fiwix/sleep.h>
15: #include <fiwix/stdio.h>
16: #include <fiwix/string.h>
17:
18: struct proc *proc_table;
19: struct proc *current;
20:
21: struct proc *proc_pool_head;
22: struct proc *proc_table_head;
23: struct proc *proc_table_tail;
24: unsigned int free_proc_slots = 0;
25:
26: static struct resource slot_resource = { NULL, NULL };
27: static struct resource pid_resource = { NULL, NULL };
28:
29: int nr_processes = 0;
30: __pid_t lastpid = 0;
31:
32: int kill_pid(__pid_t pid, __sigset_t signum)
33: {
34:     struct proc *p;
35:
36:     FOR_EACH_PROCESS(p) {
37:         if(p->pid == pid && p->state != PROC_ZOMBIE) {
38:             return send_sig(p, signum);
39:         }
40:         p = p->next;
41:     }
42:     return -ESRCH;
43: }
44:
45: int kill_pgrp(__pid_t pgid, __sigset_t signum)
46: {
47:     struct proc *p;
48:     int found;
49:
50:     found = 0;
51:     FOR_EACH_PROCESS(p) {
52:         if(p->pgid == pgid && p->state != PROC_ZOMBIE) {
53:             send_sig(p, signum);
54:             found = 1;
55:         }
56:         p = p->next;
57:     }
58:
59:     if(!found) {
60:         return -ESRCH;
61:     }
62:     return 0;
63: }
64:
65: /* sum up child (and its children) statistics */
66: void add_crusage(struct proc *p, struct rusage *cru)
67: {

```

kernel/process.c

Page 2/7

```

68:         cru->ru_utime.tv_sec = p->usage.ru_utime.tv_sec + p->cusage.ru_utime.tv_
sec;
69:         cru->ru_utime.tv_usec = p->usage.ru_utime.tv_usec + p->cusage.ru_utime.t
v_usec;
70:         if(cru->ru_utime.tv_usec >= 1000000) {
71:             cru->ru_utime.tv_sec++;
72:             cru->ru_utime.tv_usec -= 1000000;
73:         }
74:         cru->ru_stime.tv_sec = p->usage.ru_stime.tv_sec + p->cusage.ru_stime.tv_
sec;
75:         cru->ru_stime.tv_usec = p->usage.ru_stime.tv_usec + p->cusage.ru_stime.t
v_usec;
76:         if(cru->ru_stime.tv_usec >= 1000000) {
77:             cru->ru_stime.tv_sec++;
78:             cru->ru_stime.tv_usec -= 1000000;
79:         }
80:         cru->ru_maxrss = p->usage.ru_maxrss + p->cusage.ru_maxrss;
81:         cru->ru_ixrss = p->usage.ru_ixrss + p->cusage.ru_ixrss;
82:         cru->ru_idrss = p->usage.ru_idrss + p->cusage.ru_idrss;
83:         cru->ru_isrss = p->usage.ru_isrss + p->cusage.ru_isrss;
84:         cru->ru_minflt = p->usage.ru_minflt + p->cusage.ru_minflt;
85:         cru->ru_majflt = p->usage.ru_majflt + p->cusage.ru_majflt;
86:         cru->ru_nswap = p->usage.ru_nswap + p->cusage.ru_nswap;
87:         cru->ru_inblock = p->usage.ru_inblock + p->cusage.ru_inblock;
88:         cru->ru_oublock = p->usage.ru_oublock + p->cusage.ru_oublock;
89:         cru->ru_msgsnd = p->usage.ru_msgsnd + p->cusage.ru_msgsnd;
90:         cru->ru_msgrcv = p->usage.ru_msgrcv + p->cusage.ru_msgrcv;
91:         cru->ru_nsignals = p->usage.ru_nsignals + p->cusage.ru_nsignals;
92:         cru->ru_nvcsw = p->usage.ru_nvcsw + p->cusage.ru_nvcsw;
93:         cru->ru_nivcsw = p->usage.ru_nivcsw + p->cusage.ru_nivcsw;
94:     }
95:
96: void get_rusage(struct proc *p, struct rusage *ru)
97: {
98:     struct rusage cru;
99:
100:    /*
101:     * Conforms with SUSv3 which specifies that if SIGCHLD is being ignored
102:     * then the child statistics should not be added to the values returned
103:     * by RUSAGE_CHILDREN.
104:     */
105:    if(current->sigaction[SIGCHLD - 1].sa_handler == SIG_IGN) {
106:        return;
107:    }
108:
109:    add_crusage(p, &cru);
110:    memcpy_b(ru, &cru, sizeof(struct rusage));
111: }
112:
113: /* add child statistics to parent */
114: void add_rusage(struct proc *p)
115: {
116:     struct rusage cru;
117:
118:    add_crusage(p, &cru);
119:    current->cusage.ru_utime.tv_sec += cru.ru_utime.tv_sec;
120:    current->cusage.ru_utime.tv_usec += cru.ru_utime.tv_usec;
121:    if(current->cusage.ru_utime.tv_usec >= 1000000) {
122:        current->cusage.ru_utime.tv_sec++;
123:        current->cusage.ru_utime.tv_usec -= 1000000;
124:    }
125:    current->cusage.ru_stime.tv_sec += cru.ru_stime.tv_sec;
126:    current->cusage.ru_stime.tv_usec += cru.ru_stime.tv_usec;
127:    if(current->cusage.ru_stime.tv_usec >= 1000000) {
128:        current->cusage.ru_stime.tv_sec++;
129:        current->cusage.ru_stime.tv_usec -= 1000000;
130:    }

```

kernel/process.c

Page 3/7

```

131:     current->cusage.ru_maxrss += cru.ru_maxrss;
132:     current->cusage.ru_ixrss += cru.ru_ixrss;
133:     current->cusage.ru_idrss += cru.ru_idrss;
134:     current->cusage.ru_isrss += cru.ru_isrss;
135:     current->cusage.ru_minflt += cru.ru_minflt;
136:     current->cusage.ru_majflt += cru.ru_majflt;
137:     current->cusage.ru_nswap += cru.ru_nswap;
138:     current->cusage.ru_inblock += cru.ru_inblock;
139:     current->cusage.ru_oublock += cru.ru_oublock;
140:     current->cusage.ru_msgsnd += cru.ru_msgsnd;
141:     current->cusage.ru_msgrcv += cru.ru_msgrcv;
142:     current->cusage.ru_nsignals += cru.ru_nsignals;
143:     current->cusage.ru_nvcsw += cru.ru_nvcsw;
144:     current->cusage.ru_nivcsw += cru.ru_nivcsw;
145: }
146:
147: struct proc * get_next_zombie(struct proc *parent)
148: {
149:     struct proc *p;
150:
151:     if(proc_table_head == NULL) {
152:         PANIC("process table is empty!\n");
153:     }
154:
155:     FOR_EACH_PROCESS(p) {
156:         if(p->ppid == parent->pid && p->state == PROC_ZOMBIE) {
157:             return p;
158:         }
159:         p = p->next;
160:     }
161:
162:     return NULL;
163: }
164:
165: pid_t remove_zombie(struct proc *p)
166: {
167:     struct proc *pp;
168:     pid_t pid;
169:
170:     pid = p->pid;
171:     kfree(p->tss.esp0);
172:     p->rss--;
173:     kfree(P2V(p->tss.cr3));
174:     p->rss--;
175:     pp = get_proc_by_pid(p->ppid);
176:     release_proc(p);
177:     if(pp) {
178:         pp->children--;
179:     }
180:     return pid;
181: }
182:
183: /*
184:  * An orphaned process group is a process group in which the parent of every
185:  * member is either itself a member of the group or is not a member of the
186:  * group's session.
187:  */
188: int is_orphaned_pgrp(pid_t pgid)
189: {
190:     struct proc *p, *pp;
191:     int retval;
192:
193:     retval = 0;
194:     lock_resource(&slot_resource);
195:
196:     FOR_EACH_PROCESS(p) {
197:         if(p->pgid == pgid) {

```

kernel/process.c

Page 4/7

```

198:             if(p->state != PROC_ZOMBIE) {
199:                 pp = get_proc_by_pid(p->ppid);
200:                 /* return if only one is found that breaks the r
ule */
201:                 if(pp->pgid != pgid || pp->sid == p->sid) {
202:                     break;
203:                 }
204:             }
205:         }
206:         p = p->next;
207:     }
208:
209:     unlock_resource(&slot_resource);
210:     return retval;
211: }
212:
213: struct proc * get_proc_free(void)
214: {
215:     struct proc *p = NULL;
216:
217:     if(free_proc_slots <= SAFE_SLOTS && !IS_SUPERUSER) {
218:         printk("WARNING: %s(): the remaining slots are only for root use
r!\n", __FUNCTION__);
219:         return NULL;
220:     }
221:
222:     lock_resource(&slot_resource);
223:
224:     if(proc_pool_head) {
225:
226:         /* get (remove) a process slot from the free list */
227:         p = proc_pool_head;
228:         proc_pool_head = proc_pool_head->next;
229:
230:         free_proc_slots--;
231:     } else {
232:         printk("WARNING: %s(): no more slots free in proc table!\n", __F
UNCTION__);
233:     }
234:
235:     unlock_resource(&slot_resource);
236:     return p;
237: }
238:
239: void release_proc(struct proc *p)
240: {
241:     lock_resource(&slot_resource);
242:
243:     /* remove a process from the proc_table */
244:     if(p == proc_table_tail) {
245:         if(proc_table_head == proc_table_tail) {
246:             proc_table_head = proc_table_tail = NULL;
247:         } else {
248:             proc_table_tail = proc_table_tail->prev;
249:             proc_table_tail->next = NULL;
250:         }
251:     } else {
252:         p->prev->next = p->next;
253:         p->next->prev = p->prev;
254:     }
255:
256:     /* initialize and put a process slot back in the free list */
257:     memset_b(p, NULL, sizeof(struct proc));
258:     p->next = proc_pool_head;
259:     proc_pool_head = p;
260:     free_proc_slots++;
261:

```

kernel/process.c

Page 5/7

```

262:         unlock_resource(&slot_resource);
263:     }
264:
265: int get_unused_pid(void)
266: {
267:     short int loop;
268:     struct proc *p;
269:
270:     loop = 0;
271:     lock_resource(&pid_resource);
272:
273: loop:
274:     lastpid++;
275:     if(lastpid > MAX_PID_VALUE) {
276:         loop++;
277:         lastpid = INIT;
278:     }
279:     if(loop > 1) {
280:         printk("WARNING: %s(): system ran out of PID numbers!\n");
281:         return 0;
282:     }
283:     FOR_EACH_PROCESS(p) {
284:         /*
285:          * Make sure the kernel never reuses active pid, pgid
286:          * or sid values.
287:          */
288:         if(lastpid == p->pid || lastpid == p->pgid || lastpid == p->sid)
289:             goto loop;
290:     }
291:     p = p->next;
292: }
293:
294:     unlock_resource(&pid_resource);
295:     return lastpid;
296: }
297:
298: struct proc * get_proc_by_pid(__pid_t pid)
299: {
300:     struct proc *p;
301:
302:     FOR_EACH_PROCESS(p) {
303:         if(p->pid == pid) {
304:             return p;
305:         }
306:         p = p->next;
307:     }
308:
309:     return NULL;
310: }
311:
312: int get_new_user_fd(int fd)
313: {
314:     int n;
315:
316:     for(n = fd; n < OPEN_MAX && n < current->rlim[RLIMIT_NOFILE].rlim_cur; n
317: ++) {
318:         if(current->fd[n] == 0) {
319:             current->fd[n] = -1;
320:             current->fd_flags[n] = 0;
321:             return n;
322:         }
323:     }
324:     return -EMFILE;
325: }
326:

```

kernel/process.c

Page 6/7

```

327: void release_user_fd(int ufd)
328: {
329:     current->fd[ufd] = 0;
330: }
331:
332: struct proc * kernel_process(const char *name, int (*fn)(void))
333: {
334:     struct proc *p;
335:
336:     p = get_proc_free();
337:     proc_slot_init(p);
338:     p->pid = get_unused_pid();
339:     p->ppid = 0;
340:     p->flags |= PF_KPROC;
341:     p->priority = DEF_PRIORITY;
342:     if(!(p->tss.esp0 = kmalloc())) {
343:         release_proc(p);
344:         return NULL;
345:     }
346:     p->tss.esp0 += PAGE_SIZE - 4;
347:     p->rss++;
348:     p->tss.cr3 = (unsigned int)kpage_dir;
349:     p->tss.eip = (unsigned int)fn;
350:     p->tss.esp = p->tss.esp0;
351:     sprintk(p->pidstr, "%d", p->pid);
352:     sprintk(p->argv0, "%s", name);
353:     runnable(p);
354:     return p;
355: }
356:
357: void proc_slot_init(struct proc *p)
358: {
359:     int n;
360:
361:     /* insert process at the end of proc_table */
362:     lock_resource(&slot_resource);
363:     if(proc_table_head == NULL) {
364:         p->prev = NULL;
365:         p->next = NULL;
366:         proc_table_head = proc_table_tail = p;
367:     } else {
368:         p->prev = proc_table_tail;
369:         p->next = NULL;
370:         proc_table_tail->next = p;
371:         proc_table_tail = p;
372:     }
373:     p->prev_sleep = p->next_sleep = NULL;
374:     p->prev_run = p->next_run = NULL;
375:     unlock_resource(&slot_resource);
376:
377:     memset_b(&p->tss, NULL, sizeof(struct i386tss));
378:     p->tss.io_bitmap_addr = (unsigned int)&p->io_bitmap;
379:
380:     /*
381:      * Currently all io_bitmap bits are set to 0, which means full access.
382:      * This must be changed to 1 once we have fixed the ioperm() system
383:      * call.
384:      */
385:     for(n = 0; n < IO_BITMAP_SIZE + 1; n++) {
386:         p->io_bitmap[n] = 0; /* FIXME: change it to 1 */
387:     }
388:     p->state = PROC_IDLE;
389: }
390:
391: void proc_init(void)
392: {
393:     int n;

```

kernel/process.c

Page 7/7

```
394:         struct proc *p;
395:
396:         memset_b(proc_table, NULL, proc_table_size);
397:
398:         /* free list initialization */
399:         proc_pool_head = NULL;
400:         n = (proc_table_size / sizeof(struct proc)) - 1;
401:         do {
402:             p = &proc_table[n];
403:
404:             /* fill the free list */
405:             p->next = proc_pool_head;
406:             proc_pool_head = p;
407:             free_proc_slots++;
408:         } while (n--);
409:         proc_table_head = proc_table_tail = NULL;
410: }
```

kernel/sched.c

Page 1/2

```

1: /*
2:  * fiwix/kernel/sched.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/kernel.h>
10: #include <fiwix/sched.h>
11: #include <fiwix/process.h>
12: #include <fiwix/sleep.h>
13: #include <fiwix/segments.h>
14: #include <fiwix/timer.h>
15: #include <fiwix/pic.h>
16: #include <fiwix/stdio.h>
17: #include <fiwix/string.h>
18:
19: extern struct seg_desc gdt[NR_GDT_ENTRIES];
20: int need_resched = 0;
21:
22: static void context_switch(struct proc *next)
23: {
24:     struct proc *prev;
25:
26:     CLI();
27:     kstat.ctxt++;
28:     prev = current;
29:     set_tss(next);
30:     current = next;
31:     do_switch(&prev->tss.esp, &prev->tss.eip, next->tss.esp, next->tss.eip,
next->tss.cr3, TSS);
32:     STI();
33: }
34:
35: void set_tss(struct proc *p)
36: {
37:     struct seg_desc *g;
38:
39:     g = &gdt[TSS / sizeof(struct seg_desc)];
40:
41:     g->sd_lobase = (unsigned int)p;
42:     g->sd_loflags = SD_TSSPRESENT;
43:     g->sd_hibase = (char)(((unsigned int)p) >> 24);
44: }
45:
46: /* Round Robin algorithm */
47: void do_sched(void)
48: {
49:     int count;
50:     struct proc *p, *selected;
51:
52:     /* let the current running process consume its time slice */
53:     if(!need_resched && current->state == PROC_RUNNING && current->cpu_count
> 0) {
54:         return;
55:     }
56:
57:     need_resched = 0;
58:     for(;;) {
59:         count = -1;
60:         selected = &proc_table[IDLE];
61:
62:         FOR_EACH_PROCESS_RUNNING(p) {
63:             if(p->cpu_count > count) {
64:                 count = p->cpu_count;
65:                 selected = p;

```


kernel/sched.c

Page 2/2

```
66:             }
67:             p = p->next_run;
68:         }
69:         if(count) {
70:             break;
71:         }
72:
73:         /* reassigns new quantum to all running processes */
74:         FOR_EACH_PROCESS_RUNNING(p) {
75:             p->cpu_count = p->priority;
76:             p = p->next_run;
77:         }
78:     }
79:     if(current != selected) {
80:         context_switch(selected);
81:     }
82: }
83:
84: void sched_init(void)
85: {
86:     get_system_time();
87:
88:     /* this should be more unpredictable */
89:     kstat.random_seed = CURRENT_TIME;
90: }
```

kernel/signal.c

Page 1/4

```

1: /*
2:  * fiwix/kernel/signal.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/kernel.h>
10: #include <fiwix/errno.h>
11: #include <fiwix/process.h>
12: #include <fiwix/signal.h>
13: #include <fiwix/sigcontext.h>
14: #include <fiwix/sleep.h>
15: #include <fiwix/sched.h>
16: #include <fiwix/syscalls.h>
17: #include <fiwix/mm.h>
18: #include <fiwix/stdio.h>
19: #include <fiwix/string.h>
20:
21: int send_sig(struct proc *p, __sigset_t signum)
22: {
23:     if(signum > NSIG || !p) {
24:         return -EINVAL;
25:     }
26:
27:     if(!IS_SUPERUSER && current->euid != p->euid && current->sid != p->sid)
{
28:         return -EPERM;
29:     }
30:
31:     /* kernel processes can't receive signals */
32:     if(p->flags & PF_KPROC) {
33:         return 0;
34:     }
35:
36:     switch(signum) {
37:         case 0:
38:             return 0;
39:         case SIGKILL:
40:         case SIGCONT:
41:             if(p->state == PROC_STOPPED) {
42:                 runnable(p);
43:                 need_resched = 1;
44:             }
45:             /* discard all pending stop signals */
46:             p->sigpending &= SIG_MASK(SIGSTOP);
47:             p->sigpending &= SIG_MASK(SIGTSTP);
48:             p->sigpending &= SIG_MASK(SIGTTIN);
49:             p->sigpending &= SIG_MASK(SIGTTOU);
50:             break;
51:         case SIGSTOP:
52:         case SIGTSTP:
53:         case SIGTTIN:
54:         case SIGTTOU:
55:             /* discard all pending SIGCONT signals */
56:             p->sigpending &= SIG_MASK(SIGCONT);
57:             break;
58:         default:
59:             break;
60:     }
61:
62:     /*
63:      * Force some signals, that a process cannot ignore, by changing its
64:      * signal disposition to SIG_DFL.
65:      */
66:     switch(signum) {

```

kernel/signal.c

Page 2/4

```

67:         case SIGFPE:
68:         case SIGSEGV:
69:             if(p->sigaction[signum - 1].sa_handler == SIG_IGN) {
70:                 p->sigaction[signum - 1].sa_handler = SIG_DFL;
71:             }
72:             break;
73:     }
74:
75:     if(p->sigaction[signum - 1].sa_handler == SIG_DFL) {
76:         /*
77:          * INIT process is special, it only gets signals that have the
78:          * signal handler installed. This avoids to bring down the
79:          * system accidentally.
80:          */
81:         if(p->pid == INIT) {
82:             return 0;
83:         }
84:
85:         /* SIGCHLD is ignored by default */
86:         if(signum == SIGCHLD) {
87:             return 0;
88:         }
89:     }
90:
91:     if(p->sigaction[signum - 1].sa_handler == SIG_IGN) {
92:         /* if SIGCHLD is ignored reap its children (prevent zombies) */
93:         if(signum == SIGCHLD) {
94:             while(sys_waitpid(-1, NULL, WNOHANG) > 0) {
95:                 continue;
96:             }
97:         }
98:         return 0;
99:     }
100:
101:     p->sigpending |= 1 << (signum - 1);
102:     p->usage.ru_nsignals++;
103:
104:     /* wake up the process only if the signal is not blocked */
105:     if(!(p->sigblocked & (1 << (signum - 1)))) {
106:         wakeup_proc(p);
107:     }
108:
109:     return 0;
110: }
111:
112: int issig(void)
113: {
114:     __sigset_t signum;
115:     unsigned int mask;
116:     struct proc *p;
117:
118:     if(!(current->sigpending & ~current->sigblocked)) {
119:         return 0;
120:     }
121:
122:     for(signum = 1, mask = 1; signum < NSIG; signum++, mask <= 1) {
123:         if(current->sigpending & mask) {
124:             if(signum == SIGCHLD) {
125:                 if(current->sigaction[signum - 1].sa_handler ==
SIG_IGN) {
126:                     /* this process ignores SIGCHLD */
127:                     while((p = get_next_zombie(current))) {
128:                         remove_zombie(p);
129:                     }
130:                 } else {
131:                     if(current->sigaction[signum - 1].sa_han
dler != SIG_DFL) {

```

kernel/signal.c

Page 3/4

```

132:                                     return signal;
133:                                     }
134:                                 }
135:                                 } else {
136:                                     if(current->sigaction[signal - 1].sa_handler !=
SIG_IGN) {
137:                                         return signal;
138:                                     }
139:                                 }
140:                                 current->sigpending &= ~mask;
141:                             }
142:                         }
143:                         return 0;
144: }
145:
146: void psig(unsigned int stack)
147: {
148:     int len;
149:     __sigset_t signal;
150:     unsigned int mask;
151:     struct sigcontext *sc;
152:     struct proc *p;
153:
154:     sc = (struct sigcontext *)stack;
155:     for(signal = 1, mask = 1; signal < NSIG; signal++, mask <<= 1) {
156:         if(current->sigpending & mask) {
157:             current->sigpending &= ~mask;
158:
159:             if((unsigned int)current->sigaction[signal - 1].sa_handler) {
160:                 current->sigexecuting = mask;
161:                 if(!(current->sigaction[signal - 1].sa_flags & S
A_NODEFER)) {
162:                     current->sigblocked |= mask;
163:                 }
164:
165:                 /* save the current sigcontext */
166:                 memcpy_b(&current->sc[signal - 1], sc, sizeof(st
ruct sigcontext));
167:                 /* setup the jump to the user signal handler */
168:                 len = ((int)end_sighandler_trampoline - (int)sig
handler_trampoline);
169:                 sc->oldesp -= len;
170:                 sc->oldesp -= 4;
171:                 sc->oldesp &= ~3;          /* round up */
172:                 memcpy_b((void *)sc->oldesp, sighandler_trampoline, len);
173:                 sc->ecx = (unsigned int)current->sigaction[signal - 1].sa_handler;
174:                 sc->eax = signal;
175:                 sc->eip = sc->oldesp;
176:
177:                 if(current->sigaction[signal - 1].sa_flags & SA_
RESETHAND) {
178:                     current->sigaction[signal - 1].sa_handler
r = SIG_DFL;
179:                 }
180:                 return;
181:             }
182:             if(current->sigaction[signal - 1].sa_handler == SIG_DFL)
{
183:                 switch(signal) {
184:                     case SIGCONT:
185:                         runnable(current);
186:                         need_resched = 1;
187:                         break;
188:                     case SIGSTOP:

```

kernel/signal.c

Page 4/4

```

189:                                case SIGTSTP:
190:                                case SIGTTIN:
191:                                case SIGTTOU:
192:                                    current->exit_code = signum;
193:                                    not_runnable(current, PROC_STOPP
ED);
194:                                    if (!(current->sigaction[signum -
1].sa_flags & SA_NOCLDSTOP)) {
195:                                        if ((p = get_proc_by_pid(
current->ppid))) {
196:                                            send_sig(p, SIGC
HLD);
197:                                            /* needed for jo
b control */
198:                                            wakeup(&sys_wait
4);
199:                                        }
200:                                    }
201:                                    need_resched = 1;
202:                                    break;
203:                                case SIGCHLD:
204:                                    break;
205:                                default:
206:                                    do_exit(signum);
207:                                }
208:                            }
209:                    }
210:    }
211:
212:    /* coming from a system call that needs to be restarted */
213:    if (sc->err > 0) {
214:        if (sc->eax == -ERESTART) {
215:            sc->eax = sc->err;        /* syscall was saved in 'err' */
216:            sc->eip -= 2;            /* point again to 'int 0x80' */
217:        }
218:    }
219: }

```

kernel/sleep.c

Page 1/4

```

1: /*
2:  * fiwix/kernel/sleep.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/kernel.h>
10: #include <fiwix/limits.h>
11: #include <fiwix/sleep.h>
12: #include <fiwix/sched.h>
13: #include <fiwix/signal.h>
14: #include <fiwix/process.h>
15: #include <fiwix/stdio.h>
16: #include <fiwix/string.h>
17:
18: #define NR_BUCKETS          ((NR_PROCS * 10) / 100) /* 10% of NR_PROCS */
19: #define SLEEP_HASH(addr)   ((addr) % (NR_BUCKETS))
20:
21: struct proc *sleep_hash_table[NR_BUCKETS];
22: struct proc *proc_run_head;
23: static unsigned int area = 0;
24:
25: void runnable(struct proc *p)
26: {
27:     if(p->state == PROC_RUNNING) {
28:         printk("WARNING: %s(): process with pid '%d' is already running!
\n", __FUNCTION__, p->pid);
29:         return;
30:     }
31:
32:     if(proc_run_head) {
33:         p->next_run = proc_run_head;
34:         proc_run_head->prev_run = p;
35:     }
36:     proc_run_head = p;
37:     p->state = PROC_RUNNING;
38: }
39:
40: void not_runnable(struct proc *p, int state)
41: {
42:     if(p->next_run) {
43:         p->next_run->prev_run = p->prev_run;
44:     }
45:     if(p->prev_run) {
46:         p->prev_run->next_run = p->next_run;
47:     }
48:     if(p == proc_run_head) {
49:         proc_run_head = p->next_run;
50:     }
51:     p->prev_run = p->next_run = NULL;
52:     p->state = state;
53: }
54:
55: int sleep(void *address, int state)
56: {
57:     unsigned long int flags;
58:     struct proc **h;
59:     int signum, i;
60:
61:     /* return if it has signals */
62:     if(state == PROC_INTERRUPTIBLE) {
63:         if((signum = issig())) {
64:             return signum;
65:         }
66:     }

```

kernel/sleep.c

Page 2/4

```

67:
68:         if(current->state == PROC_SLEEPING) {
69:             printk("WARNING: %s(): process with pid '%d' is already sleeping
!\n", __FUNCTION__, current->pid);
70:             return 0;
71:         }
72:
73:         SAVE_FLAGS(flags); CLI();
74:         i = SLEEP_HASH((unsigned int) address);
75:         h = &sleep_hash_table[i];
76:
77:         /* insert process in the head */
78:         if(!*h) {
79:             *h = current;
80:             (*h)->prev_sleep = (*h)->next_sleep = NULL;
81:         } else {
82:             current->prev_sleep = NULL;
83:             current->next_sleep = *h;
84:             (*h)->prev_sleep = current;
85:             *h = current;
86:         }
87:         current->sleep_address = address;
88:         not_runnable(current, PROC_SLEEPING);
89:
90:         do_sched();
91:
92:         signum = 0;
93:         if(state == PROC_INTERRUPTIBLE) {
94:             signum = issig();
95:         }
96:
97:         RESTORE_FLAGS(flags);
98:         return signum;
99:     }
100:
101: void wakeup(void *address)
102: {
103:     unsigned long int flags;
104:     struct proc **h;
105:     int i;
106:
107:     SAVE_FLAGS(flags); CLI();
108:     i = SLEEP_HASH((unsigned int) address);
109:     h = &sleep_hash_table[i];
110:
111:     while(*h) {
112:         if((*h)->sleep_address == address) {
113:             (*h)->sleep_address = NULL;
114:             (*h)->cpu_count = (*h)->priority;
115:             runnable(*h);
116:             need_resched = 1;
117:             if((*h)->next_sleep) {
118:                 (*h)->next_sleep->prev_sleep = (*h)->prev_sleep;
119:             }
120:             if((*h)->prev_sleep) {
121:                 (*h)->prev_sleep->next_sleep = (*h)->next_sleep;
122:             }
123:             if(h == &sleep_hash_table[i]) { /* if it's the head */
124:                 *h = (*h)->next_sleep;
125:                 continue;
126:             }
127:         }
128:         if(*h) {
129:             h = &(*h)->next_sleep;
130:         }
131:     }
132:     RESTORE_FLAGS(flags);

```

kernel/sleep.c

Page 3/4

```

133: }
134:
135: void wakeup_proc(struct proc *p)
136: {
137:     unsigned long int flags;
138:     struct proc **h;
139:     int i;
140:
141:     if(p->state != PROC_SLEEPING && p->state != PROC_STOPPED) {
142:         return;
143:     }
144:
145:     SAVE_FLAGS(flags); CLI();
146:
147:     /* stopped processes don't have sleep address */
148:     if(p->sleep_address) {
149:         if(p->next_sleep) {
150:             p->next_sleep->prev_sleep = p->prev_sleep;
151:         }
152:         if(p->prev_sleep) {
153:             p->prev_sleep->next_sleep = p->next_sleep;
154:         }
155:
156:         i = SLEEP_HASH((unsigned int)p->sleep_address);
157:         h = &sleep_hash_table[i];
158:
159:         if(*h == p) { /* if it's the head */
160:             *h = (*h)->next_sleep;
161:         }
162:     }
163:     p->sleep_address = NULL;
164:     runnable(p);
165:     need_resched = 1;
166:
167:     RESTORE_FLAGS(flags);
168: }
169:
170: void lock_resource(struct resource *resource)
171: {
172:     unsigned long int flags;
173:
174:     for(;;) {
175:         SAVE_FLAGS(flags); CLI();
176:         if(resource->locked) {
177:             resource->wanted = 1;
178:             RESTORE_FLAGS(flags);
179:             sleep(resource, PROC_UNINTERRUPTIBLE);
180:         } else {
181:             break;
182:         }
183:     }
184:     resource->locked = 1;
185:     RESTORE_FLAGS(flags);
186: }
187:
188: void unlock_resource(struct resource *resource)
189: {
190:     unsigned long int flags;
191:
192:     SAVE_FLAGS(flags); CLI();
193:     resource->locked = 0;
194:     if(resource->wanted) {
195:         resource->wanted = 0;
196:         wakeup(resource);
197:     }
198:     RESTORE_FLAGS(flags);
199: }

```


kernel/sleep.c

Page 4/4

```
200:
201: int lock_area(unsigned int type)
202: {
203:     unsigned long int flags;
204:     int retval;
205:
206:     SAVE_FLAGS(flags); CLI();
207:     retval = area & type;
208:     area |= type;
209:     RESTORE_FLAGS(flags);
210:
211:     return retval;
212: }
213:
214: int unlock_area(unsigned int type)
215: {
216:     unsigned long int flags;
217:     int retval;
218:
219:     SAVE_FLAGS(flags); CLI();
220:     retval = area & type;
221:     area &= ~type;
222:     RESTORE_FLAGS(flags);
223:
224:     return retval;
225: }
226:
227: void sleep_init(void)
228: {
229:     proc_run_head = NULL;
230:     memset_b(sleep_hash_table, NULL, sizeof(sleep_hash_table));
231: }
```

kernel/syscalls.c

Page 1/7

```

1: /*
2:  * fiwix/kernel/syscalls.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/syscalls.h>
10: #include <fiwix/mm.h>
11: #include <fiwix/errno.h>
12: #include <fiwix/string.h>
13:
14: #ifdef __DEBUG__
15: #include <fiwix/stdio.h>
16: #endif /* __DEBUG__ */
17:
18: void free_name(const char *name)
19: {
20:     kfree((unsigned int)name);
21: }
22:
23: /*
24:  * This function has two objectives:
25:  * 1. to check the memory address validity of the char pointer supplied by the
26:  *    user, while at the same time limit its length to PAGE_SIZE (4096) bytes.
27:  * 2. to create a copy of 'filename' in the kernel data space before using it.
28:  */
29: int malloc_name(const char *filename, char **name)
30: {
31:     struct vma *vma;
32:     unsigned int start;
33:     short int n, len;
34:     char *b;
35:
36:     /*
37:      * Verifies if the 'vma' array of that process is not empty. It can
38:      * only be empty during the initialization of INIT, when it calls to
39:      * sys_execve and sys_open without having yet a proper setup.
40:      */
41:     if(current->vma[0].s_type != 0) {
42:         if(!filename) {
43:             return -EFAULT;
44:         }
45:         start = (unsigned int)filename;
46:         if(!(vma = find_vma_region(start))) {
47:             return -EFAULT;
48:         }
49:         if(!(vma->prot & PROT_READ)) {
50:             return -EFAULT;
51:         }
52:         len = MIN(vma->end - start, PAGE_SIZE);
53:         if(len < PAGE_SIZE) {
54:             if((vma = find_vma_region(vma->end))) {
55:                 if(vma->prot & PROT_READ) {
56:                     len = PAGE_SIZE;
57:                 }
58:             }
59:         }
60:     } else {
61:         len = PAGE_SIZE;
62:     }
63:     if(!(b = (char *)kmalloc())) {
64:         return -ENOMEM;
65:     }
66:     *name = b;
67:     for(n = 0; n < len; n++) {

```

kernel/syscalls.c

Page 2/7

```

68:         if(!(*b = *filename)) {
69:             return 0;
70:         }
71:         b++;
72:         filename++;
73:     }
74:
75:     free_name(*name);
76:     return -ENAMETOOLONG;
77: }
78:
79: int check_user_permission(struct inode *i)
80: {
81:     if(!IS_SUPERUSER) {
82:         if(current->euid != i->i_uid) {
83:             return 1;
84:         }
85:     }
86:     return 0;
87: }
88:
89: int check_group(struct inode *i)
90: {
91:     int n;
92:     __gid_t gid;
93:
94:     if(current->flags & PF_USEREAL) {
95:         gid = current->gid;
96:     } else {
97:         gid = current->egid;
98:     }
99:
100:    if(i->i_gid == gid) {
101:        return 0;
102:    }
103:
104:    for(n = 0; n < NGROUPS_MAX; n++) {
105:        if(current->groups[n] == -1) {
106:            break;
107:        }
108:        if(current->groups[n] == i->i_gid) {
109:            return 0;
110:        }
111:    }
112:    return 1;
113: }
114:
115: int check_user_area(int type, const void *addr, unsigned int size)
116: {
117:     struct vma *vma;
118:     unsigned int start;
119:
120:     /*
121:      * Verifies if the 'vma' array of that process is not empty. It can
122:      * only be empty during the initialization of INIT, when it calls to
123:      * sys_execve and sys_open without having yet a proper setup.
124:      */
125:     if(current->vma[0].s_type != 0) {
126:         start = (unsigned int)addr;
127:         if(!(vma = find_vma_region(start))) {
128:             return -EFAULT;
129:         }
130:
131:         for(;;) {
132:             if(type == VERIFY_WRITE) {
133:                 if(!(vma->prot & PROT_WRITE)) {
134:                     return -EFAULT;

```

kernel/syscalls.c

Page 3/7

```

135:         }
136:     } else {
137:         if(!(vma->prot & PROT_READ)) {
138:             return -EFAULT;
139:         }
140:     }
141:     if(start + size <= vma->end) {
142:         break;
143:     }
144:     if(!(vma = find_vma_region(vma->end))) {
145:         return -EFAULT;
146:     }
147: }
148: }
149:
150: return 0;
151: }
152:
153: int check_permission(int mask, struct inode *i)
154: {
155:     __uid_t uid;
156:
157:     if(current->flags & PF_USEREAL) {
158:         uid = current->uid;
159:     } else {
160:         uid = current->euid;
161:     }
162:
163:     if(uid == 0) {
164:         return 0;
165:     }
166:     if(i->i_uid == uid) {
167:         if((((i->i_mode >> 6) & 7) & mask) == mask) {
168:             return 0;
169:         }
170:     }
171:     if(!check_group(i)) {
172:         if((((i->i_mode >> 3) & 7) & mask) == mask) {
173:             return 0;
174:         }
175:     }
176:     if(((i->i_mode & 7) & mask) == mask) {
177:         return 0;
178:     }
179:
180:     return -EACCES;
181: }
182:
183: /* Linux 2.0.40 ABI system call (some from 2.2.26) */
184: void *syscall_table[] = {
185:     NULL, /* 0 */ /* sys_setup (-ENOSYS) */
186:     sys_exit,
187:     sys_fork,
188:     sys_read,
189:     sys_write,
190:     sys_open, /* 5 */
191:     sys_close,
192:     sys_waitpid,
193:     sys_creat,
194:     sys_link,
195:     sys_unlink, /* 10 */
196:     sys_execve,
197:     sys_chdir,
198:     sys_time,
199:     sys_mknod,
200:     sys_chmod, /* 15 */
201:     sys_chown,

```

kernel/syscalls.c

Page 4/7

```

202:         NULL,                                /* sys_break (-ENOSYS) */
203:         sys_stat,
204:         sys_lseek,
205:         sys_getpid,                            /* 20 */
206:         sys_mount,
207:         sys_umount,
208:         sys_setuid,
209:         sys_getuid,
210:         sys_stime,                            /* 25 */
211:         NULL, // sys_ptrace
212:         sys_alarm,
213:         sys_fstat,
214:         sys_pause,
215:         sys_utime,                            /* 30 */
216:         NULL,                                /* sys_stty (-ENOSYS) */
217:         NULL,                                /* sys_gtty (-ENOSYS) */
218:         sys_access,
219:         NULL, // sys_nice
220:         sys_ftime,                            /* 35 */
221:         sys_sync,
222:         sys_kill,
223:         sys_rename,
224:         sys_mkdir,
225:         sys_rmdir,                            /* 40 */
226:         sys_dup,
227:         sys_pipe,
228:         sys_times,
229:         NULL, // sys_prof
230:         sys_brk,                              /* 45 */
231:         sys_setgid,
232:         sys_getgid,
233:         sys_signal,
234:         sys_geteuid,
235:         sys_getegid,                          /* 50 */
236:         NULL, // sys_acct
237:         sys_umount2,
238:         NULL,                                /* sys_lock (-ENOSYS) */
239:         sys_ioctl,
240:         sys_fcntl,                            /* 55 */
241:         NULL,                                /* sys_mpx (-ENOSYS) */
242:         sys_setpgid,
243:         NULL,                                /* sys_ulimit (-ENOSYS) */
244:         sys_olduname,
245:         sys_umask,                            /* 60 */
246:         sys_chroot,
247:         sys_ustat,
248:         sys_dup2,
249:         sys_getppid,
250:         sys_getpgrp,                          /* 65 */
251:         sys_setsid,
252:         sys_sigaction,
253:         sys_sgetmask,
254:         sys_ssetmask,
255:         sys_setreuid,                         /* 70 */
256:         sys_setregid,
257:         sys_sigsuspend,
258:         sys_sigpending,
259:         sys_sethostname,
260:         sys_setrlimit,                       /* 75 */
261:         sys_getrlimit,
262:         sys_getrusage,
263:         sys_gettimeofday,
264:         sys_settimeofday,
265:         sys_getgroups,                        /* 80 */
266:         sys_setgroups,
267:         old_select,
268:         sys_symlink,

```

kernel/syscalls.c

Page 5/7

```

269:      sys_lstat,
270:      sys_readlink,                /* 85 */
271:      NULL, // sys_uselib
272:      NULL, // sys_swapon
273:      sys_reboot,
274:      NULL, // old_readdir
275:      old_mmap,                    /* 90 */
276:      sys_munmap,
277:      sys_truncate,
278:      sys_ftruncate,
279:      sys_fchmod,
280:      sys_fchown,                  /* 95 */
281:      NULL, // sys_getpriority
282:      NULL, // sys_setpriority
283:      NULL,                        /* sys_profil (-ENOSYS) */
284:      sys_statfs,
285:      sys_fstatfs,                 /* 100 */
286:      sys_ioperm,
287:      sys_socketcall, // sys_socketcall XXX
288:      NULL, // sys_syslog
289:      sys_setitimer,
290:      sys_getitimer,               /* 105 */
291:      sys_newstat,
292:      sys_newlstat,
293:      sys_newfstat,
294:      sys_uname,
295:      sys_iopl,                    /* 110 */
296:      NULL, // sys_vhangup
297:      NULL,                        /* sys_idle (-ENOSYS) */
298:      NULL, // sys_vm86old
299:      sys_wait4,
300:      NULL, // sys_swapoff         /* 115 */
301:      sys_sysinfo,
302:      NULL, // sys_ipc
303:      sys_fsync,
304:      sys_sigreturn,
305:      NULL, // sys_clone           /* 120 */
306:      sys_setdomainname,
307:      sys_newuname,
308:      NULL, // sys_modify_ldt
309:      NULL, // sys_adjtimex
310:      sys_mprotect,                /* 125 */
311:      sys_sigprocmask,
312:      NULL, // sys_create_module
313:      NULL, // sys_init_module
314:      NULL, // sys_delete_module
315:      NULL, // sys_get_kernel_syms /* 130 */
316:      NULL, // sys_quotactl
317:      sys_getpgid,
318:      sys_fchdir,
319:      NULL, // sys_bdflush
320:      NULL, // sys_sysfs           /* 135 */
321:      sys_personality,
322:      NULL,                        /* afs_syscall (-ENOSYS) */
323:      sys_setfsuid,
324:      sys_setfsgid,
325:      sys_llseek,                  /* 140 */
326:      sys_getdents,
327:      sys_select,
328:      sys_flock,
329:      NULL, // sys_msync
330:      NULL, // sys_readv           /* 145 */
331:      NULL, // sys_writev
332:      sys_getsid,
333:      sys_fdatasync,
334:      NULL, // sys_sysctl
335:      NULL, // sys_mlock           /* 150 */

```

kernel/syscalls.c

Page 6/7

```

336:     NULL,    // sys_munlock
337:     NULL,    // sys_mlockall
338:     NULL,    // sys_munlockall
339:     NULL,    // sys_sched_setparam
340:     NULL,    // sys_sched_getparam /* 155 */
341:     NULL,    // sys_sched_setscheduler
342:     NULL,    // sys_sched_getscheduler
343:     NULL,    // sys_sched_yield
344:     NULL,    // sys_sched_get_priority_max
345:     NULL,    // sys_sched_get_priority_min /* 160 */
346:     NULL,    // sys_sched_rr_get_interval
347:     sys_nanosleep,
348:     NULL,    // sys_mremap
349:
350:     NULL,
351:     NULL,
352:     NULL,
353:     NULL,
354:     NULL,
355:     NULL,
356:     NULL,
357:     NULL,
358:     NULL,
359:     NULL,
360:     NULL,
361:     NULL,
362:     NULL,
363:     NULL,
364:     NULL,
365:     NULL,
366:     NULL,
367:     NULL,
368:     sys_chown,          /* 182 */
369:     sys_getcwd,
370:     NULL,
371:     NULL,
372:     NULL,
373:     NULL,
374:     NULL,
375:     NULL,
376:     sys_fork,          /* 190 (sys_vfork) */
377: };
378:
379: static void do_bad_syscall(unsigned int num)
380: {
381: #ifdef __DEBUG__
382:     printk("***** (pid %d) system call %d not supported yet *****\n", current->pid, num);
383: #endif /* __DEBUG__ */
384: }
385:
386: /*
387:  * The argument 'struct sigcontext' is needed because there are some system
388:  * calls (such as sys_iopl and sys_fork) that need to get information from
389:  * certain registers (EFLAGS and ESP). The rest of system calls will ignore
390:  * such extra argument.
391:  */
392: int do_syscall(unsigned int num, int arg1, int arg2, int arg3, int arg4, int arg
5, struct sigcontext sc)
393: {
394:     int (*sys_func)(int, ...);
395:
396:     if(num > NR_SYSCALLS) {
397:         do_bad_syscall(num);
398:         return -ENOSYS;
399:     }
400:     sys_func = syscall_table[num];

```

kernel/syscalls.c

Page 7/7

```
401:         if(!sys_func) {
402:             do_bad_syscall(num);
403:             return -ENOSYS;
404:         }
405:         current->sp = (unsigned int)&sc;
406:         return sys_func(arg1, arg2, arg3, arg4, arg5, &sc);
407: }
```


kernel/timer.c

```

1: /*
2:  * fiwix/kernel/timer.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/kernel.h>
10: #include <fiwix/segments.h>
11: #include <fiwix/cmos.h>
12: #include <fiwix/pit.h>
13: #include <fiwix/timer.h>
14: #include <fiwix/time.h>
15: #include <fiwix/pic.h>
16: #include <fiwix/sched.h>
17: #include <fiwix/pic.h>
18: #include <fiwix/cmos.h>
19: #include <fiwix/signal.h>
20: #include <fiwix/process.h>
21: #include <fiwix/sleep.h>
22: #include <fiwix/errno.h>
23: #include <fiwix/stdio.h>
24: #include <fiwix/string.h>
25:
26: /*
27:  * timer.c implements a callout table using a singly linked list.
28:  *
29:  * head
30:  * +-----+ +-----+ ... +-----+
31:  * |data|next| |data|next| ... |data|next|
32:  * |-----> |-----> ... |-----> /
33:  * +-----+ +-----+ ... +-----+
34:  * (callout) (callout) ... (callout)
35:  */
36:
37: struct callout callout_pool[NR_CALLOUTS];
38: struct callout *callout_pool_head;
39: struct callout *callout_head;
40:
41: static char month[12] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
42: unsigned int avenrun[3] = { 0, 0, 0 };
43:
44: static struct bh timer_bh = { 0, &irq_timer_bh, NULL };
45: static struct bh callouts_bh = { 0, &do_callouts_bh, NULL };
46: static struct interrupt irq_config_timer = { 0, "timer", &irq_timer, NULL };
47:
48: static unsigned int count_active_procs(void)
49: {
50:     int counter;
51:     struct proc *p;
52:
53:     counter = 0;
54:     FOR_EACH_PROCESS_RUNNING(p) {
55:         counter += FIXED_1;
56:         p = p->next_run;
57:     }
58:     return counter;
59: }
60:
61: static void calc_load(void)
62: {
63:     unsigned int active_procs;
64:     static int count = LOAD_FREQ;
65:
66:     if(count-- > 0) {
67:         return;

```

```

68:         }
69:
70:         count = LOAD_FREQ;
71:         active_procs = count_active_procs();
72:         CALC_LOAD(avenrun[0], EXP_1, active_procs);
73:         CALC_LOAD(avenrun[1], EXP_5, active_procs);
74:         CALC_LOAD(avenrun[2], EXP_15, active_procs);
75:     }
76:
77:     static struct callout *get_free_callout(void)
78:     {
79:         struct callout *new;
80:
81:         new = NULL;
82:         if(callout_pool_head) {
83:             new = callout_pool_head;
84:             callout_pool_head = callout_pool_head->next;
85:             new->next = NULL;
86:         }
87:         return new;
88:     }
89:
90:     static void put_free_callout(struct callout *old)
91:     {
92:         old->next = callout_pool_head;
93:         callout_pool_head = old;
94:     }
95:
96:     static void do_del_callout(struct callout *c)
97:     {
98:         struct callout **tmp;
99:
100:        if(callout_head) {
101:            tmp = &callout_head;
102:            while(*tmp) {
103:                if((*tmp) == c) {
104:                    if((*tmp)->next != NULL) {
105:                        *tmp = (*tmp)->next;
106:                        (*tmp)->expires += c->expires;
107:                    } else {
108:                        *tmp = NULL;
109:                    }
110:                    put_free_callout(c);
111:                    break;
112:                }
113:                tmp = &(*tmp)->next;
114:            }
115:        }
116:    }
117:
118: void add_callout(struct callout_req *creq, unsigned int ticks)
119: {
120:     unsigned long int flags;
121:     struct callout *c, **tmp;
122:
123:     del_callout(creq);
124:     SAVE_FLAGS(flags); CLI();
125:
126:     if(!(c = get_free_callout())) {
127:         printk("WARNING: %s(): no more callout slots!\n", __FUNCTION__);
128:         RESTORE_FLAGS(flags);
129:         return;
130:     }
131:
132:     /* setup the new callout */
133:     memset_b(c, NULL, sizeof(struct callout));
134:     c->expires = ticks;

```

kernel/timer.c

Page 3/7

```

135:     c->fn = creq->fn;
136:     c->arg = creq->arg;
137:
138:     if(!callout_head) {
139:         callout_head = c;
140:     } else {
141:         tmp = &callout_head;
142:         while(*tmp) {
143:             if((*tmp)->expires > c->expires) {
144:                 (*tmp)->expires -= c->expires;
145:                 c->next = *tmp;
146:                 break;
147:             }
148:             c->expires -= (*tmp)->expires;
149:             tmp = &(*tmp)->next;
150:         }
151:         *tmp = c;
152:     }
153:     RESTORE_FLAGS(flags);
154: }
155:
156: void del_callout(struct callout_req *creq)
157: {
158:     unsigned long int flags;
159:     struct callout *c;
160:
161:     SAVE_FLAGS(flags); CLI();
162:     c = callout_head;
163:     while(c) {
164:         if(c->fn == creq->fn && c->arg == creq->arg) {
165:             do_del_callout(c);
166:             break;
167:         }
168:         c = c->next;
169:     }
170:     RESTORE_FLAGS(flags);
171: }
172:
173: void irq_timer(int num, struct sigcontext *sc)
174: {
175:     if((++kstat.ticks % HZ) == 0) {
176:         CURRENT_TIME++;
177:         kstat.uptime++;
178:     }
179:
180:     timer_bh.flags |= BH_ACTIVE;
181:
182:     if(sc->cs == KERNEL_CS) {
183:         current->usage.ru_stime.tv_usec += TICK;
184:         if(current->usage.ru_stime.tv_usec >= 1000000) {
185:             current->usage.ru_stime.tv_sec++;
186:             current->usage.ru_stime.tv_usec -= 1000000;
187:         }
188:         if(current->pid != IDLE) {
189:             kstat.cpu_system++;
190:         }
191:     } else {
192:         current->usage.ru_utime.tv_usec += TICK;
193:         if(current->usage.ru_utime.tv_usec >= 1000000) {
194:             current->usage.ru_utime.tv_sec++;
195:             current->usage.ru_utime.tv_usec -= 1000000;
196:         }
197:         if(current->pid != IDLE) {
198:             kstat.cpu_user++;
199:         }
200:         if(current->it_virt_value > 0) {
201:             current->it_virt_value--;

```

kernel/timer.c

Page 4/7

```

202:         if(!current->it_virt_value) {
203:             current->it_virt_value = current->it_virt_interv
al;
204:             send_sig(current, SIGVTALRM);
205:         }
206:     }
207: }
208: }
209:
210: unsigned long int tv2ticks(const struct timeval *tv)
211: {
212:     return((tv->tv_sec * HZ) + tv->tv_usec * HZ / 1000000);
213: }
214:
215: void ticks2tv(long int ticks, struct timeval *tv)
216: {
217:     tv->tv_sec = ticks / HZ;
218:     tv->tv_usec = (ticks % HZ) * 1000000 / HZ;
219: }
220:
221: int setitimer(int which, const struct itimerval *new_value, struct itimerval *ol
d_value)
222: {
223:     switch(which) {
224:         case ITIMER_REAL:
225:             if((unsigned int)old_value) {
226:                 ticks2tv(current->it_real_interval, &old_value->
it_interval);
227:                 ticks2tv(current->it_real_value, &old_value->it_
value);
228:             }
229:             current->it_real_interval = tv2ticks(&new_value->it_inte
rval);
230:             current->it_real_value = tv2ticks(&new_value->it_value);
231:             break;
232:         case ITIMER_VIRTUAL:
233:             if((unsigned int)old_value) {
234:                 ticks2tv(current->it_virt_interval, &old_value->
it_interval);
235:                 ticks2tv(current->it_virt_value, &old_value->it_
value);
236:             }
237:             current->it_virt_interval = tv2ticks(&new_value->it_inte
rval);
238:             current->it_virt_value = tv2ticks(&new_value->it_value);
239:             break;
240:         case ITIMER_PROF:
241:             if((unsigned int)old_value) {
242:                 ticks2tv(current->it_prof_interval, &old_value->
it_interval);
243:                 ticks2tv(current->it_prof_value, &old_value->it_
value);
244:             }
245:             current->it_prof_interval = tv2ticks(&new_value->it_inte
rval);
246:             current->it_prof_value = tv2ticks(&new_value->it_value);
247:             break;
248:         default:
249:             return -EINVAL;
250:     }
251:     return 0;
252: }
253: }
254:
255: unsigned long int mktime(struct mt *mt)
256: {
257:     int n, total_days;

```

kernel/timer.c

Page 5/7

```

258:         unsigned long int seconds;
259:
260:         total_days = 0;
261:
262:         for(n = UNIX_EPOCH; n < mt->mt_year; n++) {
263:             total_days += DAYS_PER_YEAR(n);
264:         }
265:         for(n = 0; n < (mt->mt_month - 1); n++) {
266:             total_days += month[n];
267:             if(n == 1) {
268:                 total_days += LEAP_YEAR(mt->mt_year) ? 1 : 0;
269:             }
270:         }
271:
272:         total_days += (mt->mt_day - 1);
273:         seconds = total_days * SECS_PER_DAY;
274:         seconds += mt->mt_hour * SECS_PER_HOUR;
275:         seconds += mt->mt_min * SECS_PER_MIN;
276:         seconds += mt->mt_sec;
277:         return seconds;
278:     }
279:
280: void irq_timer_bh(void)
281: {
282:     struct proc *p;
283:
284:     if(current->usage.ru_utime.tv_sec + current->usage.ru_stime.tv_sec > cur
rent->rlim[RLIMIT_CPU].rlim_cur) {
285:         send_sig(current, SIGXCPU);
286:     }
287:
288:     if(current->it_prof_value > 0) {
289:         current->it_prof_value--;
290:         if(!current->it_prof_value) {
291:             current->it_prof_value = current->it_prof_interval;
292:             send_sig(current, SIGPROF);
293:         }
294:     }
295:
296:     calc_load();
297:     FOR_EACH_PROCESS(p) {
298:         if(p->timeout > 0 && p->timeout < INFINITE_WAIT) {
299:             p->timeout--;
300:             if(!p->timeout) {
301:                 wakeup_proc(p);
302:             }
303:         }
304:         if(p->it_real_value > 0) {
305:             p->it_real_value--;
306:             if(!p->it_real_value) {
307:                 p->it_real_value = p->it_real_interval;
308:                 send_sig(p, SIGALRM);
309:             }
310:         }
311:         p = p->next;
312:     }
313:
314:     /* callouts */
315:     if(callout_head) {
316:         if(callout_head->expires > 0) {
317:             callout_head->expires--;
318:             if(!callout_head->expires) {
319:                 callouts_bh.flags |= BH_ACTIVE;
320:             }
321:         } else {
322:             printk("%s(): callout losing ticks.\n", __FUNCTION__);
323:             callouts_bh.flags |= BH_ACTIVE;

```

kernel/timer.c

Page 6/7

```
324:         }
325:     }
326:
327:     if(current->pid > IDLE && --current->cpu_count <= 0) {
328:         current->cpu_count = 0;
329:         need_resched = 1;
330:     }
331: }
332:
333: void do_callouts_bh(void)
334: {
335:     struct callout *c;
336:     void (*fn)(unsigned int);
337:     unsigned int arg;
338:
339:     while(callout_head) {
340:         if(callout_head->expires) {
341:             break;
342:         }
343:         if(lock_area(AREA_CALLOUT)) {
344:             continue;
345:         }
346:         fn = callout_head->fn;
347:         arg = callout_head->arg;
348:         c = callout_head;
349:         callout_head = callout_head->next;
350:         put_free_callout(c);
351:         unlock_area(AREA_CALLOUT);
352:         fn(arg);
353:     }
354: }
355:
356: void get_system_time(void)
357: {
358:     short int cmos_century;
359:     struct mt mt;
360:
361:     /* read date and time from CMOS */
362:     mt.mt_sec = cmos_read_date(CMOS_SEC);
363:     mt.mt_min = cmos_read_date(CMOS_MIN);
364:     mt.mt_hour = cmos_read_date(CMOS_HOUR);
365:     mt.mt_day = cmos_read_date(CMOS_DAY);
366:     mt.mt_month = cmos_read_date(CMOS_MONTH);
367:     mt.mt_year = cmos_read_date(CMOS_YEAR);
368:     cmos_century = cmos_read_date(CMOS_CENTURY);
369:     mt.mt_year += cmos_century * 100;
370:
371:     kstat.boot_time = CURRENT_TIME = mktime(&mt);
372: }
373:
374: void set_system_time(__time_t t)
375: {
376:     int sec, spm, min, hour, d, m, y;
377:
378:     sec = t;
379:     y = 1970;
380:     while(sec >= (DAYS_PER_YEAR(y) * SECS_PER_DAY)) {
381:         sec -= (DAYS_PER_YEAR(y) * SECS_PER_DAY);
382:         y++;
383:     }
384:
385:     m = 0;
386:     while(sec > month[m] * SECS_PER_DAY) {
387:         spm = month[m] * SECS_PER_DAY;
388:         if(m == 1) {
389:             spm = LEAP_YEAR(y) ? spm + SECS_PER_DAY : spm;
390:         }

```

kernel/timer.c

Page 7/7

```

391:         sec -= spm;
392:         m++;
393:     }
394:     m++;
395:
396:     d = 1;
397:     while(sec >= SECS_PER_DAY) {
398:         sec -= SECS_PER_DAY;
399:         d++;
400:     }
401:
402:     hour = 0;
403:     while(sec >= SECS_PER_HOUR) {
404:         sec -= SECS_PER_HOUR;
405:         hour++;
406:     }
407:
408:     min = 0;
409:     while(sec >= SECS_PER_MIN) {
410:         sec -= SECS_PER_MIN;
411:         min++;
412:     }
413:
414:     /* write date and time to CMOS */
415:     cmos_write_date(CMOS_SEC, sec);
416:     cmos_write_date(CMOS_MIN, min);
417:     cmos_write_date(CMOS_HOUR, hour);
418:     cmos_write_date(CMOS_DAY, d);
419:     cmos_write_date(CMOS_MONTH, m);
420:     cmos_write_date(CMOS_YEAR, y % 100);
421:     cmos_write_date(CMOS_CENTURY, (y - (y % 100)) / 100);
422:
423:     CURRENT_TIME = t;
424: }
425:
426: void timer_init(void)
427: {
428:     int n;
429:     struct callout *c;
430:
431:     add_bh(&timer_bh);
432:     add_bh(&callouts_bh);
433:
434:     pit_init(HZ);
435:
436:     memset_b(callout_pool, NULL, sizeof(callout_pool));
437:
438:     /* callout free list initialization */
439:     callout_pool_head = NULL;
440:     n = NR_CALLOUTS;
441:     while(n--) {
442:         c = &callout_pool[n];
443:         put_free_callout(c);
444:     }
445:     callout_head = NULL;
446:
447:     printk("clock      -           %d      type=PIT Hz=%d\n", TIMER_IRQ, H
Z);
448:     if(!register_irq(TIMER_IRQ, &irq_config_timer)) {
449:         enable_irq(TIMER_IRQ);
450:     }
451: }

```

kernel/traps.c

Page 1/6

```

1: /*
2:  * fiwix/kernel/traps.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/kernel.h>
10: #include <fiwix/traps.h>
11: #include <fiwix/cpu.h>
12: #include <fiwix/mm.h>
13: #include <fiwix/process.h>
14: #include <fiwix/signal.h>
15: #include <fiwix/stdio.h>
16: #include <fiwix/string.h>
17: #include <fiwix/sched.h>
18:
19: /*
20:  * PS/2 System Control Port B
21:  * -----
22:  * bit 7 -> system board RAM parity check
23:  * bit 6 -> channel check
24:  * bit 5 -> timer 2 (speaker time) output
25:  * bit 4 -> refresh request (toggle)
26:  * bit 3 -> channel check status
27:  * bit 2 -> parity check status
28:  * bit 1 -> speaker data status
29:  * bit 0 -> timer 2 gate to speaker status
30:  */
31: #define PS2_SYSCTRL_B 0x61 /* PS/2 system control port B (read) */
32:
33: struct traps traps_table[NR_EXCEPTIONS] = {
34:     { "Divide Error", do_divide_error, 0 },
35:     { "Debug", do_debug, 0 },
36:     { "NMI Interrupt", do_nmi_interrupt, 0 },
37:     { "Breakpoint", do_breakpoint, 0 },
38:     { "Overflow", do_overflow, 0 },
39:     { "BOUND Range Exceeded", do_bound, 0 },
40:     { "Invalid Opcode", do_invalid_opcode, 0 },
41:     { "Device Not Available (No Math Coprocessor)", do_no_math_coprocessor,
0 },
42:     { "Double Fault", do_double_fault, 1 },
43:     { "Coprocessor Segment Overrun", do_coprocessor_segment_overrun, 0 },
44:     { "Invalid TSS", do_invalid_tss, 1 },
45:     { "Segment Not Present", do_segment_not_present, 1 },
46:     { "Stack-Segment Fault", do_stack_segment_fault, 1 },
47:     { "General Protection", do_general_protection, 1 },
48:     { "Page Fault", do_page_fault, 1 },
49:     { "Intel reserved", do_reserved, 0 },
50:     { "x87 FPU Floating-Point Error", do_floating_point_error, 0 },
51:     { "Alignment Check", do_alignment_check, 1 },
52:     { "Machine Check", do_machine_check, 0 },
53:     { "SIMD Floating-Point Exception", do_simd_fault, 0 },
54:     { "Intel reserved", do_reserved, 0 },
55:     { "Intel reserved", do_reserved, 0 },
56:     { "Intel reserved", do_reserved, 0 },
57:     { "Intel reserved", do_reserved, 0 },
58:     { "Intel reserved", do_reserved, 0 },
59:     { "Intel reserved", do_reserved, 0 },
60:     { "Intel reserved", do_reserved, 0 },
61:     { "Intel reserved", do_reserved, 0 },
62:     { "Intel reserved", do_reserved, 0 },
63:     { "Intel reserved", do_reserved, 0 },
64:     { "Intel reserved", do_reserved, 0 },
65:     { "Intel reserved", do_reserved, 0 }
66: };

```


kernel/traps.c

Page 2/6

```
67:
68: void do_divide_error(unsigned int trap, struct sigcontext *sc)
69: {
70:     if(dump_registers(trap, sc)) {
71:         PANIC("");
72:     }
73:     send_sig(current, SIGFPE);
74:     return;
75: }
76:
77: void do_debug(unsigned int trap, struct sigcontext *sc)
78: {
79:     if(dump_registers(trap, sc)) {
80:         PANIC("");
81:     }
82:     send_sig(current, SIGTRAP);
83:     return;
84: }
85:
86: void do_nmi_interrupt(unsigned int trap, struct sigcontext *sc)
87: {
88:     unsigned char error;
89:
90:     error = inport_b(PS2_SYSCTRL_B);
91:
92:     printk("NMI received: ", error);
93:     switch(error) {
94:         case 0x80:
95:             printk("parity check occurred. Defective RAM chips?\n");
96:             break;
97:         default:
98:             printk("unknown error 0x%x\n", error);
99:             break;
100:    }
101:
102:    if(dump_registers(trap, sc)) {
103:        PANIC("");
104:    }
105:    send_sig(current, SIGSEGV);
106:    return;
107: }
108:
109: void do_breakpoint(unsigned int trap, struct sigcontext *sc)
110: {
111:     if(dump_registers(trap, sc)) {
112:         PANIC("");
113:     }
114:     send_sig(current, SIGTRAP);
115:     return;
116: }
117:
118: void do_overflow(unsigned int trap, struct sigcontext *sc)
119: {
120:     if(dump_registers(trap, sc)) {
121:         PANIC("");
122:     }
123:     send_sig(current, SIGSEGV);
124:     return;
125: }
126:
127: void do_bound(unsigned int trap, struct sigcontext *sc)
128: {
129:     if(dump_registers(trap, sc)) {
130:         PANIC("");
131:     }
132:     send_sig(current, SIGSEGV);
133:     return;
```

```
134: }
135:
136: void do_invalid_opcode(unsigned int trap, struct sigcontext *sc)
137: {
138:     if(dump_registers(trap, sc)) {
139:         PANIC("");
140:     }
141:     send_sig(current, SIGILL);
142:     return;
143: }
144:
145: void do_no_math_coprocessor(unsigned int trap, struct sigcontext *sc)
146: {
147:     /* floating-point emulation would go here */
148:
149:     if(dump_registers(trap, sc)) {
150:         PANIC("No coprocessor/emulation found.\n");
151:     }
152:     send_sig(current, SIGILL);
153:     return;
154: }
155:
156: void do_double_fault(unsigned int trap, struct sigcontext *sc)
157: {
158:     if(dump_registers(trap, sc)) {
159:         PANIC("");
160:     }
161:     send_sig(current, SIGSEGV);
162:     return;
163: }
164:
165: void do_coprocessor_segment_overrun(unsigned int trap, struct sigcontext *sc)
166: {
167:     if(dump_registers(trap, sc)) {
168:         PANIC("");
169:     }
170:     send_sig(current, SIGFPE);
171:     return;
172: }
173:
174: void do_invalid_tss(unsigned int trap, struct sigcontext *sc)
175: {
176:     if(dump_registers(trap, sc)) {
177:         PANIC("");
178:     }
179:     send_sig(current, SIGSEGV);
180:     return;
181: }
182:
183: void do_segment_not_present(unsigned int trap, struct sigcontext *sc)
184: {
185:     if(dump_registers(trap, sc)) {
186:         PANIC("");
187:     }
188:     send_sig(current, SIGBUS);
189:     return;
190: }
191:
192: void do_stack_segment_fault(unsigned int trap, struct sigcontext *sc)
193: {
194:     if(dump_registers(trap, sc)) {
195:         PANIC("");
196:     }
197:     send_sig(current, SIGBUS);
198:     return;
199: }
200:
```

kernel/traps.c

Page 4/6

```
201: void do_general_protection(unsigned int trap, struct sigcontext *sc)
202: {
203:     if(dump_registers(trap, sc)) {
204:         PANIC("");
205:     }
206:     send_sig(current, SIGSEGV);
207:     return;
208: }
209:
210: /* do_page_fault() resides in mm/fault.c */
211:
212: void do_reserved(unsigned int trap, struct sigcontext *sc)
213: {
214:     if(dump_registers(trap, sc)) {
215:         PANIC("");
216:     }
217:     send_sig(current, SIGSEGV);
218:     return;
219: }
220:
221: void do_floating_point_error(unsigned int trap, struct sigcontext *sc)
222: {
223:     if(dump_registers(trap, sc)) {
224:         PANIC("");
225:     }
226:     send_sig(current, SIGFPE);
227:     return;
228: }
229:
230: void do_alignment_check(unsigned int trap, struct sigcontext *sc)
231: {
232:     if(dump_registers(trap, sc)) {
233:         PANIC("");
234:     }
235:     send_sig(current, SIGSEGV);
236:     return;
237: }
238:
239: void do_machine_check(unsigned int trap, struct sigcontext *sc)
240: {
241:     if(dump_registers(trap, sc)) {
242:         PANIC("");
243:     }
244:     send_sig(current, SIGSEGV);
245:     return;
246: }
247:
248: void do_simd_fault(unsigned int trap, struct sigcontext *sc)
249: {
250:     if(dump_registers(trap, sc)) {
251:         PANIC("");
252:     }
253:     send_sig(current, SIGSEGV);
254:     return;
255: }
256:
257: void trap_handler(unsigned int trap, struct sigcontext sc)
258: {
259:     traps_table[trap].handler(trap, &sc);
260:
261:     /* avoids confusion with -RESTART return value */
262:     sc.err = -sc.err;
263: }
264:
265: const char * elf_lookup_symbol(unsigned int addr)
266: {
267:     Elf32_Sym *sym;
```

kernel/traps.c

Page 5/6

```

268:         unsigned int n;
269:
270:         sym = (Elf32_Sym *)syntab->sh_addr;
271:         for(n = 0; n < syntab->sh_size / sizeof(Elf32_Sym); n++, sym++) {
272:             if(ELF32_ST_TYPE(sym->st_info) != STT_FUNC) {
273:                 continue;
274:             }
275:             if(addr >= sym->st_value && addr < (sym->st_value + sym->st_size
)) {
276:                 return (const char *)strtab->sh_addr + sym->st_name;
277:             }
278:         }
279:         return NULL;
280:     }
281:
282: void stack_backtrace(void)
283: {
284:     int n;
285:     unsigned int addr, *sp;
286:     const char *str;
287:
288:     printk("Stack:\n");
289:     GET_ESP(sp);
290:     sp = (unsigned int *)sp;
291:     for(n = 1; n <= 32; n++) {
292:         printk(" %08x", *sp);
293:         sp++;
294:         if(!(n % 8)) {
295:             printk("\n");
296:         }
297:     }
298:     printk("Backtrace:\n");
299:     GET_ESP(sp);
300:     sp = (unsigned int *)sp;
301:     for(n = 0; n < 256; n++) {
302:         addr = *sp;
303:         str = elf_lookup_symbol(addr);
304:         if(str) {
305:             printk("<0x%08x> %s()\n", addr, str);
306:         }
307:         sp++;
308:     }
309: }
310:
311: int dump_registers(unsigned int trap, struct sigcontext *sc)
312: {
313:     unsigned int cr2;
314:
315:     printk("\n");
316:     if(trap == 14) { /* Page Fault */
317:         GET_CR2(cr2);
318:         printk("%s at 0x%08x (%s) with error code 0x%08x (0b%b)\n", trap
s_table[trap].name, cr2, sc->err & PFAULT_W ? "writing" : "reading", sc->err, sc->err);
319:     } else {
320:         printk("EXCEPTION: %s", traps_table[trap].name);
321:         if(traps_table[trap].errcode) {
322:             printk(": error code 0x%08x (0b%b)", sc->err, sc->err);
323:         }
324:         printk("\n");
325:     }
326:
327:     printk("Process '%s' with pid %d", current->argv0, current->pid);
328:     if(sc->cs == KERNEL_CS) {
329:         printk(" in '%s()'.", elf_lookup_symbol(sc->eip));
330:     }
331:     printk("\n");
332:

```

kernel/traps.c

Page 6/6

```
333:         printk(" cs: 0x%08x\teip: 0x%08x\tefl: 0x%08x\t ss: 0x%08x\tesp: 0x%08x\n", sc->cs, sc->eip, sc->eflags, sc->oldss, sc->oldesp);
334:         printk("eax: 0x%08x\tebx: 0x%08x\tecx: 0x%08x\tdx: 0x%08x\n", sc->eax, sc->ebx, sc->ecx, sc->edx);
335:         printk("esi: 0x%08x\tedi: 0x%08x\tesp: 0x%08x\tebp: 0x%08x\n", sc->esi, sc->edi, sc->esp, sc->ebp);
336:         printk(" ds: 0x%08x\t es: 0x%08x\t fs: 0x%08x\t gs: 0x%08x\n", sc->ds, sc->es, sc->fs, sc->gs);
337:
338:         if(sc->cs == KERNEL_CS) {
339:             stack_backtrace();
340:         }
341:
342:         /* panics if the exception has been in kernel mode */
343:         if(current->flags & PF_KPROC || sc->cs == KERNEL_CS) {
344:             return 1;
345:         }
346:
347:         return 0;
348: }
```

kernel/syscalls/access.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/access.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/stat.h>
11: #include <fiwix/errno.h>
12: #include <fiwix/string.h>
13:
14: #ifdef __DEBUG__
15: #include <fiwix/stdio.h>
16: #include <fiwix/process.h>
17: #endif /*__DEBUG__*/
18:
19: int sys_access(const char *filename, __mode_t mode)
20: {
21:     struct inode *i;
22:     char *tmp_name;
23:     int errno;
24:
25: #ifdef __DEBUG__
26:     printk("(pid %d) sys_access('%s', %d)", current->pid, filename, mode);
27: #endif /*__DEBUG__*/
28:
29:     if((mode & S_IRWXO) != mode) {
30:         return -EINVAL;
31:     }
32:     if((errno = malloc_name(filename, &tmp_name)) < 0) {
33:         return errno;
34:     }
35:     current->flags |= PF_USERREAL;
36:     if((errno = namei(tmp_name, &i, NULL, FOLLOW_LINKS))) {
37:         current->flags &= ~PF_USERREAL;
38:         free_name(tmp_name);
39:         return errno;
40:     }
41:     if(mode & TO_WRITE) {
42:         if(S_ISREG(i->i_mode) || S_ISDIR(i->i_mode) || S_ISLNK(i->i_mode
)) {
43:             if(IS_RDONLY_FS(i)) {
44:                 current->flags &= ~PF_USERREAL;
45:                 iput(i);
46:                 free_name(tmp_name);
47:                 return -EROFS;
48:             }
49:         }
50:     }
51:     errno = check_permission(mode, i);
52:
53: #ifdef __DEBUG__
54:     printk("-> returning %d\n", errno);
55: #endif /*__DEBUG__*/
56:
57:     current->flags &= ~PF_USERREAL;
58:     iput(i);
59:     free_name(tmp_name);
60:     return errno;
61: }

```

kernel/syscalls/alarm.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/alarm.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/time.h>
10:
11: #ifdef __DEBUG__
12: #include <fiwix/stdio.h>
13: #include <fiwix/process.h>
14: #endif /*__DEBUG__*/
15:
16: int sys_alarm(unsigned int secs)
17: {
18:     struct itimerval value, ovalue;
19:
20: #ifdef __DEBUG__
21:     printk("(pid %d) sys_alarm(%d)\n", current->pid, secs);
22: #endif /*__DEBUG__*/
23:
24:     value.it_interval.tv_sec = 0;
25:     value.it_interval.tv_usec = 0;
26:     value.it_value.tv_sec = secs;
27:     value.it_value.tv_usec = 0;
28:     setitimer(ITIMER_REAL, &value, &ovalue);
29:
30:     /*
31:      * If there are still some usecs left and since the return value has
32:      * not enough granularity for them, then just add 1 second to it.
33:      */
34:     if(ovalue.it_value.tv_usec) {
35:         ovalue.it_value.tv_sec++;
36:     }
37:
38:     return ovalue.it_value.tv_sec;
39: }
```

kernel/syscalls/brk.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/brk.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/process.h>
9: #include <fiwix/mm.h>
10: #include <fiwix/mman.h>
11: #include <fiwix/errno.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #endif /*__DEBUG__*/
16:
17: int sys_brk(unsigned int brk)
18: {
19:     unsigned int newbrk;
20:
21: #ifdef __DEBUG__
22:     printk("(pid %d) sys_brk(0x%08x) -> ", current->pid, brk);
23: #endif /*__DEBUG__*/
24:
25:     if(!brk || brk < current->brk_lower) {
26: #ifdef __DEBUG__
27:         printk("0x%08x\n", current->brk);
28: #endif /*__DEBUG__*/
29:         return current->brk;
30:     }
31:
32:     newbrk = PAGE_ALIGN(brk);
33:     if(newbrk == current->brk || newbrk < current->brk_lower) {
34: #ifdef __DEBUG__
35:         printk("0x%08x\n", current->brk);
36: #endif /*__DEBUG__*/
37:         return brk;
38:     }
39:
40:     if(brk < current->brk) {
41:         do_munmap(newbrk, current->brk - newbrk);
42:         current->brk = brk;
43:         return brk;
44:     }
45:     if(!expand_heap(newbrk)) {
46:         current->brk = brk;
47:     } else {
48:         return -ENOMEM;
49:     }
50: #ifdef __DEBUG__
51:     printk("0x%08x\n", current->brk);
52: #endif /*__DEBUG__*/
53:     return current->brk;
54: }

```


kernel/syscalls/chdir.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/chdir.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/stat.h>
10: #include <fiwix/errno.h>
11: #include <fiwix/string.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #include <fiwix/process.h>
16: #endif /* __DEBUG__ */
17:
18: int sys_chdir(const char *dirname)
19: {
20:     struct inode *i;
21:     char *tmp_name;
22:     int errno;
23:
24: #ifdef __DEBUG__
25:     printk("(pid %d) sys_chdir('%s')\n", current->pid, dirname);
26: #endif /* __DEBUG__ */
27:
28:     if((errno = malloc_name(dirname, &tmp_name)) < 0) {
29:         return errno;
30:     }
31:     if((errno = namei(tmp_name, &i, NULL, FOLLOW_LINKS)) {
32:         free_name(tmp_name);
33:         return errno;
34:     }
35:     if(!S_ISDIR(i->i_mode)) {
36:         iput(i);
37:         free_name(tmp_name);
38:         return -ENOTDIR;
39:     }
40:     if((errno = check_permission(TO_EXEC, i)) {
41:         iput(i);
42:         free_name(tmp_name);
43:         return errno;
44:     }
45:     iput(current->pwd);
46:     current->pwd = i;
47:     free_name(tmp_name);
48:     return 0;
49: }
```

kernel/syscalls/chmod.c

Page 1/1

```

1:  /*
2:  *  fiwix/kernel/syscalls/chmod.c
3:  *
4:  *  Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  *  Distributed under the terms of the Fiwix License.
6:  */
7:
8:  #include <fiwix/types.h>
9:  #include <fiwix/kernel.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/stat.h>
12: #include <fiwix/errno.h>
13: #include <fiwix/string.h>
14:
15: #ifdef __DEBUG__
16: #include <fiwix/stdio.h>
17: #include <fiwix/process.h>
18: #endif /*__DEBUG__ */
19:
20: int sys_chmod(const char *filename, __mode_t mode)
21: {
22:     struct inode *i;
23:     char *tmp_name;
24:     int errno;
25:
26: #ifdef __DEBUG__
27:     printk("(pid %d) sys_chmod('%s', %d)\n", current->pid, filename, mode);
28: #endif /*__DEBUG__ */
29:
30:     if((errno = malloc_name(filename, &tmp_name)) < 0) {
31:         return errno;
32:     }
33:     if((errno = namei(tmp_name, &i, NULL, FOLLOW_LINKS))) {
34:         free_name(tmp_name);
35:         return errno;
36:     }
37:
38:     if(IS_RDONLY_FS(i)) {
39:         iput(i);
40:         free_name(tmp_name);
41:         return -EROFS;
42:     }
43:     if(check_user_permission(i)) {
44:         iput(i);
45:         free_name(tmp_name);
46:         return -EPERM;
47:     }
48:
49:     i->i_mode &= S_IFMT;
50:     i->i_mode |= mode & ~S_IFMT;
51:     i->i_ctime = CURRENT_TIME;
52:     i->dirty = 1;
53:     iput(i);
54:     free_name(tmp_name);
55:     return 0;
56: }

```

kernel/syscalls/chown.c

Page 1/2

```

1: /*
2:  * fiwix/kernel/syscalls/chown.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/kernel.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/stat.h>
12: #include <fiwix/errno.h>
13: #include <fiwix/string.h>
14:
15: #ifdef __DEBUG__
16: #include <fiwix/stdio.h>
17: #include <fiwix/process.h>
18: #endif /*__DEBUG__ */
19:
20: int sys_chown(const char *filename, __uid_t owner, __gid_t group)
21: {
22:     struct inode *i;
23:     char *tmp_name;
24:     int errno;
25:
26: #ifdef __DEBUG__
27:     printk("(pid %d) sys_chown('%s', %d, %d)\n", current->pid, filename, own
er, group);
28: #endif /*__DEBUG__ */
29:
30:     if((errno = malloc_name(filename, &tmp_name)) < 0) {
31:         return errno;
32:     }
33:     if((errno = namei(tmp_name, &i, NULL, !FOLLOW_LINKS))) {
34:         free_name(tmp_name);
35:         return errno;
36:     }
37:
38:     if(IS_RDONLY_FS(i)) {
39:         iput(i);
40:         free_name(tmp_name);
41:         return -EROFS;
42:     }
43:     if(check_user_permission(i)) {
44:         iput(i);
45:         free_name(tmp_name);
46:         return -EPERM;
47:     }
48:
49:     if(owner == (__uid_t)-1) {
50:         owner = i->i_uid;
51:     } else {
52:         i->i_mode &= ~(S_ISUID);
53:         i->i_ctime = CURRENT_TIME;
54:     }
55:     if(group == (__gid_t)-1) {
56:         group = i->i_gid;
57:     } else {
58:         i->i_mode &= ~(S_ISGID);
59:         i->i_ctime = CURRENT_TIME;
60:     }
61:
62:     i->i_uid = owner;
63:     i->i_gid = group;
64:     i->dirty = 1;
65:     iput(i);
66:     free_name(tmp_name);

```

kernel/syscalls/chown.c

Page 2/2

```
67:         return 0;  
68:     }
```

kernel/syscalls/chroot.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/chroot.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/stat.h>
10: #include <fiwix/errno.h>
11: #include <fiwix/string.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #include <fiwix/process.h>
16: #endif /* __DEBUG__ */
17:
18: int sys_chroot(const char *dirname)
19: {
20:     struct inode *i;
21:     char *tmp_name;
22:     int errno;
23:
24: #ifdef __DEBUG__
25:     printk("(pid %d) sys_chroot('%s')\n", current->pid, dirname);
26: #endif /* __DEBUG__ */
27:
28:     if((errno = malloc_name(dirname, &tmp_name)) < 0) {
29:         return errno;
30:     }
31:     if((errno = namei(tmp_name, &i, NULL, FOLLOW_LINKS)) {
32:         free_name(tmp_name);
33:         return errno;
34:     }
35:     if(!S_ISDIR(i->i_mode)) {
36:         iput(i);
37:         free_name(tmp_name);
38:         return -ENOTDIR;
39:     }
40:     iput(current->root);
41:     current->root = i;
42:     free_name(tmp_name);
43:     return 0;
44: }
```

kernel/syscalls/close.c

Page 1/1

```

1:  /*
2:  *  fiwix/kernel/syscalls/close.c
3:  *
4:  *  Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  *  Distributed under the terms of the Fiwix License.
6:  */
7:
8:  #include <fiwix/syscalls.h>
9:  #include <fiwix/locks.h>
10: #include <fiwix/errno.h>
11: #include <fiwix/stdio.h>
12:
13: int sys_close(unsigned int ufd)
14: {
15:     unsigned int fd;
16:     struct inode *i;
17:
18: #ifdef __DEBUG__
19:     printk("(pid %d) sys_close(%d)\n", current->pid, ufd);
20: #endif /*__DEBUG__ */
21:
22:     CHECK_UFD(ufd);
23:     fd = current->fd[ufd];
24:     release_user_fd(ufd);
25:
26:     if(--fd_table[fd].count) {
27:         return 0;
28:     }
29:     i = fd_table[fd].inode;
30:     flock_release_inode(i);
31:     if(i->fsop && i->fsop->close) {
32:         i->fsop->close(i, &fd_table[fd]);
33:         release_fd(fd);
34:         iput(i);
35:         return 0;
36:     }
37:     printk("WARNING: %s(): ufd %d without the close() method!\n", __FUNCTION
__, ufd);
38:     return -EINVAL;
39: }

```

kernel/syscalls/creat.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/creat.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/syscalls.h>
10: #include <fiwix/fcntl.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #include <fiwix/process.h>
15: #endif /*__DEBUG__*/
16:
17: int sys_creat(const char *filename, __mode_t mode)
18: {
19: #ifdef __DEBUG__
20:     printk("(pid %d) sys_creat('%s', %d)\n", current->pid, filename, mode);
21: #endif /*__DEBUG__*/
22:     return sys_open(filename, O_CREAT | O_WRONLY | O_TRUNC, mode);
23: }
```

kernel/syscalls/dup2.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/dup2.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/syscalls.h>
9: #include <fiwix/process.h>
10: #include <fiwix/errno.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #endif /*__DEBUG__*/
15:
16: int sys_dup2(int old_ufd, int new_ufd)
17: {
18: #ifdef __DEBUG__
19:     printk("(pid %d) sys_dup2(%d, %d)", current->pid, old_ufd, new_ufd);
20: #endif /*__DEBUG__*/
21:
22:     CHECK_UFD(old_ufd);
23:     if(new_ufd < 0 || new_ufd > OPEN_MAX) {
24:         return -EINVAL;
25:     }
26:     if(old_ufd == new_ufd) {
27:         return new_ufd;
28:     }
29:     if(current->fd[new_ufd]) {
30:         sys_close(new_ufd);
31:     }
32:     if((new_ufd = get_new_user_fd(new_ufd)) < 0) {
33:         return new_ufd;
34:     }
35:     current->fd[new_ufd] = current->fd[old_ufd];
36:     fd_table[current->fd[new_ufd]].count++;
37: #ifdef __DEBUG__
38:     printk(" --> returning %d\n", new_ufd);
39: #endif /*__DEBUG__*/
40:     return new_ufd;
41: }
```


kernel/syscalls/dup.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/dup.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/syscalls.h>
10: #include <fiwix/errno.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #include <fiwix/process.h>
15: #endif /*__DEBUG__*/
16:
17: int sys_dup(unsigned int ufd)
18: {
19:     int new_ufd;
20:
21: #ifdef __DEBUG__
22:     printk("(pid %d) sys_dup(%d)", current->pid, ufd);
23: #endif /*__DEBUG__*/
24:
25:     CHECK_UFD(ufd);
26:     if((new_ufd = get_new_user_fd(0)) < 0) {
27:         return new_ufd;
28:     }
29:
30: #ifdef __DEBUG__
31:     printk(" -> %d\n", new_ufd);
32: #endif /*__DEBUG__*/
33:
34:     current->fd[new_ufd] = current->fd[ufd];
35:     fd_table[current->fd[new_ufd]].count++;
36:     return new_ufd;
37: }
```

kernel/syscalls/execve.c

Page 1/6

```

1:  /*
2:  * fiwix/kernel/syscalls/execve.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8:  #include <fiwix/syscalls.h>
9:  #include <fiwix/stat.h>
10: #include <fiwix/buffer.h>
11: #include <fiwix/mm.h>
12: #include <fiwix/process.h>
13: #include <fiwix/fcntl.h>
14: #include <fiwix/errno.h>
15: #include <fiwix/string.h>
16:
17: #ifdef __DEBUG__
18: #include <fiwix/stdio.h>
19: #endif /* __DEBUG__ */
20:
21: static int initialize_barg(struct binargs *barg, char *argv[], char *envp[])
22: {
23:     int n, errno;
24:
25:     for(n = 0; n < ARG_MAX; n++) {
26:         barg->page[n] = 0;
27:     }
28:     barg->argv_len = barg->envp_len = 0;
29:
30:     for(n = 0; argv[n]; n++) {
31:         if((errno = check_user_area(VERIFY_READ, argv[n], sizeof(char *)))
32: )) {
33:             return errno;
34:         }
35:         barg->argv_len += strlen(argv[n]) + 1;
36:     }
37:     barg->argc = n;
38:
39:     for(n = 0; envp[n]; n++) {
40:         if((errno = check_user_area(VERIFY_READ, envp[n], sizeof(char *)))
41: )) {
42:             return errno;
43:         }
44:         barg->envp_len += strlen(envp[n]) + 1;
45:     }
46:     barg->envc = n;
47:     return 0;
48: }
49: static void free_barg_pages(struct binargs *barg)
50: {
51:     int n;
52:
53:     for(n = 0; n < ARG_MAX; n++) {
54:         if(barg->page[n]) {
55:             kfree(barg->page[n]);
56:         }
57:     }
58: }
59:
60: static int add_strings(struct binargs *barg, char *filename, char *interpreter,
61: char *args)
62: {
63:     int n, p, offset;
64:     unsigned int ae_str_len;
65:     char *page;

```

kernel/syscalls/execve.c

Page 2/6

```

65:
66:     /*
67:      * For a script we need to substitute the saved argv[0] by the original
68:      * 'filename' supplied in execve(), otherwise the interpreter won't be
69:      * able to find the script file.
70:      */
71:     p = ARG_MAX - 1;
72:     ae_str_len = barg->argv_len + barg->envp_len + 4;
73:     p -= ae_str_len / PAGE_SIZE;
74:     offset = PAGE_SIZE - (ae_str_len % PAGE_SIZE);
75:     if(offset == PAGE_SIZE) {
76:         offset = 0;
77:         p++;
78:     }
79:     page = (char *)barg->page[p];
80:     while(*(page + offset)) {
81:         offset++;
82:         barg->argv_len--;
83:         if(offset == PAGE_SIZE) {
84:             p++;
85:             offset = 0;
86:             page = (char *)barg->page[p];
87:         }
88:     }
89:     barg->argv_len--;
90:
91:
92:     p = ARG_MAX - 1;
93:     barg->argv_len += strlen(interpreter) + 1;
94:     barg->argv_len += strlen(args) ? strlen(args) + 1 : 0;
95:     barg->argv_len += strlen(filename) + 1;
96:     barg->argc++;
97:     if(*args) {
98:         barg->argc++;
99:     }
100:    ae_str_len = barg->argv_len + barg->envp_len + 4;
101:    p -= ae_str_len / PAGE_SIZE;
102:    offset = PAGE_SIZE - (ae_str_len % PAGE_SIZE);
103:    if(offset == PAGE_SIZE) {
104:        offset = 0;
105:        p++;
106:    }
107:    barg->offset = offset;
108:    for(n = p; n < ARG_MAX; n++) {
109:        if(!barg->page[n]) {
110:            if(!(barg->page[n] = kmalloc())) {
111:                free_barg_pages(barg);
112:                return -ENOMEM;
113:            }
114:        }
115:    }
116:
117:    /* interpreter */
118:    page = (char *)barg->page[p];
119:    while(*interpreter) {
120:        *(page + offset) = *interpreter;
121:        offset++;
122:        interpreter++;
123:        if(offset == PAGE_SIZE) {
124:            p++;
125:            offset = 0;
126:            page = (char *)barg->page[p];
127:        }
128:    }
129:    *(page + offset++) = NULL;
130:    if(offset == PAGE_SIZE) {
131:        p++;

```

kernel/syscalls/execve.c

Page 3/6

```

132:         offset = 0;
133:     }
134:
135:     /* args */
136:     page = (char *)barg->page[p];
137:     if(*args) {
138:         while(*args) {
139:             *(page + offset) = *args;
140:             offset++;
141:             args++;
142:             if(offset == PAGE_SIZE) {
143:                 p++;
144:                 offset = 0;
145:                 page = (char *)barg->page[p];
146:             }
147:         }
148:         *(page + offset++) = NULL;
149:         if(offset == PAGE_SIZE) {
150:             p++;
151:             offset = 0;
152:         }
153:     }
154:
155:     /* original script ('filename' with path) at argv[0] */
156:     page = (char *)barg->page[p];
157:     while(*filename) {
158:         *(page + offset) = *filename;
159:         offset++;
160:         filename++;
161:         if(offset == PAGE_SIZE) {
162:             p++;
163:             offset = 0;
164:             page = (char *)barg->page[p];
165:         }
166:     }
167:     *(page + offset) = NULL;
168:
169:     return 0;
170: }
171:
172: static int copy_strings(struct binargs *barg, char *argv[], char *envp[])
173: {
174:     int n, p, offset;
175:     unsigned int ae_str_len;
176:     char *page, *str;
177:
178:     p = ARG_MAX - 1;
179:     ae_str_len = barg->argv_len + barg->envp_len + 4;
180:     p -= ae_str_len / PAGE_SIZE;
181:     offset = PAGE_SIZE - (ae_str_len % PAGE_SIZE);
182:     if(offset == PAGE_SIZE) {
183:         offset = 0;
184:         p++;
185:     }
186:     barg->offset = offset;
187:     for(n = p; n < ARG_MAX; n++) {
188:         if(!(barg->page[n] = kmalloc())) {
189:             free_barg_pages(barg);
190:             return -ENOMEM;
191:         }
192:     }
193:     for(n = 0; n < barg->argc; n++) {
194:         str = argv[n];
195:         page = (char *)barg->page[p];
196:         while(*str) {
197:             *(page + offset) = *str;
198:             offset++;

```

kernel/syscalls/execve.c

Page 4/6

```

199:         str++;
200:         if(offset == PAGE_SIZE) {
201:             p++;
202:             offset = 0;
203:             page = (char *)barg->page[p];
204:         }
205:     }
206:     *(page + offset++) = NULL;
207:     if(offset == PAGE_SIZE) {
208:         p++;
209:         offset = 0;
210:     }
211: }
212: for(n = 0; n < barg->envc; n++) {
213:     str = envp[n];
214:     page = (char *)barg->page[p];
215:     while(*str) {
216:         *(page + offset) = *str;
217:         offset++;
218:         str++;
219:         if(offset == PAGE_SIZE) {
220:             p++;
221:             offset = 0;
222:             page = (char *)barg->page[p];
223:         }
224:     }
225:     *(page + offset++) = NULL;
226:     if(offset == PAGE_SIZE) {
227:         p++;
228:         offset = 0;
229:     }
230: }
231:
232: return 0;
233: }
234:
235: static int do_execve(const char *filename, char *argv[], char *envp[], struct si
gcontext *sc)
236: {
237:     char interpreter[NAME_MAX + 1], args[NAME_MAX + 1], name[NAME_MAX + 1];
238:     __blk_t block;
239:     struct buffer *buf;
240:     struct inode *i;
241:     struct binargs barg;
242:     char *data, *tmp_name;
243:     int errno;
244:
245:     if((errno = initialize_barg(&barg, &(*argv), &(*envp))) < 0) {
246:         return errno;
247:     }
248:
249:     /* save 'argv' and 'envp' into the kernel address space */
250:     if((errno = copy_strings(&barg, &(*argv), &(*envp))) {
251:         return errno;
252:     }
253:
254:     if(!(data = (void *)kmalloc())) {
255:         return -ENOMEM;
256:     }
257:
258:     if((errno = malloc_name(filename, &tmp_name)) < 0) {
259:         kfree((unsigned int)data);
260:         free_barg_pages(&barg);
261:         return errno;
262:     }
263:     strcpy(name, tmp_name);
264:     free_name(tmp_name);

```

kernel/syscalls/execve.c

Page 5/6

```

265:
266:
267: loop:
268:     if((errno = namei(name, &i, NULL, FOLLOW_LINKS)) {
269:         free_barg_pages(&barg);
270:         kfree((unsigned int)data);
271:         return errno;
272:     }
273:
274:     if(!S_ISREG(i->i_mode)) {
275:         iput(i);
276:         free_barg_pages(&barg);
277:         kfree((unsigned int)data);
278:         return -EACCES;
279:     }
280:     if(check_permission(TO_EXEC, i) < 0) {
281:         iput(i);
282:         free_barg_pages(&barg);
283:         kfree((unsigned int)data);
284:         return -EACCES;
285:     }
286:
287:     if((block = bmap(i, 0, FOR_READING)) < 0) {
288:         iput(i);
289:         free_barg_pages(&barg);
290:         kfree((unsigned int)data);
291:         return block;
292:     }
293:     if(!(buf = bread(i->dev, block, i->sb->s_blocksize))) {
294:         iput(i);
295:         free_barg_pages(&barg);
296:         kfree((unsigned int)data);
297:         return -EIO;
298:     }
299:
300:     /*
301:     * The contents of the buffer is copied and then freed immediately to
302:     * make sure that it won't conflict while zeroing the BSS fractional
303:     * page, in case that the same block is requested during the page fault.
304:     */
305:     memcpy_b(data, buf->data, i->sb->s_blocksize);
306:     brelse(buf);
307:
308:     errno = elf_load(i, &barg, sc, data);
309:     if(errno == -ENOEXEC) {
310:         /* OK, looks like it was not an ELF binary; let's see if it is a
script */
311:         memset_b(interpreter, 0, NAME_MAX + 1);
312:         memset_b(args, 0, NAME_MAX + 1);
313:         errno = script_load(interpreter, args, data);
314:         if(!errno) {
315:             /* yes, it is! */
316:             iput(i);
317:             if((errno = add_strings(&barg, name, interpreter, args))
) {
318:                 free_barg_pages(&barg);
319:                 kfree((unsigned int)data);
320:                 return errno;
321:             }
322:             strcpy(name, interpreter);
323:             goto loop;
324:         }
325:     }
326:
327:     if(!errno) {
328:         if(i->i_mode & S_ISUID) {
329:             current->euid = i->i_uid;

```

kernel/syscalls/execve.c

Page 6/6

```

330:         }
331:         if(i->i_mode & S_ISGID) {
332:             current->egid = i->i_gid;
333:         }
334:     }
335:
336:     iput(i);
337:     free_barg_pages(&barg);
338:     kfree((unsigned int)data);
339:     return errno;
340: }
341:
342: int sys_execve(const char *filename, char *argv[], char *envp[], int arg4, int a
rg5, struct sigcontext *sc)
343: {
344:     char argv0[NAME_MAX + 1];
345:     int n, errno;
346:
347: #ifdef __DEBUG__
348:     printk("(pid %d) sys_execve('%s', ...) \n", current->pid, filename);
349: #endif /*__DEBUG__*/
350:
351:     /* copy filename into kernel address space */
352:     strncpy(argv0, argv[0], NAME_MAX);
353:     if((errno = do_execve(filename, &(*argv), &(*envp), sc)) {
354:         return errno;
355:     }
356:
357:     strncpy(current->argv0, argv0, NAME_MAX);
358:     for(n = 0; n < OPEN_MAX; n++) {
359:         if(current->fd[n] && (current->fd_flags[n] & FD_CLOEXEC)) {
360:             sys_close(n);
361:         }
362:     }
363:
364:     current->suid = current->euid;
365:     current->sgid = current->egid;
366:     current->sigpending = 0;
367:     current->sigexecuting = 0;
368:     for(n = 0; n < NSIG; n++) {
369:         current->sigaction[n].sa_mask = 0;
370:         current->sigaction[n].sa_flags = 0;
371:         if(current->sigaction[n].sa_handler != SIG_IGN) {
372:             current->sigaction[n].sa_handler = SIG_DFL;
373:         }
374:     }
375:     current->sleep_address = NULL;
376:     current->flags |= PF_PEXEC;
377:     return 0;
378: }

```

kernel/syscalls/exit.c

Page 1/2

```

1: /*
2:  * fiwix/kernel/syscalls/exit.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/kernel.h>
10: #include <fiwix/syscalls.h>
11: #include <fiwix/process.h>
12: #include <fiwix/sched.h>
13: #include <fiwix/mman.h>
14: #include <fiwix/sleep.h>
15: #include <fiwix/stdio.h>
16: #include <fiwix/string.h>
17:
18: void do_exit(int exit_code)
19: {
20:     int n;
21:     struct proc *p, *init;
22:
23: #ifdef __DEBUG__
24:     printk("\n");
25:     printk("sys_exit(pid %d, ppid %d)\n", current->pid, current->ppid);
26:     printk("-----\n");
27: #endif /*__DEBUG__ */
28:
29:     release_binary();
30:     current->argv = NULL;
31:     current->envp = NULL;
32:
33:     init = get_proc_by_pid(INIT);
34:     FOR_EACH_PROCESS(p) {
35:         if(SESS_LEADER(current)) {
36:             if(p->sid == current->sid && p->state != PROC_ZOMBIE) {
37:                 p->pgid = 0;
38:                 p->sid = 0;
39:                 p->ctty = NULL;
40:                 send_sig(p, SIGHUP);
41:                 send_sig(p, SIGCONT);
42:             }
43:         }
44:
45:         /* make INIT inherit the children of this exiting process */
46:         if(p->ppid == current->pid) {
47:             p->ppid = INIT;
48:             init->children++;
49:             current->children--;
50:             if(p->state == PROC_ZOMBIE) {
51:                 send_sig(init, SIGCHLD);
52:                 if(init->sleep_address == &sys_wait4) {
53:                     wakeup_proc(init);
54:                 }
55:             }
56:         }
57:         p = p->next;
58:     }
59:
60:     if(SESS_LEADER(current)) {
61:         disassociate_ctty(current->ctty);
62:     }
63:
64:     for(n = 0; n < OPEN_MAX; n++) {
65:         if(current->fd[n]) {
66:             sys_close(n);
67:         }

```


kernel/syscalls/exit.c

Page 2/2

```

68:         }
69:
70:         iput(current->root);
71:         current->root = NULL;
72:         iput(current->pwd);
73:         current->pwd = NULL;
74:         current->exit_code = exit_code;
75:         if(!--nr_processes) {
76:             printk("\n");
77:             printk("WARNING: the last user process has exited. The kernel wi
11 stop itself.\n");
78:             stop_kernel();
79:         }
80:
81:         /* notify the parent about the child's death */
82:         if((p = get_proc_by_pid(current->ppid)) {
83:             send_sig(p, SIGCHLD);
84:             if(p->sleep_address == &sys_wait4) {
85:                 wakeup_proc(p);
86:             }
87:         }
88:
89:         current->sigpending = 0;
90:         current->sigblocked = 0;
91:         current->sigexecuting = 0;
92:         for(n = 0; n < NSIG; n++) {
93:             current->sigaction[n].sa_mask = 0;
94:             current->sigaction[n].sa_flags = 0;
95:             current->sigaction[n].sa_handler = SIG_IGN;
96:         }
97:
98:         not_runnable(current, PROC_ZOMBIE);
99:         need_resched = 1;
100:        do_sched();
101:    }
102:
103:    int sys_exit(int exit_code)
104:    {
105:    #ifdef __DEBUG__
106:        printk("(pid %d) sys_exit()\n", current->pid);
107:    #endif /*__DEBUG__*/
108:
109:        /* exit code in the second byte.
110:         * 15          8 7          0
111:         * +-----+-----+-----+
112:         * | exit code (0-255) |          0          |
113:         * +-----+-----+-----+
114:         */
115:        do_exit((exit_code & 0xFF) << 8);
116:        return 0;
117:    }

```

kernel/syscalls/fchdir.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/fchdir.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/process.h>
10: #include <fiwix/stat.h>
11: #include <fiwix/errno.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #endif /* __DEBUG__ */
16:
17: int sys_fchdir(unsigned int ufd)
18: {
19:     struct inode *i;
20:
21: #ifdef __DEBUG__
22:     printk("(pid %d) sys_fchdir(%d)\n", current->pid, ufd);
23: #endif /* __DEBUG__ */
24:
25:     CHECK_UFD(ufd);
26:     i = fd_table[current->fd[ufd]].inode;
27:     if(!S_ISDIR(i->i_mode)) {
28:         return -ENOTDIR;
29:     }
30:     iput(current->pwd);
31:     current->pwd = i;
32:     current->pwd->count++;
33:     return 0;
34: }
```

kernel/syscalls/fchmod.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/fchmod.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/types.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/stat.h>
12: #include <fiwix/errno.h>
13:
14: #ifdef __DEBUG__
15: #include <fiwix/stdio.h>
16: #include <fiwix/process.h>
17: #endif /*__DEBUG__ */
18:
19: int sys_fchmod(int ufd, __mode_t mode)
20: {
21:     struct inode *i;
22:
23: #ifdef __DEBUG__
24:     printk("(pid %d) sys_fchmod(%d, %d)\n", current->pid, ufd, mode);
25: #endif /*__DEBUG__ */
26:
27:     CHECK_UFD(ufd);
28:     i = fd_table[current->fd[ufd]].inode;
29:
30:     if(IS_RDONLY_FS(i)) {
31:         return -EROFS;
32:     }
33:     if(check_user_permission(i)) {
34:         return -EPERM;
35:     }
36:
37:     i->i_mode &= S_IFMT;
38:     i->i_mode |= mode & ~S_IFMT;
39:     i->i_ctime = CURRENT_TIME;
40:     i->dirty = 1;
41:     return 0;
42: }
```

kernel/syscalls/fchown.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/fchown.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/types.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/stat.h>
12: #include <fiwix/process.h>
13: #include <fiwix/errno.h>
14:
15: #ifdef __DEBUG__
16: #include <fiwix/stdio.h>
17: #endif /*__DEBUG__*/
18:
19: int sys_fchown(int ufd, __uid_t owner, __gid_t group)
20: {
21:     struct inode *i;
22:
23: #ifdef __DEBUG__
24:     printk("(pid %d) sys_fchown(%d, %d, %d)\n", current->pid, ufd, owner, gr
oup);
25: #endif /*__DEBUG__*/
26:
27:     CHECK_UFD(ufd);
28:     i = fd_table[current->fd[ufd]].inode;
29:
30:     if(IS_RDONLY_FS(i)) {
31:         return -EROFS;
32:     }
33:     if(check_user_permission(i)) {
34:         return -EPERM;
35:     }
36:
37:     if(owner == (__uid_t)-1) {
38:         owner = i->i_uid;
39:     } else {
40:         i->i_mode &= ~(S_ISUID);
41:     }
42:     if(group == (__gid_t)-1) {
43:         group = i->i_gid;
44:     } else {
45:         i->i_mode &= ~(S_ISGID);
46:     }
47:
48:     i->i_uid = owner;
49:     i->i_gid = group;
50:     i->i_ctime = CURRENT_TIME;
51:     i->dirty = 1;
52:     return 0;
53: }
```

kernel/syscalls/fcntl.c

Page 1/2

```

1: /*
2:  * fiwix/kernel/syscalls/fcntl.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/syscalls.h>
9: #include <fiwix/fcntl.h>
10: #include <fiwix/locks.h>
11: #include <fiwix/errno.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #include <fiwix/process.h>
16: #endif /* __DEBUG__ */
17:
18: int sys_fcntl(int ufd, int cmd, unsigned long int arg)
19: {
20:     int new_ufd, errno;
21:
22: #ifdef __DEBUG__
23:     printk("(pid %d) sys_fcntl(%d, %d, 0x%08x)\n", current->pid, ufd, cmd, a
rg);
24: #endif /* __DEBUG__ */
25:
26:     CHECK_UFD(ufd);
27:     switch(cmd) {
28:         case F_DUPFD:
29:             CHECK_UFD(ufd);
30:             if(arg >= OPEN_MAX) {
31:                 return -EINVAL;
32:             }
33:             if((new_ufd = get_new_user_fd(arg)) < 0) {
34:                 return new_ufd;
35:             }
36:             current->fd[new_ufd] = current->fd[ufd];
37:             fd_table[current->fd[new_ufd]].count++;
38: #ifdef __DEBUG__
39:             printk("\t--> returning %d\n", new_ufd);
40: #endif /* __DEBUG__ */
41:             return new_ufd;
42:         case F_GETFD:
43:             return (current->fd_flags[ufd] & FD_CLOEXEC);
44:         case F_SETFD:
45:             current->fd_flags[ufd] = (arg & FD_CLOEXEC);
46:             break;
47:         case F_GETFL:
48:             return fd_table[current->fd[ufd]].flags;
49:         case F_SETFL:
50:             fd_table[current->fd[ufd]].flags &= ~(O_APPEND | O_NONBL
OCK);
51:             fd_table[current->fd[ufd]].flags |= arg & (O_APPEND | O_
NONBLOCK);
52:             break;
53:         case F_GETLK:
54:         case F_SETLK:
55:         case F_SETLKW:
56:             if((errno = check_user_area(VERIFY_READ, (void *)arg, si
zeof(struct flock)))) {
57:                 return errno;
58:             }
59:             return posix_lock(ufd, cmd, (struct flock *)arg);
60:         default:
61:             return -EINVAL;
62:     }
63:     return 0;

```

```
64: }
```

kernel/syscalls/fdatasync.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/fdatasync.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/syscalls.h>
9:
10: #ifdef __DEBUG__
11: #include <fiwix/stdio.h>
12: #include <fiwix/process.h>
13: #endif /*__DEBUG__ */
14:
15: int sys_fdatasync(int ufd)
16: {
17: #ifdef __DEBUG__
18:     printk("(pid %d) sys_fdatasync(%d)\n", current->pid, ufd);
19: #endif /*__DEBUG__ */
20:
21:     return sys_fsync(ufd);
22: }
```

kernel/syscalls/flock.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/flock.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/process.h>
10: #include <fiwix/locks.h>
11: #include <fiwix/errno.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #endif /*__DEBUG__*/
16:
17: int sys_flock(int ufd, int op)
18: {
19:     struct inode *i;
20:
21: #ifdef __DEBUG__
22:     printk("(pid %d) sys_flock(%d, %d)\n", current->pid, ufd, op);
23: #endif /*__DEBUG__*/
24:
25:     CHECK_UFD(ufd);
26:     i = fd_table[current->fd[ufd]].inode;
27:     return flock_inode(i, op);
28: }
```


kernel/syscalls/fork.c

Page 1/3

```

1: /*
2:  * fiwix/kernel/syscalls/fork.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/kernel.h>
10: #include <fiwix/types.h>
11: #include <fiwix/segments.h>
12: #include <fiwix/sigcontext.h>
13: #include <fiwix/process.h>
14: #include <fiwix/sched.h>
15: #include <fiwix/sleep.h>
16: #include <fiwix/mm.h>
17: #include <fiwix/errno.h>
18: #include <fiwix/stdio.h>
19: #include <fiwix/string.h>
20:
21: int sys_fork(int arg1, int arg2, int arg3, int arg4, int arg5, struct sigcontext
*sc)
22: {
23:     int count, pages;
24:     unsigned int n;
25:     unsigned int *child_pgid;
26:     struct sigcontext *stack;
27:     struct proc *child, *p;
28:     struct vma *vma;
29:     __pid_t pid;
30:
31: #ifdef __DEBUG__
32:     printk("(pid %d) sys_fork()\n", current->pid);
33: #endif /*__DEBUG__ */
34:
35:     /* check the number of processes already allocated by this UID */
36:     count = 0;
37:     FOR_EACH_PROCESS(p) {
38:         if(p->uid == current->uid) {
39:             count++;
40:         }
41:         p = p->next;
42:     }
43:     if(count > current->rlim[RLIMIT_NPROC].rlim_cur) {
44:         printk("WARNING: %s(): RLIMIT_NPROC exceeded.\n", __FUNCTION__);
45:         return -EAGAIN;
46:     }
47:
48:     if(!(pid = get_unused_pid())) {
49:         return -EAGAIN;
50:     }
51:     if(!(child = get_proc_free())) {
52:         return -EAGAIN;
53:     }
54:
55:     /*
56:      * This memcpy() will overwrite the prev and next pointers, so that's
57:      * the reason why proc_slot_init() is separated from get_proc_free().
58:      */
59:     memcpy_b(child, current, sizeof(struct proc));
60:
61:     proc_slot_init(child);
62:     child->pid = pid;
63:     memset_b(&child->tss, NULL, sizeof(struct i386tss));
64:     sprintf(child->pidstr, "%d", child->pid);
65:
66:     if(!(child_pgid = (void *)kmalloc()) {

```

kernel/syscalls/fork.c

Page 2/3

```

67:         release_proc(child);
68:         return -ENOMEM;
69:     }
70:     child->rss++;
71:     memcpy_b(child_pgdirt, kpage_dir, PAGE_SIZE);
72:     child->tss.cr3 = V2P((unsigned int)child_pgdirt);
73:
74:     child->ppid = current->pid;
75:     child->flags = 0;
76:     child->children = 0;
77:     child->cpu_count = child->priority;
78:     child->start_time = CURRENT_TICKS;
79:     child->sleep_address = NULL;
80:
81:     memcpy_b(child->vma, current->vma, sizeof(child->vma));
82:     vma = child->vma;
83:     for(n = 0; n < VMA_REGIONS && vma->start; n++, vma++) {
84:         if(vma->inode) {
85:             vma->inode->count++;
86:         }
87:     }
88:
89:     child->sigpending = 0;
90:     child->sigexecuting = 0;
91:     memset_b(&child->sc, NULL, sizeof(struct sigcontext));
92:     memset_b(&child->usage, NULL, sizeof(struct rusage));
93:     memset_b(&child->cusage, NULL, sizeof(struct rusage));
94:     child->it_real_interval = 0;
95:     child->it_real_value = 0;
96:     child->it_virt_interval = 0;
97:     child->it_virt_value = 0;
98:     child->it_prof_interval = 0;
99:     child->it_prof_value = 0;
100:
101:     if(!(child->tss.esp0 = kmalloc())) {
102:         kfree((unsigned int)child_pgdirt);
103:         kfree((unsigned int)child->vma);
104:         release_proc(child);
105:         return -ENOMEM;
106:     }
107:
108:     if(!(pages = clone_pages(child))) {
109:         printk("WARNING: %s(): not enough memory, can't clone pages.\n",
__FUNCTION__);
110:         free_page_tables(child);
111:         kfree((unsigned int)child_pgdirt);
112:         kfree((unsigned int)child->vma);
113:         release_proc(child);
114:         return -ENOMEM;
115:     }
116:     child->rss += pages;
117:     invalidate_tlb();
118:
119:     child->tss.esp0 += PAGE_SIZE - 4;
120:     child->rss++;
121:     child->tss.ss0 = KERNEL_DS;
122:
123:     memcpy_b((unsigned int *) (child->tss.esp0 & PAGE_MASK), (void *) ((unsigned
ed int) (sc) & PAGE_MASK), PAGE_SIZE);
124:     stack = (struct sigcontext *) ((child->tss.esp0 & PAGE_MASK) + ((unsigned
int) (sc) & ~PAGE_MASK));
125:
126:     child->tss.eip = (unsigned int) return_from_syscall;
127:     child->tss.esp = (unsigned int) stack;
128:     stack->eax = 0;          /* child returns 0 */
129:
130:     /* increase file descriptors usage */

```

kernel/syscalls/fork.c

Page 3/3

```
131:         for (n = 0; n < OPEN_MAX; n++) {
132:             if (current->fd[n]) {
133:                 fd_table[current->fd[n]].count++;
134:             }
135:         }
136:         if (current->root) {
137:             current->root->count++;
138:         }
139:         if (current->pwd) {
140:             current->pwd->count++;
141:         }
142:
143:         kstat.processes++;
144:         nr_processes++;
145:         current->children++;
146:         runnable(child);
147:
148:         return child->pid;         /* parent returns child's PID */
149: }
```

kernel/syscalls/fstat.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/fstat.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/syscalls.h>
10: #include <fiwix/statbuf.h>
11: #include <fiwix/errno.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #include <fiwix/process.h>
16: #endif /* __DEBUG__ */
17:
18: int sys_fstat(unsigned int ufd, struct old_stat *statbuf)
19: {
20:     struct inode *i;
21:     int errno;
22:
23: #ifdef __DEBUG__
24:     printk("(pid %d) sys_fstat(%d, 0x%08x) -> returning structure\n", current->pid, ufd, (unsigned int) statbuf);
25: #endif /* __DEBUG__ */
26:
27:     CHECK_UFD(ufd);
28:     if((errno = check_user_area(VERIFY_WRITE, statbuf, sizeof(struct old_stat)))) {
29:         return errno;
30:     }
31:     i = fd_table[current->fd[ufd]].inode;
32:     statbuf->st_dev = i->dev;
33:     statbuf->st_ino = i->ino;
34:     statbuf->st_mode = i->i_mode;
35:     statbuf->st_nlink = i->i_nlink;
36:     statbuf->st_uid = i->i_uid;
37:     statbuf->st_gid = i->i_gid;
38:     statbuf->st_rdev = i->rdev;
39:     statbuf->st_size = i->i_size;
40:     statbuf->st_atime = i->i_atime;
41:     statbuf->st_mtime = i->i_mtime;
42:     statbuf->st_ctime = i->i_ctime;
43:     return 0;
44: }

```

kernel/syscalls/fstatfs.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/fstatfs.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/statfs.h>
10: #include <fiwix/errno.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #include <fiwix/process.h>
15: #endif /*__DEBUG__*/
16:
17: int sys_fstatfs(unsigned int ufd, struct statfs *statfsbuf)
18: {
19:     struct inode *i;
20:     int errno;
21:
22: #ifdef __DEBUG__
23:     printk("(pid %d) sys_fstatfs(%d, 0x%08x)\n", current->pid, ufd, (unsigned
d int)statfsbuf);
24: #endif /*__DEBUG__*/
25:
26:     CHECK_UFD(ufd);
27:     if((errno = check_user_area(VERIFY_WRITE, statfsbuf, sizeof(struct statf
s)))) {
28:         return errno;
29:     }
30:     i = fd_table[current->fd[ufd]].inode;
31:     if(i->sb && i->sb->fsop && i->sb->fsop->statfs) {
32:         i->sb->fsop->statfs(i->sb, statfsbuf);
33:         return 0;
34:     }
35:     return -ENOSYS;
36: }
```

kernel/syscalls/fsync.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/fsync.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/filesystems.h>
10: #include <fiwix/process.h>
11: #include <fiwix/stat.h>
12: #include <fiwix/buffer.h>
13: #include <fiwix/errno.h>
14:
15: #ifdef __DEBUG__
16: #include <fiwix/stdio.h>
17: #endif /*__DEBUG__*/
18:
19: int sys_fsync(int ufd)
20: {
21:     struct inode *i;
22:
23: #ifdef __DEBUG__
24:     printk(" (pid %d) sys_fsync(%d)\n", current->pid, ufd);
25: #endif /*__DEBUG__*/
26:
27:     CHECK_UFD(ufd);
28:     i = fd_table[current->fd[ufd]].inode;
29:     if(!S_ISREG(i->i_mode)) {
30:         return -EINVAL;
31:     }
32:     if(IS_RDONLY_FS(i)) {
33:         return -EROFS;
34:     }
35:     sync_superblocks(i->dev);
36:     sync_inodes(i->dev);
37:     sync_buffers(i->dev);
38:     return 0;
39: }
```

kernel/syscalls/ftime.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/ftime.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/timeb.h>
11: #include <fiwix/timer.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #include <fiwix/process.h>
16: #endif /* __DEBUG__ */
17:
18: int sys_ftime(struct timeb *tp)
19: {
20:     int errno;
21:
22: #ifdef __DEBUG__
23:     printk("(pid %d) sys_ftime()\n", current->pid);
24: #endif /* __DEBUG__ */
25:
26:     if((errno = check_user_area(VERIFY_WRITE, tp, sizeof(struct timeb)))) {
27:         return errno;
28:     }
29:     tp->time = CURRENT_TIME;
30:     tp->millitm = ((kstat.ticks % HZ) * 1000000) / HZ;
31:     /* FIXME: 'timezone' and 'dstflag' fields are not used */
32:
33:     return 0;
34: }
```

kernel/syscalls/ftruncate.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/ftruncate.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/fcntl.h>
11: #include <fiwix/stat.h>
12: #include <fiwix/errno.h>
13:
14: #ifdef __DEBUG__
15: #include <fiwix/stdio.h>
16: #include <fiwix/process.h>
17: #endif /*__DEBUG__*/
18:
19: int sys_ftruncate(int ufd, __off_t length)
20: {
21:     struct inode *i;
22:     int errno;
23:
24: #ifdef __DEBUG__
25:     printk("(pid %d) sys_ftruncate(%d, %d)\n", current->pid, ufd, length);
26: #endif /*__DEBUG__*/
27:
28:     CHECK_UFD(ufd);
29:     i = fd_table[current->fd[ufd]].inode;
30:     if((fd_table[current->fd[ufd]].flags & O_ACCMODE) == O_RDONLY) {
31:         return -EINVAL;
32:     }
33:     if(S_ISDIR(i->i_mode)) {
34:         return -EISDIR;
35:     }
36:     if(IS_RDONLY_FS(i)) {
37:         return -EROFS;
38:     }
39:     if(check_permission(TO_WRITE, i) < 0) {
40:         return -EPERM;
41:     }
42:     if(length == i->i_size) {
43:         return 0;
44:     }
45:
46:     errno = 0;
47:     if(i->fsop && i->fsop->truncate) {
48:         inode_lock(i);
49:         errno = i->fsop->truncate(i, length);
50:         inode_unlock(i);
51:     }
52:     return errno;
53: }

```


kernel/syscalls/getcwd.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/getcwd.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/errno.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #include <fiwix/process.h>
15: #endif /*__DEBUG__*/
16:
17: int sys_getcwd(char *buf, __size_t size)
18: {
19:     int errno;
20:
21: #ifdef __DEBUG__
22:     printk("(pid %d) sys_getcwd(0x%08x, %d)\n", current->pid, (unsigned int)
buf, size);
23: #endif /*__DEBUG__*/
24:
25:     if((errno = check_user_area(VERIFY_WRITE, buf, size)) {
26:         return errno;
27:     }
28:     return -ENOSYS;
29: }
```

kernel/syscalls/getdents.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/getdents.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/dirent.h>
10: #include <fiwix/process.h>
11: #include <fiwix/stat.h>
12: #include <fiwix/errno.h>
13:
14: #ifdef __DEBUG__
15: #include <fiwix/stdio.h>
16: #endif /* __DEBUG__ */
17:
18: int sys_getdents(unsigned int ufd, struct dirent *dirent, unsigned int count)
19: {
20:     struct inode *i;
21:     int errno;
22:
23: #ifdef __DEBUG__
24:     printk("(pid %d) sys_getdents(%d, 0x%08x, %d)", current->pid, ufd, (unsi
gned int)dirent, count);
25: #endif /* __DEBUG__ */
26:
27:     CHECK_UFD(ufd);
28:     if((errno = check_user_area(VERIFY_WRITE, dirent, sizeof(struct dirent))
)) {
29:         return errno;
30:     }
31:     i = fd_table[current->fd[ufd]].inode;
32:
33:     if(!S_ISDIR(i->i_mode)) {
34:         return -ENOTDIR;
35:     }
36:
37:     if(i->fsop && i->fsop->readdir) {
38:         errno = i->fsop->readdir(i, &fd_table[current->fd[ufd]], dirent,
count);
39:     }
40:     #ifdef __DEBUG__
41:     printk(" -> returning %d\n", errno);
42:     #endif /* __DEBUG__ */
43:     return errno;
44: }
45: }

```

kernel/syscalls/getegid.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/getegid.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/process.h>
9:
10: #ifdef __DEBUG__
11: #include <fiwix/stdio.h>
12: #endif /*__DEBUG__*/
13:
14: int sys_getegid(void)
15: {
16: #ifdef __DEBUG__
17:     printk("(pid %d) sys_getegid() -> %d\n", current->pid, current->egid);
18: #endif /*__DEBUG__*/
19:     return current->egid;
20: }
```

kernel/syscalls/geteuid.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/geteuid.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/process.h>
9:
10: #ifdef __DEBUG__
11: #include <fiwix/stdio.h>
12: #endif /*__DEBUG__*/
13:
14: int sys_geteuid(void)
15: {
16: #ifdef __DEBUG__
17:     printk("(pid %d) sys_geteuid() -> %d\n", current->pid, current->euid);
18: #endif /*__DEBUG__*/
19:     return current->euid;
20: }
```

kernel/syscalls/getgid.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/getgid.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/process.h>
9:
10: #ifdef __DEBUG__
11: #include <fiwix/stdio.h>
12: #endif /*__DEBUG__*/
13:
14: int sys_getgid(void)
15: {
16: #ifdef __DEBUG__
17:     printk("(pid %d) sys_getgid() -> %d\n", current->pid, current->gid);
18: #endif /*__DEBUG__*/
19:     return current->gid;
20: }
```

kernel/syscalls/getgroups.c

Page 1/1

```

1:  /*
2:  * fiwix/kernel/syscalls/getgroups.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8:  #include <fiwix/types.h>
9:  #include <fiwix/fs.h>
10: #include <fiwix/process.h>
11: #include <fiwix/errno.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #endif /*__DEBUG__ */
16:
17: int sys_getgroups(__ssize_t size, __gid_t *list)
18: {
19:     int n, errno;
20:
21: #ifdef __DEBUG__
22:     printk("(pid %d) sys_getgroups(%d, 0x%08x)\n", current->pid, size, (unsi
gned int)list);
23: #endif /*__DEBUG__ */
24:
25:     /*
26:     * If size is 0, sys_getgroups() shall return the number of group IDs
27:     * that it would otherwise return without modifying the array pointed
28:     * to by list.
29:     */
30:     if(!size) {
31:         for(n = 0; n < NGROUPS_MAX; n++) {
32:             if(current->groups[n] == -1) {
33:                 break;
34:             }
35:         }
36:         return n;
37:     }
38:
39:     if((errno = check_user_area(VERIFY_WRITE, list, sizeof(__gid_t)))) {
40:         return errno;
41:     }
42:     for(n = 0; n < NGROUPS_MAX; n++) {
43:         if(current->groups[n] == -1) {
44:             break;
45:         }
46:         if(size) {
47:             if(n > size) {
48:                 return -EINVAL;
49:             }
50:             list[n] = (__gid_t)current->groups[n];
51:         }
52:     }
53:     return n;
54: }

```

kernel/syscalls/getitimer.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/getitimer.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/time.h>
10: #include <fiwix/process.h>
11: #include <fiwix/errno.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #endif /*__DEBUG__*/
16:
17: int sys_getitimer(int which, struct itimerval *curr_value)
18: {
19:     int errno;
20:
21: #ifdef __DEBUG__
22:     printk("(pid %d) sys_getitimer(%d, 0x%08x) -> \n", current->pid, which,
(unsigned int)curr_value);
23: #endif /*__DEBUG__*/
24:
25:     if((unsigned int)curr_value) {
26:         if((errno = check_user_area(VERIFY_WRITE, curr_value, sizeof(str
uct itimerval)))) {
27:             return errno;
28:         }
29:     }
30:
31:     switch(which) {
32:         case ITIMER_REAL:
33:             ticks2tv(current->it_real_interval, &curr_value->it_inte
rval);
34:             ticks2tv(current->it_real_value, &curr_value->it_value);
35:             break;
36:         case ITIMER_VIRTUAL:
37:             ticks2tv(current->it_virt_interval, &curr_value->it_inte
rval);
38:             ticks2tv(current->it_virt_value, &curr_value->it_value);
39:             break;
40:         case ITIMER_PROF:
41:             ticks2tv(current->it_prof_interval, &curr_value->it_inte
rval);
42:             ticks2tv(current->it_prof_value, &curr_value->it_value);
43:             break;
44:         default:
45:             return -EINVAL;
46:     }
47:     return 0;
48: }

```

kernel/syscalls/getpgid.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/getpgid.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/process.h>
10: #include <fiwix/sched.h>
11: #include <fiwix/errno.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #endif /*__DEBUG__*/
16:
17: int sys_getpgid(__pid_t pid)
18: {
19:     struct proc *p;
20:
21: #ifdef __DEBUG__
22:     printk("(pid %d) sys_getpgid(%d)\n", current->pid, pid);
23: #endif /*__DEBUG__*/
24:
25:     if(pid < 0) {
26:         return -EINVAL;
27:     }
28:     if(!pid) {
29:         return current->pgid;
30:     }
31:     FOR_EACH_PROCESS(p) {
32:         if(p->pid == pid) {
33:             return p->pgid;
34:         }
35:         p = p->next;
36:     }
37:     return -ESRCH;
38: }
```


kernel/syscalls/getpgrp.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/getpgrp.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/process.h>
9:
10: #ifdef __DEBUG__
11: #include <fiwix/stdio.h>
12: #endif /*__DEBUG__*/
13:
14: int sys_getpgrp(void)
15: {
16: #ifdef __DEBUG__
17:     printk("(pid %d) sys_getpgrp() -> %d\n", current->pid, current->pgid);
18: #endif /*__DEBUG__*/
19:     return current->pgid;
20: }
```

kernel/syscalls/getpid.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/getpid.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/process.h>
9:
10: #ifdef __DEBUG__
11: #include <fiwix/stdio.h>
12: #endif /*__DEBUG__*/
13:
14: int sys_getpid(void)
15: {
16: #ifdef __DEBUG__
17:     printk("(pid %d) sys_getpid() -> %d\n", current->pid, current->pid);
18: #endif /*__DEBUG__*/
19:     return current->pid;
20: }
```

kernel/syscalls/getppid.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/getppid.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/process.h>
9:
10: #ifdef __DEBUG__
11: #include <fiwix/stdio.h>
12: #endif /*__DEBUG__*/
13:
14: int sys_getppid(void)
15: {
16: #ifdef __DEBUG__
17:     printk("(pid %d) sys_getppid() -> %d\n", current->pid, current->ppid);
18: #endif /*__DEBUG__*/
19:     return current->ppid;
20: }
```

kernel/syscalls/getrlimit.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/getrlimit.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/resource.h>
10: #include <fiwix/process.h>
11: #include <fiwix/errno.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #endif /*__DEBUG__*/
16:
17: int sys_getrlimit(int resource, struct rlimit *rlim)
18: {
19:     int errno;
20:
21: #ifdef __DEBUG__
22:     printk("(pid %d) sys_getrlimit(%d, 0x%08x)\n", current->pid, resource, (
unsigned int)rlim);
23: #endif /*__DEBUG__*/
24:
25:     if((errno = check_user_area(VERIFY_WRITE, rlim, sizeof(struct rlimit))))
{
26:         return errno;
27:     }
28:     if(resource < 0 || resource >= RLIM_NLIMITS) {
29:         return -EINVAL;
30:     }
31:
32:     rlim->rlim_cur = current->rlim[resource].rlim_cur;
33:     rlim->rlim_max = current->rlim[resource].rlim_max;
34:     return 0;
35: }
```

kernel/syscalls/getrusage.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/getrusage.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/resource.h>
10: #include <fiwix/process.h>
11: #include <fiwix/errno.h>
12: #include <fiwix/string.h>
13:
14: #ifdef __DEBUG__
15: #include <fiwix/stdio.h>
16: #endif /* __DEBUG__ */
17:
18: int sys_getrusage(int who, struct rusage *usage)
19: {
20:     int errno;
21:
22: #ifdef __DEBUG__
23:     printk("(pid %d) sys_getrusage(%d, 0x%08x)\n", current->pid, who, (unsigned int)usage);
24: #endif /* __DEBUG__ */
25:
26:     if((errno = check_user_area(VERIFY_WRITE, usage, sizeof(struct rusage)))
) {
27:         return errno;
28:     }
29:     switch(who) {
30:         case RUSAGE_SELF:
31:             memcpy_b(usage, &current->usage, sizeof(struct rusage));
32:             break;
33:         case RUSAGE_CHILDREN:
34:             memcpy_b(usage, &current->cusage, sizeof(struct rusage))
;
35:             break;
36:         default:
37:             return -EINVAL;
38:     }
39:     return 0;
40: }
```

kernel/syscalls/getsid.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/getsid.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/process.h>
10: #include <fiwix/sched.h>
11: #include <fiwix/errno.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #endif /*__DEBUG__*/
16:
17: int sys_getsid(__pid_t pid)
18: {
19:     struct proc *p;
20:
21: #ifdef __DEBUG__
22:     printk("(pid %d) sys_getsid(%d)\n", current->pid, pid);
23: #endif /*__DEBUG__*/
24:
25:     if(pid < 0) {
26:         return -EINVAL;
27:     }
28:     if(!pid) {
29:         return current->sid;
30:     }
31:
32:     FOR_EACH_PROCESS(p) {
33:         if(p->pid == pid) {
34:             return p->sid;
35:         }
36:         p = p->next;
37:     }
38:     return -ESRCH;
39: }
```

kernel/syscalls/gettimeofday.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/gettimeofday.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/process.h>
11: #include <fiwix/time.h>
12: #include <fiwix/timer.h>
13:
14: #ifdef __DEBUG__
15: #include <fiwix/stdio.h>
16: #endif /* __DEBUG__ */
17:
18: int sys_gettimeofday(struct timeval *tv, struct timezone *tz)
19: {
20:     int errno;
21:
22: #ifdef __DEBUG__
23:     printk("(pid %d) sys_gettimeofday()\n", current->pid);
24: #endif /* __DEBUG__ */
25:
26:     if(tv) {
27:         if((errno = check_user_area(VERIFY_WRITE, tv, sizeof(struct time
val)))) {
28:             return errno;
29:         }
30:         tv->tv_sec = CURRENT_TIME;
31:         tv->tv_usec = ((kstat.ticks % HZ) * 1000000) / HZ;
32:     }
33:     if(tz) {
34:         if((errno = check_user_area(VERIFY_WRITE, tz, sizeof(struct time
zone)))) {
35:             return errno;
36:         }
37:         tz->tz_minuteswest = kstat.tz_minuteswest;
38:         tz->tz_dsttime = kstat.tz_dsttime;
39:     }
40:     return 0;
41: }
```

kernel/syscalls/getuid.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/getuid.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/process.h>
9:
10: #ifdef __DEBUG__
11: #include <fiwix/stdio.h>
12: #endif /*__DEBUG__*/
13:
14: int sys_getuid(void)
15: {
16: #ifdef __DEBUG__
17:     printk("(pid %d) sys_getuid() -> %d\n", current->pid, current->uid);
18: #endif /*__DEBUG__*/
19:
20:     return current->uid;
21: }
```


kernel/syscalls/ioctl.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/ioctl.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/process.h>
9: #include <fiwix/errno.h>
10:
11: #ifdef __DEBUG__
12: #include <fiwix/stdio.h>
13: #endif /*__DEBUG__*/
14:
15: int sys_ioctl(unsigned int fd, int cmd, unsigned long int arg)
16: {
17:     int errno;
18:     struct inode *i;
19:
20: #ifdef __DEBUG__
21:     printk("(pid %d) sys_ioctl(%d, 0x%x, 0x%08x) -> ", current->pid, fd, cmd
, arg);
22: #endif /*__DEBUG__*/
23:
24:     CHECK_UFD(fd);
25:     i = fd_table[current->fd[fd]].inode;
26:     if(i->fsop && i->fsop->ioctl) {
27:         errno = i->fsop->ioctl(i, cmd, arg);
28:
29: #ifdef __DEBUG__
30:         printk("%d\n", errno);
31: #endif /*__DEBUG__*/
32:
33:         return errno;
34:     }
35:
36: #ifdef __DEBUG__
37:     printk("%d\n", -ENOTTY);
38: #endif /*__DEBUG__*/
39:
40:     return -ENOTTY;
41: }
```

kernel/syscalls/ioperm.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/ioperm.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/process.h>
9: #include <fiwix/errno.h>
10:
11: #ifdef __DEBUG__
12: #include <fiwix/stdio.h>
13: #endif /*__DEBUG__ */
14:
15: int sys_ioperm(unsigned long int from, unsigned long int num, int turn_on)
16: {
17: #ifdef __DEBUG__
18:     printk("(pid %d) sys_ioperm(0x%08x, 0x%08x, 0x%08x)\n", current->pid, fr
om, num, turn_on);
19: #endif /*__DEBUG__ */
20:
21:     if(!IS_SUPERUSER) {
22:         return -EPERM;
23:     }
24:
25:     /* FIXME: to be implemented */
26:
27:     return 0;
28: }
```

kernel/syscalls/iopl.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/iopl.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: /*
9:  * Chapter Input/Output of IA-32 Intel(R) Architecture Software Developer's
10:  * Manual Volume 1 Basic Architecture, says the processor permits applications
11:  * to access I/O ports in either of two ways: by using I/O address space or by
12:  * using memory-mapped I/O. Linux 2.0 and Fiwix uses the first one.
13:  *
14:  * This system call sets the IOPL field in the EFLAGS register to the value of
15:  * 'level' (which is presumably zero), so the current process will have
16:  * privileges to use any port, even if that port is beyond of the default size
17:  * of the I/O bitmap in TSS (which is IO_BITMAP_SIZE = 32). Otherwise the
18:  * processor checks the I/O permission bit map to determine if access to a
19:  * specific I/O port is allowed.
20:  *
21:  * So, we leave it here as in Linux 2.0. That means, leaving to I/O bit map to
22:  * control the ports up to 0x3FF, and the rest of ports will be controlled by
23:  * using this system call.
24:  */
25:
26: #include <fiwix/process.h>
27: #include <fiwix/segments.h>
28: #include <fiwix/sigcontext.h>
29: #include <fiwix/errno.h>
30:
31: #ifdef __DEBUG__
32: #include <fiwix/stdio.h>
33: #endif /*__DEBUG__*/
34:
35: int sys_iopl(int level, int arg2, int arg3, int arg4, int arg5, struct sigcontext
t *sc)
36: {
37: #ifdef __DEBUG__
38:     printk("(pid %d) sys_iopl(%d) -> ", current->pid, level);
39: #endif /*__DEBUG__*/
40:     if(level > USR_PL) {
41: #ifdef __DEBUG__
42:         printk("--EINVAL\n");
43: #endif /*__DEBUG__*/
44:         return -EINVAL;
45:     }
46:     if(!IS_SUPERUSER) {
47: #ifdef __DEBUG__
48:         printk("--EPERM\n");
49: #endif /*__DEBUG__*/
50:         return -EPERM;
51:     }
52:
53:     sc->eflags = (sc->eflags & 0xFFFFCFFF) | (level << EF_IOPL);
54: #ifdef __DEBUG__
55:     printk("0\n");
56: #endif /*__DEBUG__*/
57:     return 0;
58: }

```

kernel/syscalls/kill.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/kill.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/process.h>
10: #include <fiwix/errno.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #endif /*__DEBUG__*/
15:
16: int sys_kill(__pid_t pid, __sigset_t signum)
17: {
18:     int count;
19:     struct proc *p;
20:
21: #ifdef __DEBUG__
22:     printk("(pid %d) sys_kill(%d, %d)\n", current->pid, pid, signum);
23: #endif /*__DEBUG__*/
24:
25:     if(signum < 1 || signum > NSIG) {
26:         return -EINVAL;
27:     }
28:     if(pid == -1) {
29:         count = 0;
30:         FOR_EACH_PROCESS(p) {
31:             if(p->pid > 1 && p != current) {
32:                 count++;
33:                 send_sig(p, signum);
34:             }
35:             p = p->next;
36:         }
37:         return count ? 0 : -ESRCH;
38:     }
39:     if(!pid) {
40:         return kill_pgrp(current->pgrp, signum);
41:     }
42:     if(pid < 1) {
43:         return kill_pgrp(-pid, signum);
44:     }
45:
46:     return kill_pid(pid, signum);
47: }
```

kernel/syscalls/link.c

Page 1/2

```

1: /*
2:  * fiwix/kernel/syscalls/link.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/stat.h>
10: #include <fiwix/errno.h>
11: #include <fiwix/string.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #include <fiwix/process.h>
16: #endif /* __DEBUG__ */
17:
18: int sys_link(const char *oldname, const char *newname)
19: {
20:     struct inode *i, *dir, *i_new, *dir_new;
21:     char *tmp_oldname, *tmp_newname, *basename;
22:     int errno;
23:
24: #ifdef __DEBUG__
25:     printk("(pid %d) sys_link('%s', '%s')\n", current->pid, oldname, newname
);
26: #endif /* __DEBUG__ */
27:
28:     if((errno = malloc_name(oldname, &tmp_oldname)) < 0) {
29:         return errno;
30:     }
31:     if((errno = malloc_name(newname, &tmp_newname)) < 0) {
32:         free_name(tmp_oldname);
33:         return errno;
34:     }
35:
36:     if((errno = namei(tmp_oldname, &i, &dir, !FOLLOW_LINKS))) {
37:         if(dir) {
38:             iput(dir);
39:         }
40:         free_name(tmp_oldname);
41:         free_name(tmp_newname);
42:         return errno;
43:     }
44:     if(S_ISDIR(i->i_mode)) {
45:         iput(i);
46:         iput(dir);
47:         free_name(tmp_oldname);
48:         free_name(tmp_newname);
49:         return -EPERM;
50:     }
51:     if(IS_RDONLY_FS(i)) {
52:         iput(i);
53:         iput(dir);
54:         free_name(tmp_oldname);
55:         free_name(tmp_newname);
56:         return -EROFS;
57:     }
58:     if(i->i_nlink == LINK_MAX) {
59:         iput(i);
60:         iput(dir);
61:         free_name(tmp_oldname);
62:         free_name(tmp_newname);
63:         return -EMLINK;
64:     }
65:
66:     basename = get_basename(tmp_newname);

```

kernel/syscalls/link.c

Page 2/2

```
67:         if((errno = namei(tmp_newname, &i_new, &dir_new, !FOLLOW_LINKS))) {
68:             if(!dir_new) {
69:                 iput(i);
70:                 iput(dir);
71:                 free_name(tmp_oldname);
72:                 free_name(tmp_newname);
73:                 return errno;
74:             }
75:         }
76:         if(!errno) {
77:             iput(i);
78:             iput(dir);
79:             iput(i_new);
80:             iput(dir_new);
81:             free_name(tmp_oldname);
82:             free_name(tmp_newname);
83:             return -EEXIST;
84:         }
85:         if(i->dev != dir_new->dev) {
86:             iput(i);
87:             iput(dir);
88:             iput(dir_new);
89:             free_name(tmp_oldname);
90:             free_name(tmp_newname);
91:             return -EXDEV;
92:         }
93:         if(check_permission(TO_EXEC | TO_WRITE, dir_new) < 0) {
94:             iput(i);
95:             iput(dir);
96:             iput(dir_new);
97:             free_name(tmp_oldname);
98:             free_name(tmp_newname);
99:             return -EACCES;
100:        }
101:
102:        if(dir_new->fsop && dir_new->fsop->link) {
103:            errno = dir_new->fsop->link(i, dir_new, basename);
104:        } else {
105:            errno = -EPERM;
106:        }
107:        iput(i);
108:        iput(dir);
109:        iput(dir_new);
110:        free_name(tmp_oldname);
111:        free_name(tmp_newname);
112:        return errno;
113:    }
```

kernel/syscalls/llseek.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/llseek.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/process.h>
11: #include <fiwix/errno.h>
12: #include <fiwix/string.h>
13:
14: #ifdef __DEBUG__
15: #include <fiwix/stdio.h>
16: #endif /* __DEBUG__ */
17:
18: int sys_llseek(unsigned int ufd, unsigned long int offset_high, unsigned long in
t offset_low, __loff_t *result, unsigned int whence)
19: {
20:     struct inode *i;
21:     __loff_t offset;
22:     __loff_t new_offset;
23:     int errno;
24:
25: #ifdef __DEBUG__
26:     printk(" (pid %d) sys_llseek(%d, %u, %u, %08x, %d)", current->pid, ufd, o
ffset_high, offset_low, result, whence);
27: #endif /* __DEBUG__ */
28:
29:     CHECK_UFD(ufd);
30:     if((errno = check_user_area(VERIFY_WRITE, result, sizeof(__loff_t)))) {
31:         return errno;
32:     }
33:     i = fd_table[current->fd[ufd]].inode;
34:     offset = (__loff_t) (((__loff_t)offset_high << 32) | offset_low);
35:     switch(whence) {
36:         case SEEK_SET:
37:             new_offset = offset;
38:             break;
39:         case SEEK_CUR:
40:             new_offset = fd_table[current->fd[ufd]].offset + offset;
41:             break;
42:         case SEEK_END:
43:             new_offset = i->i_size + offset;
44:             break;
45:         default:
46:             return -EINVAL;
47:     }
48:     fd_table[current->fd[ufd]].offset = new_offset;
49:
50:     memcpy_b(result, &new_offset, sizeof(__loff_t));
51:
52: #ifdef __DEBUG__
53:     printk(" -> returning %u\n", *result);
54: #endif /* __DEBUG__ */
55:
56:     return 0;
57: }

```

kernel/syscalls/lseek.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/lseek.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/syscalls.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/errno.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #include <fiwix/process.h>
16: #endif /* __DEBUG__ */
17:
18: int sys_lseek(unsigned int ufd, __off_t offset, unsigned int whence)
19: {
20:     struct inode *i;
21:     __off_t new_offset;
22:
23: #ifdef __DEBUG__
24:     printk("(pid %d) sys_lseek(%d, %d, %d)", current->pid, ufd, offset, when
ce);
25: #endif /* __DEBUG__ */
26:
27:     CHECK_UFD(ufd);
28:
29:     i = fd_table[current->fd[ufd]].inode;
30:     switch(whence) {
31:         case SEEK_SET:
32:             new_offset = offset;
33:             break;
34:         case SEEK_CUR:
35:             new_offset = fd_table[current->fd[ufd]].offset + offset;
36:             break;
37:         case SEEK_END:
38:             new_offset = i->i_size + offset;
39:             break;
40:         default:
41:             return -EINVAL;
42:     }
43:     if((int)new_offset < 0) {
44:         return -EINVAL;
45:     }
46:     if(i->fsop && i->fsop->lseek) {
47:         fd_table[current->fd[ufd]].offset = new_offset;
48:         new_offset = i->fsop->lseek(i, new_offset);
49:     } else {
50:         return -EPERM;
51:     }
52:
53: #ifdef __DEBUG__
54:     printk("-> returning %d\n", new_offset);
55: #endif /* __DEBUG__ */
56:
57:     return new_offset;
58: }

```


kernel/syscalls/lstat.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/lstat.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/stat.h>
10: #include <fiwix/string.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #include <fiwix/process.h>
15: #endif /*__DEBUG__*/
16:
17: int sys_lstat(const char *filename, struct old_stat *statbuf)
18: {
19:     struct inode *i;
20:     char *tmp_name;
21:     int errno;
22:
23: #ifdef __DEBUG__
24:     printk("(pid %d) sys_lstat('%s', 0x%08x) -> returning structure\n", curr
ent->pid, filename, (unsigned int)statbuf);
25: #endif /*__DEBUG__*/
26:
27:     if((errno = check_user_area(VERIFY_WRITE, statbuf, sizeof(struct old_sta
t)))) {
28:         return errno;
29:     }
30:     if((errno = malloc_name(filename, &tmp_name)) < 0) {
31:         return errno;
32:     }
33:     if((errno = namei(tmp_name, &i, NULL, !FOLLOW_LINKS))) {
34:         free_name(tmp_name);
35:         return errno;
36:     }
37:     statbuf->st_dev = i->dev;
38:     statbuf->st_ino = i->inode;
39:     statbuf->st_mode = i->i_mode;
40:     statbuf->st_nlink = i->i_nlink;
41:     statbuf->st_uid = i->i_uid;
42:     statbuf->st_gid = i->i_gid;
43:     statbuf->st_rdev = i->rdev;
44:     statbuf->st_size = i->i_size;
45:     statbuf->st_atime = i->i_atime;
46:     statbuf->st_mtime = i->i_mtime;
47:     statbuf->st_ctime = i->i_ctime;
48:     iput(i);
49:     free_name(tmp_name);
50:     return 0;
51: }

```

kernel/syscalls/Makefile

Page 1/1

```
1: # fiwix/kernel/syscalls/Makefile
2: #
3: # Copyright 2018, Jordi Sanfeliu. All rights reserved.
4: # Distributed under the terms of the Fiwix License.
5: #
6:
7: .S.o:
8:     $(CC) -traditional -I$(INCLUDE) -c -o $@ $<
9: .c.o:
10:     $(CC) $(CFLAGS) -c -o $@ $<
11:
12: SRC = $(wildcard *.c)
13: OBJS = $(patsubst %.c,%.o,$(SRC))
14:
15: lib:     $(OBJS)
16:           $(LD) $(LDFLAGS) -r $(OBJS) -o syscalls.o
17:
18: clean:
19:           rm -f *.o
20:
```

kernel/syscalls/mkdir.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/mkdir.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/stat.h>
11: #include <fiwix/errno.h>
12: #include <fiwix/string.h>
13:
14: #ifdef __DEBUG__
15: #include <fiwix/stdio.h>
16: #include <fiwix/process.h>
17: #endif /*__DEBUG__*/
18:
19: int sys_mkdir(const char *dirname, __mode_t mode)
20: {
21:     struct inode *i, *dir;
22:     char *tmp_dirname, *basename;
23:     int errno;
24:
25: #ifdef __DEBUG__
26:     printk("(pid %d) sys_mkdir('%s', %o)\n", current->pid, dirname, mode);
27: #endif /*__DEBUG__*/
28:
29:     if((errno = malloc_name(dirname, &tmp_dirname)) < 0) {
30:         return errno;
31:     }
32:     basename = remove_trailing_slash(tmp_dirname);
33:     if((errno = namei(basename, &i, &dir, !FOLLOW_LINKS)) {
34:         if(!dir) {
35:             free_name(tmp_dirname);
36:             return errno;
37:         }
38:     }
39:     if(!errno) {
40:         iput(i);
41:         iput(dir);
42:         free_name(tmp_dirname);
43:         return -EEXIST;
44:     }
45:     if(IS_RDONLY_FS(dir)) {
46:         iput(dir);
47:         free_name(tmp_dirname);
48:         return -EROFS;
49:     }
50:
51:     if(check_permission(TO_EXEC | TO_WRITE, dir) < 0) {
52:         iput(dir);
53:         free_name(tmp_dirname);
54:         return -EACCES;
55:     }
56:
57:     basename = get_basename(basename);
58:     if(dir->fsop && dir->fsop->mkdir) {
59:         errno = dir->fsop->mkdir(dir, basename, mode);
60:     } else {
61:         errno = -EPERM;
62:     }
63:     iput(dir);
64:     free_name(tmp_dirname);
65:     return errno;
66: }

```

kernel/syscalls/mknod.c

Page 1/2

```

1: /*
2:  * fiwix/kernel/syscalls/mknod.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/stat.h>
11: #include <fiwix/errno.h>
12: #include <fiwix/string.h>
13:
14: #ifdef __DEBUG__
15: #include <fiwix/stdio.h>
16: #include <fiwix/process.h>
17: #endif /*__DEBUG__ */
18:
19: int sys_mknod(const char *pathname, __mode_t mode, __dev_t dev)
20: {
21:     struct inode *i, *dir;
22:     char *tmp_name, *basename;
23:     int errno;
24:
25: #ifdef __DEBUG__
26:     printk("(pid %d) sys_mknod('%s', %d, %x)\n", current->pid, pathname, mod
e, dev);
27: #endif /*__DEBUG__ */
28:
29:     if(!S_ISCHR(mode) && !S_ISBLK(mode) && !S_ISFIFO(mode)) {
30:         return -EINVAL;
31:     }
32:     if(!S_ISFIFO(mode) && !IS_SUPERUSER) {
33:         return -EPERM;
34:     }
35:
36:     if((errno = malloc_name(pathname, &tmp_name)) < 0) {
37:         return errno;
38:     }
39:     basename = get_basename(tmp_name);
40:     if((errno = namei(tmp_name, &i, &dir, !FOLLOW_LINKS))) {
41:         if(!dir) {
42:             free_name(tmp_name);
43:             return errno;
44:         }
45:     }
46:     if(!errno) {
47:         iput(i);
48:         iput(dir);
49:         free_name(tmp_name);
50:         return -EEXIST;
51:     }
52:     if(IS_RDONLY_FS(dir)) {
53:         iput(dir);
54:         free_name(tmp_name);
55:         return -EROFS;
56:     }
57:     if(check_permission(TO_EXEC | TO_WRITE, dir) < 0) {
58:         iput(dir);
59:         free_name(tmp_name);
60:         return -EACCES;
61:     }
62:
63:     if(dir->fsop && dir->fsop->mknod) {
64:         errno = dir->fsop->mknod(dir, basename, mode, dev);
65:     } else {
66:         errno = -EPERM;

```

kernel/syscalls/mknod.c

Page 2/2

```
67:         }
68:         iput(dir);
69:         free_name(tmp_name);
70:         return errno;
71: }
```

kernel/syscalls/mount.c

Page 1/4

```

1: /*
2:  * fiwix/kernel/syscalls/mount.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/stat.h>
11: #include <fiwix/buffer.h>
12: #include <fiwix/mman.h>
13: #include <fiwix/filesystems.h>
14: #include <fiwix/mm.h>
15: #include <fiwix/errno.h>
16: #include <fiwix/string.h>
17:
18: #ifdef __DEBUG__
19: #include <fiwix/stdio.h>
20: #include <fiwix/process.h>
21: #endif /* __DEBUG__ */
22:
23: int sys_mount(const char *source, const char *target, const char *fstype, unsigned long int flags, const void *data)
24: {
25:     struct inode *i_source, *i_target;
26:     struct mount *mt;
27:     struct filesystems *fs;
28:     struct vma *vma;
29:     char *tmp_source, *tmp_target, *tmp_fstype;
30:     __dev_t dev;
31:     int len, errno;
32:
33: #ifdef __DEBUG__
34:     printk("(pid %d) sys_mount(%s, %s, %s, 0x%08x, 0x%08x\n", current->pid,
source, target, (int)fstype ? fstype : "<NULL>", flags, data);
35: #endif /* __DEBUG__ */
36:
37:     if(!IS_SUPERUSER) {
38:         return -EPERM;
39:     }
40:
41:     if((errno = malloc_name(target, &tmp_target)) < 0) {
42:         return errno;
43:     }
44:     if((errno = namei(tmp_target, &i_target, NULL, FOLLOW_LINKS))) {
45:         free_name(tmp_target);
46:         return errno;
47:     }
48:     if(!S_ISDIR(i_target->i_mode)) {
49:         iput(i_target);
50:         free_name(tmp_target);
51:         return -ENOTDIR;
52:     }
53:     if((flags & MS_MGC_VAL) == MS_MGC_VAL) {
54:         flags &= ~MS_MGC_MSK;
55:     }
56:
57:     if(flags & MS_REMOUNT) {
58:         if(!(mt = get_mount_point(i_target))) {
59:             iput(i_target);
60:             free_name(tmp_target);
61:             return -EINVAL;
62:         }
63:         fs = mt->fs;
64:         if(fs->fsop && fs->fsop->remount_fs) {
65:             if((errno = fs->fsop->remount_fs(&mt->sb, flags))) {

```

kernel/syscalls/mount.c

Page 2/4

```

66:             iput(i_target);
67:             free_name(tmp_target);
68:             return errno;
69:         }
70:     } else {
71:         iput(i_target);
72:         free_name(tmp_target);
73:         return -EINVAL;
74:     }
75:
76:     /* switching from RW to RO */
77:     if(flags & MS_RDONLY && !(mt->sb.flags & MS_RDONLY)) {
78:         dev = mt->dev;
79:         /*
80:          * FIXME: if there are files opened in RW mode then
81:          * we can't continue and must return -EBUSY.
82:          */
83:         if(fs->fsop && fs->fsop->release_superblock) {
84:             fs->fsop->release_superblock(&mt->sb);
85:         }
86:         sync_superblocks(dev);
87:         sync_inodes(dev);
88:         sync_buffers(dev);
89:     }
90:
91:     mt->sb.flags &= ~MS_RDONLY;
92:     mt->sb.flags |= (flags & MS_RDONLY);
93:     iput(i_target);
94:     free_name(tmp_target);
95:     return 0;
96: }
97:
98: if(i_target->mount_point) {
99:     iput(i_target);
100:    free_name(tmp_target);
101:    return -EBUSY;
102: }
103:
104: /* check the validity of fstype */
105: if(!(vma = find_vma_region((unsigned int)fstype))) {
106:     iput(i_target);
107:     free_name(tmp_target);
108:     return -EFAULT;
109: }
110: if(!(vma->prot & PROT_READ)) {
111:     iput(i_target);
112:     free_name(tmp_target);
113:     return -EFAULT;
114: }
115: len = MIN(vma->end - (unsigned int)fstype, PAGE_SIZE - 1);
116: if(!(tmp_fstype = (char *)kmalloc())) {
117:     iput(i_target);
118:     free_name(tmp_target);
119:     return -ENOMEM;
120: }
121: memcpy_b(tmp_fstype, fstype, len);
122:
123: if(!(fs = get_filesystem(fstype))) {
124:     iput(i_target);
125:     free_name(tmp_target);
126:     free_name(tmp_fstype);
127:     return -ENODEV;
128: }
129: dev = fs->fsop->fsdev;
130:
131: if((errno = malloc_name(source, &tmp_source)) < 0) {
132:     iput(i_target);

```

kernel/syscalls/mount.c

Page 3/4

```

133:         free_name(tmp_target);
134:         free_name(tmp_fstype);
135:         return errno;
136:     }
137:     if(fs->fsop->flags == FSOP_REQUIRES_DEV) {
138:         if((errno = namei(tmp_source, &i_source, NULL, FOLLOW_LINKS)) {
139:             iput(i_target);
140:             free_name(tmp_target);
141:             free_name(tmp_fstype);
142:             free_name(tmp_source);
143:             return errno;
144:         }
145:         if(!S_ISBLK(i_source->i_mode)) {
146:             iput(i_target);
147:             iput(i_source);
148:             free_name(tmp_target);
149:             free_name(tmp_fstype);
150:             free_name(tmp_source);
151:             return -ENOTBLK;
152:         }
153:         if(i_source->fsop && i_source->fsop->open) {
154:             if((errno = i_source->fsop->open(i_source, NULL)) {
155:                 iput(i_target);
156:                 iput(i_source);
157:                 free_name(tmp_target);
158:                 free_name(tmp_fstype);
159:                 free_name(tmp_source);
160:                 return errno;
161:             }
162:         } else {
163:             iput(i_target);
164:             iput(i_source);
165:             free_name(tmp_target);
166:             free_name(tmp_fstype);
167:             free_name(tmp_source);
168:             return -EINVAL;
169:         }
170:         dev = i_source->rdev;
171:     }
172:
173:     if(! (mt = get_free_mount_point(dev))) {
174:         if(fs->fsop->flags == FSOP_REQUIRES_DEV) {
175:             i_source->fsop->close(i_source, NULL);
176:             iput(i_source);
177:         }
178:         iput(i_target);
179:         free_name(tmp_target);
180:         free_name(tmp_fstype);
181:         free_name(tmp_source);
182:         return -EBUSY;
183:     }
184:
185:     mt->sb.flags = flags;
186:     if(fs->fsop && fs->fsop->read_superblock) {
187:         if((errno = fs->fsop->read_superblock(dev, &mt->sb)) {
188:             i_source->fsop->close(i_source, NULL);
189:             if(fs->fsop->flags == FSOP_REQUIRES_DEV) {
190:                 iput(i_source);
191:             }
192:             iput(i_target);
193:             release_mount_point(mt);
194:             free_name(tmp_target);
195:             free_name(tmp_fstype);
196:             free_name(tmp_source);
197:             return errno;
198:         }
199:     } else {

```


kernel/syscalls/mount.c

Page 4/4

```
200:             if (fs->fsop->flags == FSOP_REQUIRES_DEV) {
201:                 iput (i_source);
202:             }
203:             iput (i_target);
204:             release_mount_point (mt);
205:             free_name (tmp_target);
206:             free_name (tmp_fstype);
207:             free_name (tmp_source);
208:             return -EINVAL;
209:         }
210:
211:         mt->dev = dev;
212:         strncpy (mt->devname, tmp_source, DEVNAME_MAX);
213:         strcpy (mt->dirname, tmp_target);
214:         mt->sb.dir = i_target;
215:         mt->fs = fs;
216:         i_target->mount_point = mt->sb.root;
217:         free_name (tmp_target);
218:         free_name (tmp_fstype);
219:         free_name (tmp_source);
220:         return 0;
221:     }
```

kernel/syscalls/mprotect.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/mprotect.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/mman.h>
10: #include <fiwix/mm.h>
11: #include <fiwix/fcntl.h>
12: #include <fiwix/errno.h>
13:
14: #ifdef __DEBUG__
15: #include <fiwix/stdio.h>
16: #include <fiwix/process.h>
17: #endif /*__DEBUG__ */
18:
19: int sys_mprotect(unsigned int addr, __size_t length, int prot)
20: {
21:     struct vma *vma;
22:
23: #ifdef __DEBUG__
24:     printk("(pid %d) sys_mprotect(0x%08x, %d, %d)\n", current->pid, addr, le
ngth, prot);
25: #endif /*__DEBUG__ */
26:
27:     if((addr & ~PAGE_MASK) || length < 0) {
28:         return -EINVAL;
29:     }
30:     if(prot > (PROT_READ | PROT_WRITE | PROT_EXEC)) {
31:         return -EINVAL;
32:     }
33:     if(!(vma = find_vma_region(addr))) {
34:         return -ENOMEM;
35:     }
36:     length = PAGE_ALIGN(length);
37:     if((addr + length) > vma->end) {
38:         return -ENOMEM;
39:     }
40:     if(vma->inode && (vma->flags & MAP_SHARED)) {
41:         if(prot & PROT_WRITE) {
42:             if(!(vma->o_mode & (O_WRONLY | O_RDWR))) {
43:                 return -EACCES;
44:             }
45:         }
46:     }
47:
48:     return do_mprotect(vma, addr, length, prot);
49: }

```

kernel/syscalls/munmap.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/munmap.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/mman.h>
10:
11: #ifdef __DEBUG__
12: #include <fiwix/stdio.h>
13: #include <fiwix/process.h>
14: #endif /*__DEBUG__*/
15:
16: int sys_munmap(unsigned int addr, __size_t length)
17: {
18: #ifdef __DEBUG__
19:     printk("(pid %d) sys_munmap(0x%08x, %d)\n", current->pid, addr, length);
20: #endif /*__DEBUG__*/
21:     return do_munmap(addr, length);
22: }
```

kernel/syscalls/nanosleep.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/nanosleep.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/time.h>
11: #include <fiwix/timer.h>
12: #include <fiwix/process.h>
13: #include <fiwix/sched.h>
14: #include <fiwix/sleep.h>
15: #include <fiwix/errno.h>
16:
17: #ifdef __DEBUG__
18: #include <fiwix/stdio.h>
19: #endif /*__DEBUG__ */
20:
21: int sys_nanosleep(const struct timespec *req, struct timespec *rem)
22: {
23:     int errno;
24:     long int nsec;
25:
26: #ifdef __DEBUG__
27:     printk("(pid %d) sys_nanosleep(0x%08x, 0x%08x)\n", current->pid, (unsigned
ed int) req, (unsigned int) rem);
28: #endif /*__DEBUG__ */
29:
30:     if((errno = check_user_area(VERIFY_READ, req, sizeof(struct timespec))))
{
31:         return errno;
32:     }
33:     if(req->tv_sec < 0 || req->tv_nsec >= 1000000000L || req->tv_nsec < 0) {
34:         return -EINVAL;
35:     }
36:
37:     /*
38:      * Since the current maximum precision of the kernel is only 10ms, we
39:      * need to convert any lower request to a minimum of 10ms, even knowing
40:      * that this might increase the sleep a bit more than the requested.
41:      */
42:     nsec = req->tv_nsec;
43:     if(nsec < 100000000L) {
44:         nsec *= 10;
45:     }
46:
47:     current->timeout = (req->tv_sec * HZ) + (nsec * HZ / 1000000000L);
48:     if(current->timeout) {
49:         sleep(&sys_nanosleep, PROC_INTERRUPTIBLE);
50:         if(current->timeout) {
51:             if(rem) {
52:                 if((errno = check_user_area(VERIFY_WRITE, rem, s
sizeof(struct timespec)))) {
53:                     return errno;
54:                 }
55:                 rem->tv_sec = current->timeout / HZ;
56:                 rem->tv_nsec = (current->timeout % HZ) * 100000
000L / HZ;
57:             }
58:             return -EINTR;
59:         }
60:     }
61:     return 0;
62: }

```

kernel/syscalls/newfstat.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/newfstat.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/statbuf.h>
10: #include <fiwix/process.h>
11: #include <fiwix/errno.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #endif /*__DEBUG__*/
16:
17: int sys_newfstat(unsigned int ufd, struct new_stat *statbuf)
18: {
19:     struct inode *i;
20:     int errno;
21:
22: #ifdef __DEBUG__
23:     printk("(pid %d) sys_newfstat(%d, 0x%08x) -> returning structure\n", cur
rent->pid, ufd, (unsigned int )statbuf);
24: #endif /*__DEBUG__*/
25:
26:     CHECK_UFD(ufd);
27:     if((errno = check_user_area(VERIFY_WRITE, statbuf, sizeof(struct new_sta
t)))) {
28:         return errno;
29:     }
30:     i = fd_table[current->fd[ufd]].inode;
31:     statbuf->st_dev = i->dev;
32:     statbuf->__pad1 = 0;
33:     statbuf->st_ino = i->inode;
34:     statbuf->st_mode = i->i_mode;
35:     statbuf->st_nlink = i->i_nlink;
36:     statbuf->st_uid = i->i_uid;
37:     statbuf->st_gid = i->i_gid;
38:     statbuf->st_rdev = i->rdev;
39:     statbuf->__pad2 = 0;
40:     statbuf->st_size = i->i_size;
41:     statbuf->st_blksize = i->sb->s_blocksize;
42:     statbuf->st_blocks = i->i_blocks;
43:     if(!i->i_blocks) {
44:         statbuf->st_blocks = (i->i_size / i->sb->s_blocksize * 2);
45:         statbuf->st_blocks++;
46:     }
47:     statbuf->st_atime = i->i_atime;
48:     statbuf->__unused1 = 0;
49:     statbuf->st_mtime = i->i_mtime;
50:     statbuf->__unused2 = 0;
51:     statbuf->st_ctime = i->i_ctime;
52:     statbuf->__unused3 = 0;
53:     statbuf->__unused4 = 0;
54:     statbuf->__unused5 = 0;
55:     return 0;
56: }

```

kernel/syscalls/newlstat.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/newlstat.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/statbuf.h>
10: #include <fiwix/string.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #include <fiwix/process.h>
15: #endif /*__DEBUG__*/
16:
17: int sys_newlstat(const char *filename, struct new_stat *statbuf)
18: {
19:     struct inode *i;
20:     char *tmp_name;
21:     int errno;
22:
23: #ifdef __DEBUG__
24:     printk("(pid %d) sys_newlstat('%s', 0x%08x) -> returning structure\n", c
urrent->pid, filename, (unsigned int)statbuf);
25: #endif /*__DEBUG__*/
26:
27:     if((errno = check_user_area(VERIFY_WRITE, statbuf, sizeof(struct new_sta
t)))) {
28:         return errno;
29:     }
30:     if((errno = malloc_name(filename, &tmp_name)) < 0) {
31:         return errno;
32:     }
33:     if((errno = namei(tmp_name, &i, NULL, !FOLLOW_LINKS))) {
34:         free_name(tmp_name);
35:         return errno;
36:     }
37:     statbuf->st_dev = i->dev;
38:     statbuf->__pad1 = 0;
39:     statbuf->st_ino = i->inode;
40:     statbuf->st_mode = i->i_mode;
41:     statbuf->st_nlink = i->i_nlink;
42:     statbuf->st_uid = i->i_uid;
43:     statbuf->st_gid = i->i_gid;
44:     statbuf->st_rdev = i->rdev;
45:     statbuf->__pad2 = 0;
46:     statbuf->st_size = i->i_size;
47:     statbuf->st_blksize = i->sb->s_blocksize;
48:     statbuf->st_blocks = i->i_blocks;
49:     if(!i->i_blocks) {
50:         statbuf->st_blocks = (i->i_size / i->sb->s_blocksize) * 2;
51:         statbuf->st_blocks++;
52:     }
53:     statbuf->st_atime = i->i_atime;
54:     statbuf->__unused1 = 0;
55:     statbuf->st_mtime = i->i_mtime;
56:     statbuf->__unused2 = 0;
57:     statbuf->st_ctime = i->i_ctime;
58:     statbuf->__unused3 = 0;
59:     statbuf->__unused4 = 0;
60:     statbuf->__unused5 = 0;
61:     iput(i);
62:     free_name(tmp_name);
63:     return 0;
64: }

```

kernel/syscalls/newstat.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/newstat.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/statbuf.h>
10: #include <fiwix/string.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #include <fiwix/process.h>
15: #endif /*__DEBUG__*/
16:
17: int sys_newstat(const char *filename, struct new_stat *statbuf)
18: {
19:     struct inode *i;
20:     char *tmp_name;
21:     int errno;
22:
23: #ifdef __DEBUG__
24:     printk("(pid %d) sys_newstat('%s', 0x%08x) -> returning structure\n", cu
rrent->pid, filename, (unsigned int)statbuf);
25: #endif /*__DEBUG__*/
26:
27:     if((errno = check_user_area(VERIFY_WRITE, statbuf, sizeof(struct new_sta
t)))) {
28:         return errno;
29:     }
30:     if((errno = malloc_name(filename, &tmp_name)) < 0) {
31:         return errno;
32:     }
33:     if((errno = namei(tmp_name, &i, NULL, FOLLOW_LINKS))) {
34:         free_name(tmp_name);
35:         return errno;
36:     }
37:     statbuf->st_dev = i->dev;
38:     statbuf->__pad1 = 0;
39:     statbuf->st_ino = i->inode;
40:     statbuf->st_mode = i->i_mode;
41:     statbuf->st_nlink = i->i_nlink;
42:     statbuf->st_uid = i->i_uid;
43:     statbuf->st_gid = i->i_gid;
44:     statbuf->st_rdev = i->rdev;
45:     statbuf->__pad2 = 0;
46:     statbuf->st_size = i->i_size;
47:     statbuf->st_blksize = i->sb->s_blocksize;
48:     statbuf->st_blocks = i->i_blocks;
49:     if(!i->i_blocks) {
50:         statbuf->st_blocks = (i->i_size / i->sb->s_blocksize) * 2;
51:         statbuf->st_blocks++;
52:     }
53:     statbuf->st_atime = i->i_atime;
54:     statbuf->__unused1 = 0;
55:     statbuf->st_mtime = i->i_mtime;
56:     statbuf->__unused2 = 0;
57:     statbuf->st_ctime = i->i_ctime;
58:     statbuf->__unused3 = 0;
59:     statbuf->__unused4 = 0;
60:     statbuf->__unused5 = 0;
61:     iput(i);
62:     free_name(tmp_name);
63:     return 0;
64: }

```

kernel/syscalls/newuname.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/newuname.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/utsname.h>
10: #include <fiwix/errno.h>
11: #include <fiwix/string.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #include <fiwix/process.h>
16: #endif /* __DEBUG__ */
17:
18: int sys_newuname(struct new_utsname *uname)
19: {
20:     int errno;
21:
22: #ifdef __DEBUG__
23:     printk("(pid %d) sys_newuname(0x%08x)\n", current->pid, (int)uname);
24: #endif /* __DEBUG__ */
25:
26:     if((errno = check_user_area(VERIFY_WRITE, uname, sizeof(struct new_utsna
me)))) {
27:         return errno;
28:     }
29:     if(!uname) {
30:         return -EFAULT;
31:     }
32:     memcpy_b(uname, &sys_utsname, sizeof(struct new_utsname));
33:     return 0;
34: }
```


kernel/syscalls/old_mmap.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/old_mmap.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/mman.h>
10: #include <fiwix/mm.h>
11: #include <fiwix/fcntl.h>
12: #include <fiwix/errno.h>
13: #include <fiwix/string.h>
14:
15: #ifdef __DEBUG__
16: #include <fiwix/stdio.h>
17: #include <fiwix/process.h>
18: #endif /*__DEBUG__*/
19:
20: int old_mmap(struct mmap *mmap)
21: {
22:     unsigned int page;
23:     struct inode *i;
24:     char flags;
25:     int errno;
26:
27: #ifdef __DEBUG__
28:     printk("(pid %d) old_mmap(0x%08x, %d, 0x%02x, 0x%02x, %d, 0x%08x) -> ",
current->pid, mmap->start, mmap->length, mmap->prot, mmap->flags, mmap->offse
t);
29: #endif /*__DEBUG__*/
30:
31:     if((errno = check_user_area(VERIFY_READ, mmap, sizeof(struct mmap)))) {
32:         return errno;
33:     }
34:     if(!mmap->length) {
35:         return -EINVAL;
36:     }
37:
38:     i = NULL;
39:     flags = 0;
40:     if(!(mmap->flags & MAP_ANONYMOUS)) {
41:         CHECK_UFD(mmap->fd);
42:         if(!(i = fd_table[current->fd[mmap->fd]].inode)) {
43:             return -EBADF;
44:         }
45:         flags = fd_table[current->fd[mmap->fd]].flags & O_ACCMODE;
46:     }
47:     page = do_mmap(i, mmap->start, mmap->length, mmap->prot, mmap->flags, mm
ap->offset, P_MMAP, flags);
48: #ifdef __DEBUG__
49:     printk("0x%08x\n", page);
50: #endif /*__DEBUG__*/
51:     return page;
52: }

```

kernel/syscalls/old_select.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/old_select.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/syscalls.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/sleep.h>
12: #include <fiwix/sched.h>
13:
14: #ifdef __DEBUG__
15: #include <fiwix/stdio.h>
16: #include <fiwix/process.h>
17: #endif /*__DEBUG__ */
18:
19: int old_select(unsigned long int *params)
20: {
21:     int nfd;
22:     fd_set *readfds;
23:     fd_set *writefds;
24:     fd_set *exceptfds;
25:     struct timeval *timeout;
26:     int errno;
27:
28: #ifdef __DEBUG__
29:     printk("(pid %d) old_select(0x%08x)\n", current->pid, (int)params);
30: #endif /*__DEBUG__ */
31:
32:     if((errno = check_user_area(VERIFY_READ, (void *)params, sizeof(unsigned
int) * 5))) {
33:         return errno;
34:     }
35:     nfd = *(int *)params;
36:     readfds = *(fd_set **) (params + 1);
37:     writefds = *(fd_set **) (params + 2);
38:     exceptfds = *(fd_set **) (params + 3);
39:     timeout = *(struct timeval **) (params + 4);
40:
41:     return sys_select(nfd, readfds, writefds, exceptfds, timeout);
42: }
```

kernel/syscalls/olduname.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/olduname.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/utsname.h>
10: #include <fiwix/string.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #include <fiwix/process.h>
15: #endif /*__DEBUG__*/
16:
17: int sys_olduname(struct oldold_utsname *uname)
18: {
19:     int errno;
20:
21: #ifdef __DEBUG__
22:     printk("(pid %d) sys_olduname(0x%0x)", current->pid, uname);
23: #endif /*__DEBUG__*/
24:
25:     if((errno = check_user_area(VERIFY_WRITE, uname, sizeof(struct oldold_utsname)))) {
26:         return errno;
27:     }
28:     memcpy_b(&uname->sysname, &sys_utsname.sysname, _OLD_UTSNAME_LENGTH);
29:     memset_b(&uname->sysname + _OLD_UTSNAME_LENGTH, NULL, 1);
30:     memcpy_b(&uname->nodename, &sys_utsname.nodename, _OLD_UTSNAME_LENGTH);
31:     memset_b(&uname->nodename + _OLD_UTSNAME_LENGTH, NULL, 1);
32:     memcpy_b(&uname->release, &sys_utsname.release, _OLD_UTSNAME_LENGTH);
33:     memset_b(&uname->release + _OLD_UTSNAME_LENGTH, NULL, 1);
34:     memcpy_b(&uname->version, &sys_utsname.version, _OLD_UTSNAME_LENGTH);
35:     memset_b(&uname->version + _OLD_UTSNAME_LENGTH, NULL, 1);
36:     memcpy_b(&uname->machine, &sys_utsname.machine, _OLD_UTSNAME_LENGTH);
37:     memset_b(&uname->machine + _OLD_UTSNAME_LENGTH, NULL, 1);
38:     return 0;
39: }
```

kernel/syscalls/open.c

Page 1/3

```

1: /*
2:  * fiwix/kernel/syscalls/open.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/syscalls.h>
9: #include <fiwix/stat.h>
10: #include <fiwix/types.h>
11: #include <fiwix/fcntl.h>
12: #include <fiwix/errno.h>
13: #include <fiwix/stdio.h>
14: #include <fiwix/string.h>
15:
16: int sys_open(const char *filename, int flags, __mode_t mode)
17: {
18:     int fd, ufd;
19:     struct inode *i, *dir;
20:     char *tmp_name, *basename;
21:     int errno, follow_links, perms;
22:
23: #ifdef __DEBUG__
24:     printk("(pid %d) sys_open('%s', %o, %o)\n", current->pid, filename, flag
s, mode);
25: #endif /*__DEBUG__*/
26:
27:     if((errno = malloc_name(filename, &tmp_name)) < 0) {
28:         return errno;
29:     }
30:
31:     basename = get_basename(tmp_name);
32:     follow_links = (flags & O_NOFOLLOW) ? !FOLLOW_LINKS : FOLLOW_LINKS;
33:     if((errno = namei(tmp_name, &i, &dir, follow_links))) {
34:         if(!dir) {
35:             free_name(tmp_name);
36:             if(flags & O_CREAT) {
37:                 return -ENOENT;
38:             }
39:             return errno;
40:         }
41:     }
42:
43: #ifdef __DEBUG__
44:     printk("\t(inode = %d)\n", i ? i->inode : -1);
45: #endif /*__DEBUG__*/
46:
47:     if(!errno) {
48:         if(S_ISLNK(i->i_mode) && (flags & O_NOFOLLOW)) {
49:             iput(i);
50:             iput(dir);
51:             free_name(tmp_name);
52:             return -ELOOP;
53:         }
54:     }
55:
56:     if(flags & O_CREAT) {
57:         if(!errno && (flags & O_EXCL)) {
58:             iput(i);
59:             iput(dir);
60:             free_name(tmp_name);
61:             return -EEXIST;
62:         }
63:         if(check_permission(TO_EXEC | TO_WRITE, dir) < 0) {
64:             iput(i);
65:             iput(dir);
66:             free_name(tmp_name);

```

kernel/syscalls/open.c

Page 2/3

```

67:             return -EACCES;
68:         }
69:         if(errno) {           /* assumes -ENOENT */
70:             if(dir->fsop && dir->fsop->create) {
71:                 errno = dir->fsop->create(dir, basename, mode, &
i);
72:                 if(errno) {
73:                     iput(dir);
74:                     free_name(tmp_name);
75:                     return errno;
76:                 }
77:             } else {
78:                 iput(dir);
79:                 free_name(tmp_name);
80:                 return -EACCES;
81:             }
82:         }
83:     } else {
84:         if(errno) {
85:             iput(dir);
86:             free_name(tmp_name);
87:             return errno;
88:         }
89:         if(S_ISDIR(i->i_mode) && (flags & (O_RDWR | O_WRONLY | O_TRUNC))
) {
90:             iput(i);
91:             iput(dir);
92:             free_name(tmp_name);
93:             return -EISDIR;
94:         }
95:         mode = 0;
96:     }
97:
98:     if((flags & O_ACCMODE) == O_RDONLY) {
99:         perms = TO_READ;
100:    } else if((flags & O_ACCMODE) == O_WRONLY) {
101:        perms = TO_WRITE;
102:    } else {
103:        perms = TO_READ | TO_WRITE;
104:    }
105:    if((errno = check_permission(perms, i))) {
106:        iput(i);
107:        iput(dir);
108:        free_name(tmp_name);
109:        return errno;
110:    }
111:    if((fd = get_new_fd(i)) < 0) {
112:        iput(i);
113:        iput(dir);
114:        free_name(tmp_name);
115:        return fd;
116:    }
117:    if((ufd = get_new_user_fd(0)) < 0) {
118:        release_fd(fd);
119:        iput(i);
120:        iput(dir);
121:        free_name(tmp_name);
122:        return ufd;
123:    }
124:
125:    #ifdef __DEBUG__
126:        printk("\t(ufd = %d)\n", ufd);
127:    #endif /*__DEBUG__*/
128:
129:    fd_table[fd].flags = flags;
130:    current->fd[ufd] = fd;
131:    if(i->fsop && i->fsop->open) {

```

kernel/syscalls/open.c

Page 3/3

```
132:             if((errno = i->fsop->open(i, &fd_table[fd])) < 0) {
133:                 release_fd(fd);
134:                 release_user_fd(ufd);
135:                 iput(i);
136:                 iput(dir);
137:                 free_name(tmp_name);
138:                 return errno;
139:             }
140:             iput(dir);
141:             free_name(tmp_name);
142:             return ufd;
143:         }
144:
145:         printk("WARNING: %s(): file '%s' (inode %d) without the open() method!\n
", __FUNCTION__, tmp_name, i->inode);
146:         release_fd(fd);
147:         release_user_fd(ufd);
148:         iput(i);
149:         iput(dir);
150:         free_name(tmp_name);
151:         return -EINVAL;
152: }
```

kernel/syscalls/pause.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/pause.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/syscalls.h>
9: #include <fiwix/sched.h>
10: #include <fiwix/sleep.h>
11: #include <fiwix/errno.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #include <fiwix/process.h>
16: #endif /* __DEBUG__ */
17:
18: int sys_pause(void)
19: {
20: #ifdef __DEBUG__
21:     printk("(pid %d) sys_pause()\n", current->pid);
22: #endif /* __DEBUG__ */
23:
24:     for(;;) {
25:         if(sleep(&sys_pause, PROC_INTERRUPTIBLE)) {
26:             break;
27:         }
28:     }
29:     return -EINTR;
30: }
```

kernel/syscalls/personality.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/personality.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifdef __DEBUG__
9: #include <fiwix/stdio.h>
10: #include <fiwix/process.h>
11: #endif /*__DEBUG__ */
12:
13: int sys_personality(unsigned long int persona)
14: {
15: #ifdef __DEBUG__
16:     printk("(pid %d) sys_personality(%d) -> %d\n", current->pid, persona, pe
rsona);
17: #endif /*__DEBUG__ */
18:     return persona;
19: }
```


kernel/syscalls/pipe.c

Page 1/2

```

1: /*
2:  * fiwix/kernel/syscalls/pipe.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/filesystems.h>
10: #include <fiwix/fcntl.h>
11: #include <fiwix/stat.h>
12: #include <fiwix/errno.h>
13: #include <fiwix/stdio.h>
14:
15: int sys_pipe(int pipefd[2])
16: {
17:     int rfd, rufd;
18:     int wfd, wufd;
19:     struct filesystems *fs;
20:     struct inode *i;
21:     int errno;
22:
23: #ifdef __DEBUG__
24:     printk("(pid %d) sys_pipe()", current->pid);
25: #endif /*__DEBUG__ */
26:
27:     if(!(fs = get_filesystem("pipefs"))) {
28:         printk("WARNING: %s(): pipefs filesystem is not registered!\n",
__FUNCTION__);
29:         return -EINVAL;
30:     }
31:     if((errno = check_user_area(VERIFY_WRITE, pipefd, sizeof(int) * 2)) {
32:         return errno;
33:     }
34:     if(!(i = ialloc(&fs->mt->sb, S_IFIFO))) {
35:         return -EINVAL;
36:     }
37:     if((rfd = get_new_fd(i)) < 0) {
38:         iput(i);
39:         return -ENFILE;
40:     }
41:     if((wfd = get_new_fd(i)) < 0) {
42:         release_fd(rfd);
43:         iput(i);
44:         return -ENFILE;
45:     }
46:     if((rufd = get_new_user_fd(0)) < 0) {
47:         release_fd(rfd);
48:         release_fd(wfd);
49:         iput(i);
50:         return -EMFILE;
51:     }
52:     if((wufd = get_new_user_fd(0)) < 0) {
53:         release_fd(rfd);
54:         release_fd(wfd);
55:         release_user_fd(rufd);
56:         iput(i);
57:         return -EMFILE;
58:     }
59:
60:     pipefd[0] = rufd;
61:     pipefd[1] = wufd;
62:     current->fd[rufd] = rfd;
63:     current->fd[wufd] = wfd;
64:     fd_table[rfd].flags = O_RDONLY;
65:     fd_table[wfd].flags = O_WRONLY;
66:

```

kernel/syscalls/pipe.c

Page 2/2

```
67: #ifdef __DEBUG__
68:     printk(" -> inode=%d, rufd=%d wufd=%d (rfd=%d wfd=%d)\n", i->inode, rufd
, wufd, rfd, wfd);
69: #endif /*__DEBUG__ */
70:
71:     return 0;
72: }
```

kernel/syscalls/read.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/read.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/fcntl.h>
10: #include <fiwix/errno.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #include <fiwix/process.h>
15: #endif /*__DEBUG__*/
16:
17: int sys_read(unsigned int ufd, char *buf, int count)
18: {
19:     struct inode *i;
20:     int errno;
21:
22: #ifdef __DEBUG__
23:     printk("(pid %d) sys_read(%d, 0x%08x, %d) -> ", current->pid, ufd, buf,
count);
24: #endif /*__DEBUG__*/
25:
26:     CHECK_UFD(ufd);
27:     if((errno = check_user_area(VERIFY_WRITE, buf, count))) {
28:         return errno;
29:     }
30:     if(fd_table[current->fd[ufd]].flags & O_WRONLY) {
31:         return -EBADF;
32:     }
33:     if(!count) {
34:         return 0;
35:     }
36:     if(count < 0) {
37:         return -EINVAL;
38:     }
39:
40:     i = fd_table[current->fd[ufd]].inode;
41:     if(i->fsop && i->fsop->read) {
42:         errno = i->fsop->read(i, &fd_table[current->fd[ufd]], buf, count
);
43: #ifdef __DEBUG__
44:         printk("%d\n", errno);
45: #endif /*__DEBUG__*/
46:         return errno;
47:     }
48:     return -EINVAL;
49: }

```

kernel/syscalls/readlink.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/readlink.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/stat.h>
11: #include <fiwix/errno.h>
12: #include <fiwix/string.h>
13:
14: #ifdef __DEBUG__
15: #include <fiwix/stdio.h>
16: #include <fiwix/process.h>
17: #endif /*__DEBUG__ */
18:
19: int sys_readlink(const char *filename, char *buffer, __size_t bufsize)
20: {
21:     struct inode *i;
22:     char *tmp_name;
23:     int errno;
24:
25: #ifdef __DEBUG__
26:     printk("(pid %d) sys_readlink(%s, 0x%08x, %d)\n", current->pid, filename
, (unsigned int)buffer, bufsize);
27: #endif /*__DEBUG__ */
28:
29:     if(bufsize <= 0) {
30:         return -EINVAL;
31:     }
32:     if((errno = check_user_area(VERIFY_WRITE, buffer, bufsize)) {
33:         return errno;
34:     }
35:     if((errno = malloc_name(filename, &tmp_name)) < 0) {
36:         return errno;
37:     }
38:     if((errno = namei(tmp_name, &i, NULL, !FOLLOW_LINKS)) {
39:         free_name(tmp_name);
40:         return errno;
41:     }
42:     if(!S_ISLNK(i->i_mode)) {
43:         iput(i);
44:         free_name(tmp_name);
45:         return -EINVAL;
46:     }
47:
48:     if(i->fsop && i->fsop->readlink) {
49:         errno = i->fsop->readlink(i, buffer, bufsize);
50:         iput(i);
51:         free_name(tmp_name);
52:         return errno;
53:     }
54:     iput(i);
55:     free_name(tmp_name);
56:     return -EINVAL;
57: }

```

kernel/syscalls/reboot.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/reboot.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/syscalls.h>
10: #include <fiwix/reboot.h>
11: #include <fiwix/signal.h>
12: #include <fiwix/process.h>
13: #include <fiwix/errno.h>
14:
15: #ifdef __DEBUG__
16: #include <fiwix/stdio.h>
17: #endif /*__DEBUG__*/
18:
19: int sys_reboot(int magic1, int magic2, int flag)
20: {
21: #ifdef __DEBUG__
22:     printk("(pid %d) sys_reboot(0x%08x, %d, 0x%08x)\n", current->pid, magic1
, magic2, flag);
23: #endif /*__DEBUG__*/
24:
25:     if(!IS_SUPERUSER) {
26:         return -EPERM;
27:     }
28:     if((magic1 != BMAGIC_1) || (magic2 != BMAGIC_2)) {
29:         return -EINVAL;
30:     }
31:     switch(flag) {
32:         case BMAGIC_SOFT:
33:             ctrl_alt_del = 0;
34:             break;
35:         case BMAGIC_HARD:
36:             ctrl_alt_del = 1;
37:             break;
38:         case BMAGIC_REBOOT:
39:             reboot();
40:             break;
41:         case BMAGIC_HALT:
42:             sys_kill(-1, SIGKILL);
43:             stop_kernel();
44:             break;
45:         default:
46:             return -EINVAL;
47:     }
48:     return 0;
49: }
```

kernel/syscalls/rename.c

Page 1/2

```

1: /*
2:  * fiwix/kernel/syscalls/rename.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/stat.h>
10: #include <fiwix/errno.h>
11: #include <fiwix/string.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #include <fiwix/process.h>
16: #endif /* __DEBUG__ */
17:
18: int sys_rename(const char *oldpath, const char *newpath)
19: {
20:     struct inode *i, *dir, *i_new, *dir_new;
21:     char *tmp_oldpath, *tmp_newpath;
22:     char *oldbasename, *newbasename;
23:     int errno;
24:
25: #ifdef __DEBUG__
26:     printk("(pid %d) sys_rename('%s', '%s')\n", current->pid, oldpath, newpa
th);
27: #endif /* __DEBUG__ */
28:
29:     if((errno = malloc_name(oldpath, &tmp_oldpath)) < 0) {
30:         return errno;
31:     }
32:     if((errno = namei(tmp_oldpath, &i, &dir, !FOLLOW_LINKS)) {
33:         if(dir) {
34:             iput(dir);
35:         }
36:         free_name(tmp_oldpath);
37:         return errno;
38:     }
39:     if(IS_RDONLY_FS(i)) {
40:         iput(i);
41:         iput(dir);
42:         free_name(tmp_oldpath);
43:         return -EROFS;
44:     }
45:
46:     if((errno = malloc_name(newpath, &tmp_newpath)) < 0) {
47:         iput(i);
48:         iput(dir);
49:         free_name(tmp_oldpath);
50:         return errno;
51:     }
52:     newbasename = remove_trailing_slash(tmp_newpath);
53:     if((errno = namei(newbasename, &i_new, &dir_new, !FOLLOW_LINKS)) {
54:         if(!dir_new) {
55:             iput(i);
56:             iput(dir);
57:             free_name(tmp_oldpath);
58:             free_name(tmp_newpath);
59:             return errno;
60:         }
61:     }
62:     if(dir->dev != dir_new->dev) {
63:         errno = -EXDEV;
64:         goto end;
65:     }
66:     newbasename = get_basename(newbasename);

```

kernel/syscalls/rename.c

Page 2/2

```

67:         if((newbasename[0] == '.' && newbasename[1] == NULL) || (newbasename[0]
== '.' && newbasename[1] == '.' && newbasename[2] == NULL)) {
68:             errno = -EINVAL;
69:             goto end;
70:         }
71:
72:         oldbasename = get_basename(tmp_oldpath);
73:         oldbasename = remove_trailing_slash(oldbasename);
74:         if((oldbasename[0] == '.' && oldbasename[1] == NULL) || (oldbasename[0]
== '.' && oldbasename[1] == '.' && oldbasename[2] == NULL)) {
75:             errno = -EINVAL;
76:             goto end;
77:         }
78:
79:         if(i_new) {
80:             if(S_ISREG(i->i_mode)) {
81:                 if(S_ISDIR(i_new->i_mode)) {
82:                     errno = -EISDIR;
83:                     goto end;
84:                 }
85:             }
86:             if(S_ISDIR(i->i_mode)) {
87:                 if(!S_ISDIR(i_new->i_mode)) {
88:                     errno = -ENOTDIR;
89:                     goto end;
90:                 }
91:             }
92:             if(i->inode == i_new->inode) {
93:                 errno = 0;
94:                 goto end;
95:             }
96:         }
97:
98:         if(check_permission(TO_EXEC | TO_WRITE, dir_new) < 0) {
99:             errno = -EACCES;
100:            goto end;
101:        }
102:
103:        if(dir_new->fsop && dir_new->fsop->rename) {
104:            errno = dir_new->fsop->rename(i, dir, i_new, dir_new, oldbasenam
e, newbasename);
105:        } else {
106:            errno = -EPERM;
107:        }
108:
109:    end:
110:        iput(i);
111:        iput(dir);
112:        iput(i_new);
113:        iput(dir_new);
114:        free_name(tmp_oldpath);
115:        free_name(tmp_newpath);
116:        return errno;
117:    }

```

kernel/syscalls/rmdir.c

Page 1/2

```

1: /*
2:  * fiwix/kernel/syscalls/rmdir.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/stat.h>
10: #include <fiwix/errno.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #include <fiwix/process.h>
15: #endif /*__DEBUG__ */
16:
17: int sys_rmdir(const char *dirname)
18: {
19:     struct inode *i, *dir;
20:     char *tmp_dirname;
21:     int errno;
22:
23: #ifdef __DEBUG__
24:     printk("(pid %d) sys_rmdir(%s)\n", current->pid, dirname);
25: #endif /*__DEBUG__ */
26:
27:     if((errno = malloc_name(dirname, &tmp_dirname)) < 0) {
28:         return errno;
29:     }
30:     if((errno = namei(tmp_dirname, &i, &dir, !FOLLOW_LINKS))) {
31:         if(dir) {
32:             iput(dir);
33:         }
34:         free_name(tmp_dirname);
35:         return errno;
36:     }
37:     if(!S_ISDIR(i->i_mode)) {
38:         iput(i);
39:         iput(dir);
40:         free_name(tmp_dirname);
41:         return -ENOTDIR;
42:     }
43:     if(i == current->root || i->mount_point || i->count > 1) {
44:         iput(i);
45:         iput(dir);
46:         free_name(tmp_dirname);
47:         return -EBUSY;
48:     }
49:     if(IS_RDONLY_FS(i)) {
50:         iput(i);
51:         iput(dir);
52:         free_name(tmp_dirname);
53:         return -EROFS;
54:     }
55:     if(i == dir) {
56:         iput(i);
57:         iput(dir);
58:         free_name(tmp_dirname);
59:         return -EPERM;
60:     }
61:     if(check_permission(TO_EXEC | TO_WRITE, dir) < 0) {
62:         iput(i);
63:         iput(dir);
64:         free_name(tmp_dirname);
65:         return -EACCES;
66:     }
67:

```


kernel/syscalls/rmdir.c

Page 2/2

```
68:      /* check sticky permission bit */
69:      if(dir->i_mode & S_ISVTX) {
70:          if(check_user_permission(i) {
71:              iput(i);
72:              iput(dir);
73:              free_name(tmp_dirname);
74:              return -EPERM;
75:          }
76:      }
77:
78:      if(i->fsop && i->fsop->rmdir) {
79:          errno = i->fsop->rmdir(dir, i);
80:      } else {
81:          errno = -EPERM;
82:      }
83:      iput(i);
84:      iput(dir);
85:      free_name(tmp_dirname);
86:      return errno;
87: }
```

kernel/syscalls/select.c

Page 1/3

```

1: /*
2:  * fiwix/kernel/syscalls/select.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/process.h>
11: #include <fiwix/timer.h>
12: #include <fiwix/sched.h>
13: #include <fiwix/sleep.h>
14: #include <fiwix/errno.h>
15: #include <fiwix/stdio.h>
16: #include <fiwix/string.h>
17:
18: static int check_fds(int nfd, fd_set *rfd, fd_set *wfd, fd_set *efd)
19: {
20:     int n, bit;
21:     unsigned long int set;
22:
23:     n = bit = 0;
24:     while(bit < nfd) {
25:         bit = n * __NFDBITS;
26:         set = rfd->fds_bits[n] | wfd->fds_bits[n] | efd->fds_bits[n];
27:         while(set) {
28:             if(__FD_ISSET(bit, rfd)) {
29:                 CHECK_UFD(bit);
30:             }
31:             set >>= 1;
32:             bit++;
33:         }
34:         n++;
35:     }
36:
37:     return 0;
38: }
39:
40: static int do_check(struct inode *i, int flag)
41: {
42:     if(i->fsop && i->fsop->select) {
43:         if(i->fsop->select(i, flag)) {
44:             return 1;
45:         }
46:     }
47:
48:     return 0;
49: }
50:
51: int do_select(int nfd, fd_set *rfd, fd_set *wfd, fd_set *efd, fd_set *res_rfd, fd_set *res_wfd, fd_set *res_efd)
52: {
53:     int n, count;
54:     struct inode *i;
55:
56:     count = 0;
57:     for(;;) {
58:         for(n = 0; n < nfd; n++) {
59:             if(!current->fd[n]) {
60:                 continue;
61:             }
62:             i = fd_table[current->fd[n]].inode;
63:             if(__FD_ISSET(n, rfd)) {
64:                 if(do_check(i, SEL_R)) {
65:                     __FD_SET(n, res_rfd);
66:                     count++;

```

kernel/syscalls/select.c

Page 2/3

```

67:         }
68:     }
69:     if(__FD_ISSET(n, wfds)) {
70:         if(do_check(i, SEL_W)) {
71:             __FD_SET(n, res_wfds);
72:             count++;
73:         }
74:     }
75:     if(__FD_ISSET(n, efds)) {
76:         if(do_check(i, SEL_E)) {
77:             __FD_SET(n, res_efds);
78:             count++;
79:         }
80:     }
81: }
82:
83:     if(count || !current->timeout || current->sigpending & ~current->
>sigblocked) {
84:         break;
85:     }
86:     sleep(&do_select, PROC_INTERRUPTIBLE);
87: }
88:
89:     return count;
90: }
91:
92: int sys_select(int nfd, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, s
truct timeval *timeout)
93: {
94:     unsigned long int t;
95:     fd_set rfds, wfds, efds;
96:     fd_set res_rfds, res_wfds, res_efds;
97:     int errno;
98:
99: #ifdef __DEBUG__
100:     printk("(pid %d) sys_select(%d, 0x%08x, 0x%08x, 0x%08x, 0x%08x [%d])\n",
current->pid, nfd, (int)readfds, (int)writefds, (int)exceptfds, (int)timeout, (int)ti
meout ? tv2ticks(timeout): 0);
101: #endif /* __DEBUG__ */
102:
103:     if(nfd < 0) {
104:         return -EINVAL;
105:     }
106:     if(nfd > MIN(__FD_SETSIZE, NR_OPENS)) {
107:         nfd = MIN(__FD_SETSIZE, NR_OPENS);
108:     }
109:
110:     if(readfds) {
111:         if((errno = check_user_area(VERIFY_WRITE, readfds, sizeof(fd_set
))) {
112:             return errno;
113:         }
114:         memcpy_b(&rfds, readfds, sizeof(fd_set));
115:     } else {
116:         __FD_ZERO(&rfds);
117:     }
118:     if(writefds) {
119:         if((errno = check_user_area(VERIFY_WRITE, writefds, sizeof(fd_se
t)))) {
120:             return errno;
121:         }
122:         memcpy_b(&wfds, writefds, sizeof(fd_set));
123:     } else {
124:         __FD_ZERO(&wfds);
125:     }
126:     if(exceptfds) {
127:         if((errno = check_user_area(VERIFY_WRITE, exceptfds, sizeof(fd_s

```

```
et))) {
128:         return errno;
129:     }
130:     memcpy_b(&efds, exceptfds, sizeof(fd_set));
131: } else {
132:     __FD_ZERO(&efds);
133: }
134:
135: /* check the validity of all fds */
136: if((errno = check_fds(nfds, &rfds, &wfds, &efds)) < 0) {
137:     return errno;
138: }
139:
140: if(timeout) {
141:     t = tv2ticks(timeout);
142: } else {
143:     t = INFINITE_WAIT;
144: }
145:
146: __FD_ZERO(&res_rfds);
147: __FD_ZERO(&res_wfds);
148: __FD_ZERO(&res_efds);
149:
150: current->timeout = t;
151: if((errno = do_select(nfds, &rfds, &wfds, &efds, &res_rfds, &res_wfds, &
res_efds)) < 0) {
152:     return errno;
153: }
154: current->timeout = 0;
155:
156: if(readfds) {
157:     memcpy_b(readfds, &res_rfds, sizeof(fd_set));
158: }
159: if(writefds) {
160:     memcpy_b(writefds, &res_wfds, sizeof(fd_set));
161: }
162: if(exceptfds) {
163:     memcpy_b(exceptfds, &res_efds, sizeof(fd_set));
164: }
165: return errno;
166: }
```

kernel/syscalls/setdomainname.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/setdomainname.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/utsname.h>
10: #include <fiwix/errno.h>
11: #include <fiwix/string.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #include <fiwix/process.h>
16: #endif /* __DEBUG__ */
17:
18: int sys_setdomainname(const char *name, int length)
19: {
20:     int errno;
21:
22: #ifdef __DEBUG__
23:     printk("(pid %d) sys_setdomainname('%s', %d)\n", current->pid, name, len
gth);
24: #endif /* __DEBUG__ */
25:
26:     if((errno = check_user_area(VERIFY_READ, name, length)) {
27:         return errno;
28:     }
29:     if(!IS_SUPERUSER) {
30:         return -EPERM;
31:     }
32:     if(length < 0 || length > _UTSNAME_LENGTH) {
33:         return -EINVAL;
34:     }
35:     memcpy_b(&sys_utsname.domainname, name, length);
36:     sys_utsname.domainname[length] = NULL;
37:     return 0;
38: }
```

kernel/syscalls/setfsgid.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/setfsgid.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9:
10: #ifdef __DEBUG__
11: #include <fiwix/stdio.h>
12: #include <fiwix/process.h>
13: #endif /*__DEBUG__ */
14:
15: int sys_setfsgid(__gid_t fsgid)
16: {
17: #ifdef __DEBUG__
18:     printk("(pid %d) sys_setfsgid(%d) -> %d\n", current->pid, fsgid);
19: #endif /*__DEBUG__ */
20:     return 0;
21: }
```

kernel/syscalls/setfsuid.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/setfsuid.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9:
10: #ifdef __DEBUG__
11: #include <fiwix/stdio.h>
12: #include <fiwix/process.h>
13: #endif /*__DEBUG__*/
14:
15: int sys_setfsuid(__uid_t fsuid)
16: {
17: #ifdef __DEBUG__
18:     printk("(pid %d) sys_setfsuid(%d) -> %d\n", current->pid, fsuid);
19: #endif /*__DEBUG__*/
20:     return 0;
21: }
```

kernel/syscalls/setgid.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/setgid.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/process.h>
10: #include <fiwix/errno.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #endif /*__DEBUG__*/
15:
16: int sys_setgid(__gid_t gid)
17: {
18: #ifdef __DEBUG__
19:     printk("(pid %d) sys_setgid(%d)\n", current->pid, gid);
20: #endif /*__DEBUG__*/
21:
22:     if(IS_SUPERUSER) {
23:         current->gid = current->egid = current->sgid = gid;
24:     } else {
25:         if((current->gid == gid) || (current->sgid == gid)) {
26:             current->egid = gid;
27:         } else {
28:             return -EPERM;
29:         }
30:     }
31:     return 0;
32: }
```


kernel/syscalls/setgroups.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/setgroups.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/process.h>
11: #include <fiwix/errno.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #endif /*__DEBUG__*/
16:
17: int sys_setgroups(__ssize_t size, const __gid_t *list)
18: {
19:     int n, errno;
20:
21: #ifdef __DEBUG__
22:     printk("(pid %d) sys_setgroups(%d, 0x%08x)\n", current->pid, size, (unsi
gned int)list);
23: #endif /*__DEBUG__*/
24:
25:     if((errno = check_user_area(VERIFY_READ, list, sizeof(__gid_t))) {
26:         return errno;
27:     }
28:     if(!IS_SUPERUSER) {
29:         return -EPERM;
30:     }
31:     if(size > NGROUPS_MAX) {
32:         return -EINVAL;
33:     }
34:     for(n = 0; n < NGROUPS_MAX; n++) {
35:         current->groups[n] = -1;
36:         if(n < size) {
37:             current->groups[n] = list[n];
38:         }
39:     }
40:     return 0;
41: }
```

kernel/syscalls/sethostname.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/sethostname.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/utsname.h>
10: #include <fiwix/process.h>
11: #include <fiwix/errno.h>
12: #include <fiwix/string.h>
13:
14: #ifdef __DEBUG__
15: #include <fiwix/stdio.h>
16: #endif /* __DEBUG__ */
17:
18: int sys_sethostname(const char *name, int length)
19: {
20:     int errno;
21:     char *tmp_name;
22:
23: #ifdef __DEBUG__
24:     printk("(pid %d) sys_sethostname('%s', %d)\n", current->pid, name, length);
25: #endif /* __DEBUG__ */
26:
27:     if((errno = malloc_name(name, &tmp_name)) < 0) {
28:         return errno;
29:     }
30:     if(!IS_SUPERUSER) {
31:         free_name(tmp_name);
32:         return -EPERM;
33:     }
34:     if(length < 0 || length > _UTSNAME_LENGTH) {
35:         free_name(tmp_name);
36:         return -EINVAL;
37:     }
38:     memcpy_b(&sys_utsname.nodename, tmp_name, length);
39:     sys_utsname.nodename[length] = NULL;
40:     free_name(tmp_name);
41:     return 0;
42: }
```

kernel/syscalls/setitimer.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/setitimer.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/time.h>
10:
11: #ifdef __DEBUG__
12: #include <fiwix/stdio.h>
13: #include <fiwix/process.h>
14: #endif /*__DEBUG__ */
15:
16: int sys_setitimer(int which, const struct itimerval *new_value, struct itimerval
*old_value)
17: {
18:     int errno;
19:
20: #ifdef __DEBUG__
21:     printk("(pid %d) sys_setitimer(%d, 0x%08x, 0x%08x) -> \n", current->pid,
which, (unsigned int)new_value, (unsigned int)old_value);
22: #endif /*__DEBUG__ */
23:
24:     if((unsigned int)old_value) {
25:         if((errno = check_user_area(VERIFY_WRITE, old_value, sizeof(stru
ct itimerval)))) {
26:             return errno;
27:         }
28:     }
29:
30:     return setitimer(which, new_value, old_value);
31: }
```

kernel/syscalls/setpgid.c

Page 1/2

```

1: /*
2:  * fiwix/kernel/syscalls/setpgid.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/process.h>
10: #include <fiwix/sched.h>
11: #include <fiwix/errno.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #endif /*__DEBUG__ */
16:
17: int sys_setpgid(__pid_t pid, __pid_t pgid)
18: {
19:     struct proc *p;
20:
21: #ifdef __DEBUG__
22:     printk("(pid %d) sys_setpgid(%d, %d)", current->pid, pid, pgid);
23: #endif /*__DEBUG__ */
24:
25:     if(pid < 0 || pgid < 0) {
26:         return -EINVAL;
27:     }
28:     if(!pid) {
29:         pid = current->pid;
30:     }
31:
32:     if(!(p = get_proc_by_pid(pid))) {
33:         return -EINVAL;
34:     }
35:     if(!pgid) {
36:         pgid = p->pid;
37:     }
38:
39:     if(p != current && p->ppid != current->pid) {
40:         return -ESRCH;
41:     }
42:     if(p->sid == p->pid || p->sid != current->sid) {
43:         return -EPERM;
44:     }
45:
46:     {
47:         struct proc *p;
48:
49:         FOR_EACH_PROCESS(p) {
50:             if(p->pgid == pgid && p->sid != current->sid) {
51:                 return -EPERM;
52:             }
53:             p = p->next;
54:         }
55:     }
56:
57:     if(p != current && p->flags & PF_PEXEC) {
58:         return -EACCES;
59:     }
60:
61:     p->pgid = pgid;
62:
63: #ifdef __DEBUG__
64:     printk(" -> 0\n");
65: #endif /*__DEBUG__ */
66:
67:     return 0;

```

```
68: }
```

kernel/syscalls/setregid.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/setregid.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/process.h>
10: #include <fiwix/errno.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #endif /*__DEBUG__*/
15:
16: int sys_setregid(__gid_t gid, __gid_t egid)
17: {
18: #ifdef __DEBUG__
19:     printk("(pid %d) sys_setregid(%d, %d) -> ", current->pid, gid, egid);
20: #endif /*__DEBUG__*/
21:
22:     if(IS_SUPERUSER) {
23:         if(egid != (__uid_t)-1) {
24:             if(gid != (__uid_t)-1 || (current->egid >= 0 && current-
>gid != egid)) {
25:                 current->sgid = egid;
26:             }
27:             current->egid = egid;
28:         }
29:         if(gid != (__uid_t)-1) {
30:             current->gid = gid;
31:         }
32:     } else {
33:         if(egid != (__uid_t)-1 && (current->gid == egid || current->egid
== egid || current->sgid == egid)) {
34:             if(gid != (__uid_t)-1 || (current->egid >= 0 && current-
>gid != egid)) {
35:                 current->sgid = egid;
36:             }
37:             current->egid = egid;
38:         } else {
39:             return -EPERM;
40:         }
41:         if(gid != (__uid_t)-1 && (current->gid == gid || current->egid =
= gid)) {
42:             current->gid = gid;
43:         } else {
44:             return -EPERM;
45:         }
46:     }
47:
48:     return 0;
49: }

```

kernel/syscalls/setreuid.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/setreuid.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/process.h>
10: #include <fiwix/errno.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #endif /*__DEBUG__ */
15:
16: int sys_setreuid(__uid_t uid, __uid_t euid)
17: {
18: #ifdef __DEBUG__
19:     printk("(pid %d) sys_setreuid(%d, %d) -> ", current->pid, uid, euid);
20: #endif /*__DEBUG__ */
21:
22:     if (IS_SUPERUSER) {
23:         if (euid != (__uid_t)-1) {
24:             if (uid != (__uid_t)-1 || (current->euid >= 0 && current-
>uid != euid)) {
25:                 current->suid = euid;
26:             }
27:             current->euid = euid;
28:         }
29:         if (uid != (__uid_t)-1) {
30:             current->uid = uid;
31:         }
32:     } else {
33:         if (euid != (__uid_t)-1 && (current->uid == euid || current->euid
== euid || current->suid == euid)) {
34:             if (uid != (__uid_t)-1 || (current->euid >= 0 && current-
>uid != euid)) {
35:                 current->suid = euid;
36:             }
37:             current->euid = euid;
38:         } else {
39:             return -EPERM;
40:         }
41:         if (uid != (__uid_t)-1 && (current->uid == uid || current->euid =
= uid)) {
42:             current->uid = uid;
43:         } else {
44:             return -EPERM;
45:         }
46:     }
47:
48: #ifdef __DEBUG__
49:     printk(" 0\n");
50: #endif /*__DEBUG__ */
51:
52:     return 0;
53: }

```

kernel/syscalls/setrlimit.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/setrlimit.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/resource.h>
10: #include <fiwix/process.h>
11: #include <fiwix/errno.h>
12: #include <fiwix/string.h>
13:
14: #ifdef __DEBUG__
15: #include <fiwix/stdio.h>
16: #endif /* __DEBUG__ */
17:
18: int sys_setrlimit(int resource, const struct rlimit *rlim)
19: {
20:     int errno;
21:
22: #ifdef __DEBUG__
23:     printk("(pid %d) sys_setrlimit(%d, 0x%08x)\n", current->pid, resource, (
unsigned int)rlim);
24: #endif /* __DEBUG__ */
25:
26:     if((errno = check_user_area(VERIFY_READ, rlim, sizeof(struct rlimit))))
{
27:         return errno;
28:     }
29:     if(resource < 0 || resource >= RLIM_NLIMITS) {
30:         return -EINVAL;
31:     }
32:     if(rlim->rlim_cur > rlim->rlim_max) {
33:         return -EINVAL;
34:     }
35:     if(!IS_SUPERUSER) {
36:         if(rlim->rlim_max > current->rlim[resource].rlim_max) {
37:             return -EPERM;
38:         }
39:     }
40:     memcpy_b(&current->rlim[resource], rlim, sizeof(struct rlimit));
41:     return 0;
42: }

```


kernel/syscalls/setsid.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/setsid.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/process.h>
9: #include <fiwix/errno.h>
10: #include <fiwix/string.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #endif /*__DEBUG__*/
15:
16: int sys_setsid(void)
17: {
18:     struct proc *p;
19:
20: #ifdef __DEBUG__
21:     printk("(pid %d) sys_setsid()\n", current->pid);
22: #endif /*__DEBUG__*/
23:
24:     if(PG_LEADER(current)) {
25:         return -EPERM;
26:     }
27:     FOR_EACH_PROCESS(p) { /* POSIX ANSI/IEEE Std 1003.1-1996 4.3.2 */
28:         if(p != current && p->pgid == current->pid) {
29:             return -EPERM;
30:         }
31:         p = p->next;
32:     }
33:
34:     current->sid = current->pgid = current->pid;
35:     current->ctty = NULL;
36:     return current->sid;
37: }
```

kernel/syscalls/settimeofday.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/settimeofday.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/time.h>
11: #include <fiwix/timer.h>
12: #include <fiwix/process.h>
13: #include <fiwix/errno.h>
14:
15: #ifdef __DEBUG__
16: #include <fiwix/stdio.h>
17: #endif /*__DEBUG__ */
18:
19: int sys_settimeofday(const struct timeval *tv, const struct timezone *tz)
20: {
21:     int errno;
22:
23: #ifdef __DEBUG__
24:     printk("(pid %d) sys_settimeofday()\n", current->pid);
25: #endif /*__DEBUG__ */
26:
27:     if(!IS_SUPERUSER) {
28:         return -EPERM;
29:     }
30:
31:     if(tv) {
32:         if((errno = check_user_area(VERIFY_READ, tv, sizeof(struct timev
al)))) {
33:             return errno;
34:         }
35:         CURRENT_TIME = tv->tv_sec;
36:         set_system_time(CURRENT_TIME);
37:     }
38:     if(tz) {
39:         if((errno = check_user_area(VERIFY_READ, tz, sizeof(struct timez
one)))) {
40:             return errno;
41:         }
42:         kstat.tz_minuteswest = tz->tz_minuteswest;
43:         kstat.tz_dsttime = tz->tz_dsttime;
44:     }
45:     return 0;
46: }

```

kernel/syscalls/setuid.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/setuid.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/process.h>
10: #include <fiwix/errno.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #endif /*__DEBUG__*/
15:
16: int sys_setuid(__uid_t uid)
17: {
18: #ifdef __DEBUG__
19:     printk("(pid %d) sys_setuid(%d)\n", current->pid, uid);
20: #endif /*__DEBUG__*/
21:
22:     if(IS_SUPERUSER) {
23:         current->uid = current->suid = uid;
24:     } else {
25:         if((current->uid != uid) && (current->suid != uid)) {
26:             return -EPERM;
27:         }
28:     }
29:     current->euid = uid;
30:     return 0;
31: }
```

kernel/syscalls/sgetmask.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/sgetmask.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/process.h>
9:
10: #ifdef __DEBUG__
11: #include <fiwix/stdio.h>
12: #endif /*__DEBUG__*/
13:
14: int sys_sgetmask(void)
15: {
16: #ifdef __DEBUG__
17:     printk("(pid %d) sys_sgetmask() -> \n", current->pid);
18: #endif /*__DEBUG__*/
19:     return current->sigblocked;
20: }
```

kernel/syscalls/sigation.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/sigation.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/signal.h>
10: #include <fiwix/errno.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #include <fiwix/process.h>
15: #endif /*__DEBUG__ */
16:
17: int sys_sigation(__sigset_t signum, const struct sigaction *newaction, struct sigaction *oldaction)
18: {
19:     int errno;
20:
21: #ifdef __DEBUG__
22:     printk("(pid %d) sys_sigation(%d, 0x%08x, 0x%08x)\n", current->pid, signum, (unsigned int)newaction, (unsigned int)oldaction);
23: #endif /*__DEBUG__ */
24:
25:     if(signum < 1 || signum > NSIG) {
26:         return -EINVAL;
27:     }
28:     if(signum == SIGKILL || signum == SIGSTOP) {
29:         return -EINVAL;
30:     }
31:     if(oldaction) {
32:         if((errno = check_user_area(VERIFY_WRITE, oldaction, sizeof(struct sigaction)))) {
33:             return errno;
34:         }
35:         *oldaction = current->sigaction[signum - 1];
36:     }
37:     if(newaction) {
38:         if((errno = check_user_area(VERIFY_READ, newaction, sizeof(struct sigaction)))) {
39:             return errno;
40:         }
41:         current->sigaction[signum - 1] = *newaction;
42:         if(current->sigaction[signum - 1].sa_handler == SIG_IGN) {
43:             if(signum != SIGCHLD) {
44:                 current->sigpending &= SIG_MASK(signum);
45:             }
46:         }
47:         if(current->sigaction[signum - 1].sa_handler == SIG_DFL) {
48:             if(signum != SIGCHLD) {
49:                 current->sigpending &= SIG_MASK(signum);
50:             }
51:         }
52:     }
53:     return 0;
54: }

```

kernel/syscalls/signal.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/signal.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/syscalls.h>
9: #include <fiwix/signal.h>
10: #include <fiwix/process.h>
11: #include <fiwix/errno.h>
12: #include <fiwix/string.h>
13:
14: #ifdef __DEBUG__
15: #include <fiwix/stdio.h>
16: #endif /* __DEBUG__ */
17:
18: unsigned int sys_signal(__sigset_t signum, void(* sighandler)(int))
19: {
20:     struct sigaction s;
21:     int errno;
22:
23: #ifdef __DEBUG__
24:     printk("(pid %d) sys_signal()\n", current->pid);
25: #endif /* __DEBUG__ */
26:
27:     if(signum < 1 || signum > NSIG) {
28:         return -EINVAL;
29:     }
30:     if(signum == SIGKILL || signum == SIGSTOP) {
31:         return -EINVAL;
32:     }
33:     if(sighandler != SIG_DFL && sighandler != SIG_IGN) {
34:         if((errno = check_user_area(VERIFY_READ, sighandler, sizeof(unsig
gned int)))) {
35:             return errno;
36:         }
37:     }
38:
39:     memset_b(&s, 0, sizeof(struct sigaction));
40:     s.sa_handler = sighandler;
41:     s.sa_mask = 0;
42:     s.sa_flags = SA_RESETHAND;
43:     sighandler = current->sigaction[signum - 1].sa_handler;
44:     current->sigaction[signum - 1] = s;
45:     if(current->sigaction[signum - 1].sa_handler == SIG_IGN) {
46:         if(signum != SIGCHLD) {
47:             current->sigpending &= SIG_MASK(signum);
48:         }
49:     }
50:     if(current->sigaction[signum - 1].sa_handler == SIG_DFL) {
51:         if(signum != SIGCHLD) {
52:             current->sigpending &= SIG_MASK(signum);
53:         }
54:     }
55:     return (unsigned int)sighandler;
56: }

```

kernel/syscalls/sigpending.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/sigpending.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/signal.h>
10: #include <fiwix/process.h>
11: #include <fiwix/string.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #endif /*__DEBUG__*/
16:
17: int sys_sigpending(__sigset_t *set)
18: {
19:     int errno;
20:
21: #ifdef __DEBUG__
22:     printk("(pid %d) sys_sigpending(0x%08x) -> ", current->pid, set);
23: #endif /*__DEBUG__*/
24:
25:     if((errno = check_user_area(VERIFY_WRITE, set, sizeof(__sigset_t)))) {
26:         return errno;
27:     }
28:     memcpy_b(set, &current->sigpending, sizeof(__sigset_t));
29:     return 0;
30: }
```

kernel/syscalls/sigprocmask.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/sigprocmask.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/signal.h>
10: #include <fiwix/process.h>
11: #include <fiwix/errno.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #endif /*__DEBUG__*/
16:
17: int sys_sigprocmask(int how, const __sigset_t *set, __sigset_t *oldset)
18: {
19:     int errno;
20:
21: #ifdef __DEBUG__
22:     printk("(pid %d) sys_sigprocmask(%d, 0x%08x, 0x%08x)\n", current->pid, h
ow, set, oldset);
23: #endif /*__DEBUG__*/
24:
25:     if(oldset) {
26:         if((errno = check_user_area(VERIFY_WRITE, oldset, sizeof(__sigse
t_t)))) {
27:             return errno;
28:         }
29:         *oldset = current->sigblocked;
30:     }
31:
32:     if(set) {
33:         if((errno = check_user_area(VERIFY_READ, set, sizeof(__sigset_t)
))) {
34:             return errno;
35:         }
36:         switch(how) {
37:             case SIG_BLOCK:
38:                 current->sigblocked |= (*set & SIG_BLOCKABLE);
39:                 break;
40:             case SIG_UNBLOCK:
41:                 current->sigblocked &= ~(*set & SIG_BLOCKABLE);
42:                 break;
43:             case SIG_SETMASK:
44:                 current->sigblocked = (*set & SIG_BLOCKABLE);
45:                 break;
46:             default:
47:                 return -EINVAL;
48:         }
49:     }
50:     return 0;
51: }

```


kernel/syscalls/sigreturn.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/sigreturn.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/process.h>
9: #include <fiwix/sigcontext.h>
10: #include <fiwix/string.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #endif /*__DEBUG__ */
15:
16: int sys_sigreturn(unsigned int signum, int arg2, int arg3, int arg4, int arg5, s
truct sigcontext *sc)
17: {
18: #ifdef __DEBUG__
19:     printk("(pid %d) sys_sigreturn(0x%08x)\n", current->pid, signum);
20: #endif /*__DEBUG__ */
21:
22:     current->sigblocked &= ~current->sigexecuting;
23:     current->sigexecuting = 0;
24:     memcpy_b(sc, &current->sc[signum - 1], sizeof(struct sigcontext));
25:
26:     /*
27:      * We return here the value that the syscall was returning when it was
28:      * interrupted by a signal.
29:      */
30:     return current->sc[signum - 1].eax;
31: }
```

kernel/syscalls/sigsuspend.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/sigsuspend.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/syscalls.h>
10: #include <fiwix/signal.h>
11: #include <fiwix/process.h>
12: #include <fiwix/errno.h>
13:
14: #ifdef __DEBUG__
15: #include <fiwix/stdio.h>
16: #endif /* __DEBUG__ */
17:
18: int sys_sigsuspend(__sigset_t *mask)
19: {
20:     __sigset_t old_mask;
21:     int errno;
22:
23: #ifdef __DEBUG__
24:     printk("(pid %d) sys_sigsuspend(0x%08x) -> ", current->pid, mask);
25: #endif /* __DEBUG__ */
26:
27:     old_mask = current->sigblocked;
28:     if(mask) {
29:         if((errno = check_user_area(VERIFY_READ, mask, sizeof(__sigset_t
)))) {
30:             return errno;
31:         }
32:         current->sigblocked = (int)*mask & SIG_BLOCKABLE;
33:     } else {
34:         current->sigblocked = 0 & SIG_BLOCKABLE;
35:     }
36:     sys_pause();
37:     current->sigblocked = old_mask;
38:
39: #ifdef __DEBUG__
40:     printk("-EINTR\n");
41: #endif /* __DEBUG__ */
42:
43:     return -EINTR;
44: }
```

kernel/syscalls/socketcall.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/socketcall.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/errno.h>
9:
10: #ifdef __DEBUG__
11: #include <fiwix/stdio.h>
12: #include <fiwix/process.h>
13: #endif /*__DEBUG__ */
14:
15: int sys_socketcall(int call, unsigned long int *args)
16: {
17: #ifdef __DEBUG__
18:     printk("(pid %d) sys_socketcall(%d, 0x%08x) -> ENOENT\n", current->pid,
call, args);
19: #endif /*__DEBUG__ */
20:
21:     /* FIXME: to be implemented */
22:
23:     return -ENOENT;
24: }
```

kernel/syscalls/ssetmask.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/ssetmask.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/process.h>
9: #include <fiwix/signal.h>
10:
11: #ifdef __DEBUG__
12: #include <fiwix/stdio.h>
13: #endif /*__DEBUG__*/
14:
15: int sys_ssetmask(int newmask)
16: {
17:     int oldmask;
18:
19: #ifdef __DEBUG__
20:     printk("(pid %d) sys_ssetmask(0x%08x) -> \n", current->pid, newmask);
21: #endif /*__DEBUG__*/
22:
23:     oldmask = current->sigblocked;
24:     current->sigblocked = newmask & SIG_BLOCKABLE;
25:     return oldmask;
26: }
```

kernel/syscalls/stat.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/stat.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/statbuf.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/string.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #include <fiwix/process.h>
16: #endif /* __DEBUG__ */
17:
18: int sys_stat(const char *filename, struct old_stat *statbuf)
19: {
20:     struct inode *i;
21:     char *tmp_name;
22:     int errno;
23:
24: #ifdef __DEBUG__
25:     printk("(pid %d) sys_stat(%s, 0x%08x) -> returning structure\n", current
->pid, filename, (unsigned int) statbuf);
26: #endif /* __DEBUG__ */
27:
28:     if((errno = check_user_area(VERIFY_WRITE, statbuf, sizeof(struct old_sta
t)))) {
29:         return errno;
30:     }
31:     if((errno = malloc_name(filename, &tmp_name)) < 0) {
32:         return errno;
33:     }
34:     if((errno = namei(tmp_name, &i, NULL, FOLLOW_LINKS))) {
35:         free_name(tmp_name);
36:         return errno;
37:     }
38:     statbuf->st_dev = i->dev;
39:     statbuf->st_ino = i->inode;
40:     statbuf->st_mode = i->i_mode;
41:     statbuf->st_nlink = i->i_nlink;
42:     statbuf->st_uid = i->i_uid;
43:     statbuf->st_gid = i->i_gid;
44:     statbuf->st_rdev = i->rdev;
45:     statbuf->st_size = i->i_size;
46:     statbuf->st_atime = i->i_atime;
47:     statbuf->st_mtime = i->i_mtime;
48:     statbuf->st_ctime = i->i_ctime;
49:     iput(i);
50:     free_name(tmp_name);
51:     return 0;
52: }

```

kernel/syscalls/statfs.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/statfs.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/statfs.h>
10: #include <fiwix/errno.h>
11: #include <fiwix/string.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #include <fiwix/process.h>
16: #endif /* __DEBUG__ */
17:
18: int sys_statfs(const char *filename, struct statfs *statfsbuf)
19: {
20:     struct inode *i;
21:     char *tmp_name;
22:     int errno;
23:
24: #ifdef __DEBUG__
25:     printk("(pid %d) sys_statfs('%s', 0x%08x)\n", current->pid, filename, (u
nsigned int) statfsbuf);
26: #endif /* __DEBUG__ */
27:
28:     if((errno = check_user_area(VERIFY_WRITE, statfsbuf, sizeof(struct statf
s)))) {
29:         return errno;
30:     }
31:     if((errno = malloc_name(filename, &tmp_name)) < 0) {
32:         return errno;
33:     }
34:     if((errno = namei(tmp_name, &i, NULL, FOLLOW_LINKS))) {
35:         free_name(tmp_name);
36:         return errno;
37:     }
38:     if(i->sb && i->sb->fsop && i->sb->fsop->statfs) {
39:         i->sb->fsop->statfs(i->sb, statfsbuf);
40:         iput(i);
41:         free_name(tmp_name);
42:         return 0;
43:     }
44:     iput(i);
45:     free_name(tmp_name);
46:     return -ENOSYS;
47: }

```

kernel/syscalls/stime.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/stime.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/timer.h>
11: #include <fiwix/errno.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #include <fiwix/process.h>
16: #endif /* __DEBUG__ */
17:
18: int sys_stime(__time_t *t)
19: {
20:     int errno;
21:
22: #ifdef __DEBUG__
23:     printk("(pid %d) sys_stime(0x%08x)\n", current->pid, (unsigned int)t);
24: #endif /* __DEBUG__ */
25:
26:     if(!IS_SUPERUSER) {
27:         return -EPERM;
28:     }
29:     if((errno = check_user_area(VERIFY_READ, t, sizeof(__time_t)))) {
30:         return errno;
31:     }
32:
33:     set_system_time(*t);
34:     return 0;
35: }
```

kernel/syscalls/symlink.c

Page 1/2

```

1: /*
2:  * fiwix/kernel/syscalls/symlink.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/stat.h>
10: #include <fiwix/errno.h>
11: #include <fiwix/string.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #include <fiwix/process.h>
16: #endif /* __DEBUG__ */
17:
18: int sys_symlink(const char *oldpath, const char *newpath)
19: {
20:     struct inode *i, *dir;
21:     char *tmp_oldpath, *tmp_newpath, *basename;
22:     int errno;
23:
24: #ifdef __DEBUG__
25:     printk("(pid %d) sys_symlink('%s', '%s')\n", current->pid, oldpath, newp
ath);
26: #endif /* __DEBUG__ */
27:
28:     if((errno = malloc_name(oldpath, &tmp_oldpath)) < 0) {
29:         return errno;
30:     }
31:     if((errno = malloc_name(newpath, &tmp_newpath)) < 0) {
32:         free_name(tmp_oldpath);
33:         return errno;
34:     }
35:     basename = get_basename(tmp_newpath);
36:     if((errno = namei(tmp_newpath, &i, &dir, !FOLLOW_LINKS))) {
37:         if(!dir) {
38:             free_name(tmp_oldpath);
39:             free_name(tmp_newpath);
40:             return errno;
41:         }
42:     }
43:     if(!errno) {
44:         iput(i);
45:         iput(dir);
46:         free_name(tmp_oldpath);
47:         free_name(tmp_newpath);
48:         return -EEXIST;
49:     }
50:     if(IS_RDONLY_FS(dir)) {
51:         iput(dir);
52:         free_name(tmp_oldpath);
53:         free_name(tmp_newpath);
54:         return -EROFS;
55:     }
56:
57:     if(check_permission(TO_EXEC | TO_WRITE, dir) < 0) {
58:         iput(dir);
59:         free_name(tmp_oldpath);
60:         free_name(tmp_newpath);
61:         return -EACCES;
62:     }
63:
64:     if(dir->fsop && dir->fsop->symlink) {
65:         errno = dir->fsop->symlink(dir, basename, tmp_oldpath);
66:     } else {

```


kernel/syscalls/symlink.c

Page 2/2

```
67:             errno = -EPERM;
68:         }
69:         iput(dir);
70:         free_name(tmp_oldpath);
71:         free_name(tmp_newpath);
72:         return errno;
73: }
```

kernel/syscalls/sync.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/sync.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/buffer.h>
10: #include <fiwix/filesystems.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #include <fiwix/process.h>
15: #endif /* __DEBUG__ */
16:
17: void sys_sync(void)
18: {
19: #ifdef __DEBUG__
20:     printk("(pid %d) sys_sync()\n", current->pid);
21: #endif /* __DEBUG__ */
22:
23:     sync_superblocks(0);    /* in all devices */
24:     sync_inodes(0);        /* in all devices */
25:     sync_buffers(0);       /* in all devices */
26:     return;
27: }
```

kernel/syscalls/sysinfo.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/sysinfo.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/system.h>
11: #include <fiwix/sched.h>
12: #include <fiwix/mm.h>
13: #include <fiwix/string.h>
14:
15: #ifdef __DEBUG__
16: #include <fiwix/stdio.h>
17: #include <fiwix/process.h>
18: #endif /*__DEBUG__ */
19:
20: int sys_sysinfo(struct sysinfo *info)
21: {
22:     struct sysinfo tmp_info;
23:     struct proc *p;
24:     int errno;
25:
26: #ifdef __DEBUG__
27:     printk("(pid %d) sys_sysinfo(0x%08x)\n ", current->pid, (unsigned int)in
fo);
28: #endif /*__DEBUG__ */
29:
30:     if((errno = check_user_area(VERIFY_WRITE, info, sizeof(struct sysinfo)))
) {
31:         return errno;
32:     }
33:     memset_b(&tmp_info, NULL, sizeof(struct sysinfo));
34:     tmp_info.loads[0] = avenrun[0] << (SI_LOAD_SHIFT - FSHIFT);
35:     tmp_info.loads[1] = avenrun[1] << (SI_LOAD_SHIFT - FSHIFT);
36:     tmp_info.loads[2] = avenrun[2] << (SI_LOAD_SHIFT - FSHIFT);
37:     tmp_info.uptime = kstat.uptime;
38:     tmp_info.totalram = kstat.total_mem_pages << PAGE_SHIFT;
39:     tmp_info.freeram = kstat.free_pages << PAGE_SHIFT;
40:     tmp_info.sharedram = 0;
41:     tmp_info.bufferram = kstat.buffers * 1024;
42:     tmp_info.totalswap = 0;
43:     tmp_info.freeswap = 0;
44:     FOR_EACH_PROCESS(p) {
45:         tmp_info.procs++;
46:         p = p->next;
47:     }
48:
49:     memcpy_b(info, &tmp_info, sizeof(struct sysinfo));
50:     return 0;
51: }

```

kernel/syscalls/time.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/time.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/kernel.h>
10: #include <fiwix/fs.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #include <fiwix/process.h>
15: #endif /*__DEBUG__*/
16:
17: int sys_time(__time_t *tloc)
18: {
19:     int errno;
20:
21: #ifdef __DEBUG__
22:     printk("(pid %d) sys_time() -> ", current->pid);
23: #endif /*__DEBUG__*/
24:
25:     if(tloc) {
26:         if((errno = check_user_area(VERIFY_WRITE, tloc, sizeof(__time_t)
))) {
27:             return errno;
28:         }
29:         *tloc = CURRENT_TIME;
30:     }
31:
32: #ifdef __DEBUG__
33:     printk("%d\n", CURRENT_TIME);
34: #endif /*__DEBUG__*/
35:
36:     return CURRENT_TIME;
37: }
```

kernel/syscalls/times.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/times.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/types.h>
10: #include <fiwix/syscalls.h>
11: #include <fiwix/times.h>
12:
13: #ifdef __DEBUG__
14: #include <fiwix/stdio.h>
15: #include <fiwix/process.h>
16: #endif /* __DEBUG__ */
17:
18: int sys_times(struct tms *buf)
19: {
20:     int errno;
21:
22: #ifdef __DEBUG__
23:     printk("(pid %d) sys_times(0x%08x) -> ", (unsigned int )buf);
24: #endif /* __DEBUG__ */
25:
26:     if((errno = check_user_area(VERIFY_WRITE, buf, sizeof(struct tms)))) {
27:         return errno;
28:     }
29:     if(buf) {
30:         buf->tms_utime = tv2ticks(&current->usage.ru_utime);
31:         buf->tms_stime = tv2ticks(&current->usage.ru_stime);
32:         buf->tms_cutime = tv2ticks(&current->cusage.ru_utime);
33:         buf->tms_cstime = tv2ticks(&current->cusage.ru_stime);
34:     }
35:
36:     return kstat.ticks;
37: }
```

kernel/syscalls/truncate.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/truncate.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/stat.h>
11: #include <fiwix/errno.h>
12: #include <fiwix/string.h>
13:
14: #ifdef __DEBUG__
15: #include <fiwix/stdio.h>
16: #include <fiwix/process.h>
17: #endif /*__DEBUG__ */
18:
19: int sys_truncate(const char *path, __off_t length)
20: {
21:     struct inode *i;
22:     char *tmp_name;
23:     int errno;
24:
25: #ifdef __DEBUG__
26:     printk("(pid %d) sys_truncate(%s, %d)\n", current->pid, path, length);
27: #endif /*__DEBUG__ */
28:
29:     if((errno = malloc_name(path, &tmp_name)) < 0) {
30:         return errno;
31:     }
32:     if((errno = namei(tmp_name, &i, NULL, FOLLOW_LINKS)) {
33:         free_name(tmp_name);
34:         return errno;
35:     }
36:     if(S_ISDIR(i->i_mode)) {
37:         iput(i);
38:         free_name(tmp_name);
39:         return -EISDIR;
40:     }
41:     if(IS_RDONLY_FS(i)) {
42:         iput(i);
43:         free_name(tmp_name);
44:         return -EROFS;
45:     }
46:     if(check_permission(TO_WRITE, i) < 0) {
47:         iput(i);
48:         free_name(tmp_name);
49:         return -EACCES;
50:     }
51:     if(length == i->i_size) {
52:         iput(i);
53:         free_name(tmp_name);
54:         return 0;
55:     }
56:
57:     errno = 0;
58:     if(i->fsop && i->fsop->truncate) {
59:         inode_lock(i);
60:         errno = i->fsop->truncate(i, length);
61:         inode_unlock(i);
62:     }
63:     iput(i);
64:     free_name(tmp_name);
65:     return errno;
66: }

```

kernel/syscalls/umask.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/umask.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/process.h>
10: #include <fiwix/stat.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #endif /*__DEBUG__*/
15:
16: int sys_umask(__mode_t mask)
17: {
18:     __mode_t old_umask;
19:
20: #ifdef __DEBUG__
21:     printk("(pid %d) sys_umask(%d)\n", current->pid, mask);
22: #endif /*__DEBUG__*/
23:
24:     old_umask = current->umask;
25:     current->umask = mask & (S_IRWXU | S_IRWXG | S_IRWXO);
26:     return old_umask;
27: }
```

kernel/syscalls/umount2.c

Page 1/2

```

1: /*
2:  * fiwix/kernel/syscalls/umount2.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/filesystems.h>
11: #include <fiwix/stat.h>
12: #include <fiwix/sleep.h>
13: #include <fiwix/devices.h>
14: #include <fiwix/buffer.h>
15: #include <fiwix/errno.h>
16: #include <fiwix/stdio.h>
17: #include <fiwix/string.h>
18:
19: static struct resource umount_resource = { NULL, NULL };
20:
21: int sys_umount2(const char *target, int flags)
22: {
23:     struct inode *i_target;
24:     struct mount *mt = NULL;
25:     struct filesystems *fs;
26:     struct device *d;
27:     struct inode dummy_i;
28:     struct superblock *sb;
29:     char *tmp_target;
30:     __dev_t dev;
31:     int errno;
32:
33: #ifdef __DEBUG__
34:     printk("(pid %d) sys_umount2(%s, 0x%08x)\n", current->pid, target, flags
);
35: #endif /*__DEBUG__ */
36:
37:     if(!IS_SUPERUSER) {
38:         return -EPERM;
39:     }
40:     if((errno = malloc_name(target, &tmp_target)) < 0) {
41:         return errno;
42:     }
43:     if((errno = namei(tmp_target, &i_target, NULL, FOLLOW_LINKS)) {
44:         free_name(tmp_target);
45:         return errno;
46:     }
47:     if(!S_ISBLK(i_target->i_mode) && !S_ISDIR(i_target->i_mode)) {
48:         iput(i_target);
49:         free_name(tmp_target);
50:         return -EINVAL;
51:     }
52:
53:     if(!(mt = get_mount_point(i_target))) {
54:         iput(i_target);
55:         free_name(tmp_target);
56:         return -EINVAL;
57:     }
58:     if(S_ISBLK(i_target->i_mode)) {
59:         dev = i_target->rdev;
60:     } else {
61:         dev = i_target->sb->dev;
62:     }
63:
64:     if(!(sb = get_superblock(dev))) {
65:         printk("WARNING: %s(): unable to get superblock from device %d,%
d\n", __FUNCTION__, MAJOR(dev), MINOR(dev));

```


kernel/syscalls/umount2.c

Page 2/2

```

66:         iput(i_target);
67:         free_name(tmp_target);
68:         return -EINVAL;
69:     }
70:
71:     /*
72:      * We must free now the inode in order to avoid having its 'count' to 2
73:      * when calling check_fs_busy(), specially if sys_umount() was called
74:      * using the mount-point instead of the device.
75:      */
76:     iput(i_target);
77:     free_name(tmp_target);
78:
79:     if(check_fs_busy(dev, sb->root)) {
80:         return -EBUSY;
81:     }
82:
83:     lock_resource(&umount_resource);
84:
85:     fs = mt->fs;
86:     if(fs->fsop && fs->fsop->release_superblock) {
87:         fs->fsop->release_superblock(sb);
88:     }
89:     if(sb->fsop->flags & FSOP_REQUIRES_DEV) {
90:         if(!(d = get_device(BLK_DEV, dev))) {
91:             printk("WARNING: %s(): block device %d,%d not registered
!\n", __FUNCTION__, MAJOR(dev), MINOR(dev));
92:             unlock_resource(&umount_resource);
93:             return -EINVAL;
94:         }
95:         memset_b(&dummy_i, 0, sizeof(struct inode));
96:         dummy_i.dev = dummy_i.rdev = dev;
97:         if(d && d->fsop && d->fsop->close) {
98:             d->fsop->close(&dummy_i, NULL);
99:         }
100:     }
101:
102:     sb->dir->mount_point = NULL;
103:     iput(sb->root);
104:     iput(sb->dir);
105:
106:     sync_superblocks(dev);
107:     sync_inodes(dev);
108:     sync_buffers(dev);
109:     invalidate_buffers(dev);
110:     invalidate_inodes(dev);
111:
112:     release_mount_point(mt);
113:     unlock_resource(&umount_resource);
114:     return 0;
115: }

```

kernel/syscalls/umount.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/umount.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/syscalls.h>
9: #include <fiwix/process.h>
10:
11: #ifdef __DEBUG__
12: #include <fiwix/stdio.h>
13: #endif /*__DEBUG__ */
14:
15: int sys_umount(const char *target)
16: {
17: #ifdef __DEBUG__
18:     printk("(pid %d) sys_umount(%s)\n", current->pid, target);
19: #endif /*__DEBUG__ */
20:
21:     return sys_umount2(target, 0);
22: }
```

kernel/syscalls/uname.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/uname.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/utsname.h>
10: #include <fiwix/string.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #include <fiwix/process.h>
15: #endif /*__DEBUG__ */
16:
17: int sys_uname(struct old_utsname *uname)
18: {
19:     int errno;
20:
21: #ifdef __DEBUG__
22:     printk("(pid %d) sys_uname(0x%08x) -> returning ", current->pid, (unsign
ed int)uname);
23: #endif /*__DEBUG__ */
24:
25:     if((errno = check_user_area(VERIFY_WRITE, uname, sizeof(struct old_utsna
me)))) {
26:         return errno;
27:     }
28:     memcpy_b(&uname->sysname, &sys_utsname.sysname, sizeof(sys_utsname.sysna
me));
29:     memcpy_b(&uname->nodename, &sys_utsname.nodename, sizeof(sys_utsname.nod
ename));
30:     memcpy_b(&uname->release, &sys_utsname.release, sizeof(sys_utsname.relea
se));
31:     memcpy_b(&uname->version, &sys_utsname.version, sizeof(sys_utsname.versi
on));
32:     memcpy_b(&uname->machine, &sys_utsname.machine, sizeof(sys_utsname.machi
ne));
33:     return 0;
34: }

```

kernel/syscalls/unlink.c

Page 1/2

```

1: /*
2:  * fiwix/kernel/syscalls/unlink.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/syscalls.h>
10: #include <fiwix/stat.h>
11: #include <fiwix/errno.h>
12: #include <fiwix/string.h>
13:
14: #ifdef __DEBUG__
15: #include <fiwix/stdio.h>
16: #include <fiwix/process.h>
17: #endif /* __DEBUG__ */
18:
19: int sys_unlink(const char *filename)
20: {
21:     struct inode *i, *dir;
22:     char *tmp_name, *basename;
23:     int errno;
24:
25: #ifdef __DEBUG__
26:     printk("(pid %d) sys_unlink('%s')\n", current->pid, filename);
27: #endif /* __DEBUG__ */
28:
29:     if((errno = malloc_name(filename, &tmp_name)) < 0) {
30:         return errno;
31:     }
32:     if((errno = namei(tmp_name, &i, &dir, !FOLLOW_LINKS)) {
33:         if(dir) {
34:             iput(dir);
35:         }
36:         free_name(tmp_name);
37:         return errno;
38:     }
39:     if(S_ISDIR(i->i_mode)) {
40:         iput(i);
41:         iput(dir);
42:         free_name(tmp_name);
43:         return -EPERM; /* Linux returns -EISDIR */
44:     }
45:     if(IS_RDONLY_FS(i)) {
46:         iput(i);
47:         iput(dir);
48:         free_name(tmp_name);
49:         return -EROFS;
50:     }
51:     if(check_permission(TO_EXEC | TO_WRITE, dir) < 0) {
52:         iput(i);
53:         iput(dir);
54:         free_name(tmp_name);
55:         return -EACCES;
56:     }
57:
58:     /* check sticky permission bit */
59:     if(dir->i_mode & S_ISVTX) {
60:         if(check_user_permission(i)) {
61:             iput(i);
62:             iput(dir);
63:             free_name(tmp_name);
64:             return -EPERM;
65:         }
66:     }
67:

```

kernel/syscalls/unlink.c

Page 2/2

```
68:         basename = get_basename(filename);
69:         if(dir->fsop && dir->fsop->unlink) {
70:             errno = dir->fsop->unlink(dir, i, basename);
71:         } else {
72:             errno = -EPERM;
73:         }
74:         iput(i);
75:         iput(dir);
76:         free_name(tmp_name);
77:         return errno;
78: }
```

kernel/syscalls/ustat.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/ustat.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/filesystems.h>
11: #include <fiwix/ustat.h>
12: #include <fiwix/statfs.h>
13: #include <fiwix/errno.h>
14: #include <fiwix/string.h>
15:
16: #ifdef __DEBUG__
17: #include <fiwix/stdio.h>
18: #include <fiwix/process.h>
19: #endif /*__DEBUG__*/
20:
21: int sys_ustat(__dev_t dev, struct ustat *ubuf)
22: {
23:     struct superblock *sb;
24:     struct statfs statfsbuf;
25:     int errno;
26:
27: #ifdef __DEBUG__
28:     printk("(pid %d) sys_ustat(%d, 0x%08x)\n", current->pid, dev, (int)ubuf)
;
29: #endif /*__DEBUG__*/
30:     if((errno = check_user_area(VERIFY_WRITE, ubuf, sizeof(struct ustat))))
{
31:         return errno;
32:     }
33:     if(!(sb = get_superblock(dev))) {
34:         return -EINVAL;
35:     }
36:     if(sb->fsop && sb->fsop->statfs) {
37:         sb->fsop->statfs(sb, &statfsbuf);
38:         memset_b(ubuf, NULL, sizeof(struct ustat));
39:         ubuf->f_tfree = statfsbuf.f_bfree;
40:         ubuf->f_tinode = statfsbuf.f_ffree;
41:         return 0;
42:     }
43:     return -ENOSYS;
44: }

```

kernel/syscalls/utime.c

Page 1/2

```

1: /*
2:  * fiwix/kernel/syscalls/utime.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/utime.h>
11: #include <fiwix/stat.h>
12: #include <fiwix/errno.h>
13: #include <fiwix/string.h>
14:
15: #ifdef __DEBUG__
16: #include <fiwix/stdio.h>
17: #include <fiwix/process.h>
18: #endif /*__DEBUG__ */
19:
20: int sys_utime(const char *filename, struct utimbuf *times)
21: {
22:     struct inode *i;
23:     char *tmp_name;
24:     int errno;
25:
26: #ifdef __DEBUG__
27:     printk("(pid %d) sys_utime('%s', 0x%08x)\n", current->pid, filename, (in
t)times);
28: #endif /*__DEBUG__ */
29:
30:     if((errno = malloc_name(filename, &tmp_name)) < 0) {
31:         return errno;
32:     }
33:     if((errno = namei(tmp_name, &i, NULL, FOLLOW_LINKS))) {
34:         free_name(tmp_name);
35:         return errno;
36:     }
37:
38:     if(IS_RDONLY_FS(i)) {
39:         iput(i);
40:         free_name(tmp_name);
41:         return -EROFS;
42:     }
43:
44:     if(!times) {
45:         if(check_user_permission(i) || check_permission(TO_WRITE, i)) {
46:             iput(i);
47:             free_name(tmp_name);
48:             return -EACCES;
49:         }
50:         i->i_atime = CURRENT_TIME;
51:         i->i_mtime = CURRENT_TIME;
52:     } else {
53:         if((errno = check_user_area(VERIFY_READ, times, sizeof(struct ut
imbuf)))) {
54:             iput(i);
55:             free_name(tmp_name);
56:             return errno;
57:         }
58:         if(check_user_permission(i)) {
59:             iput(i);
60:             free_name(tmp_name);
61:             return -EPERM;
62:         }
63:         i->i_atime = times->actime;
64:         i->i_mtime = times->modtime;
65:     }

```

kernel/syscalls/utime.c

Page 2/2

```
66:
67:         i->i_ctime = CURRENT_TIME;
68:         i->dirty = 1;
69:         iput(i);
70:         free_name(tmp_name);
71:         return 0;
72: }
```


kernel/syscalls/wait4.c

Page 1/2

```

1: /*
2:  * fiwix/kernel/syscalls/wait4.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/resource.h>
11: #include <fiwix/signal.h>
12: #include <fiwix/sched.h>
13: #include <fiwix/sleep.h>
14: #include <fiwix/errno.h>
15:
16: #ifdef __DEBUG__
17: #include <fiwix/stdio.h>
18: #include <fiwix/process.h>
19: #endif /*__DEBUG__*/
20:
21: int sys_wait4(__pid_t pid, int *status, int options, struct rusage *ru)
22: {
23:     struct proc *p;
24:     int flag, signum, errno;
25:
26: #ifdef __DEBUG__
27:     printk("(pid %d) sys_wait4(%d, status, %d)\n", current->pid, pid, option
s);
28: #endif /*__DEBUG__*/
29:
30:     if(ru) {
31:         if((errno = check_user_area(VERIFY_WRITE, ru, sizeof(struct rusa
ge)))) {
32:             return errno;
33:         }
34:     }
35:     while(current->children) {
36:         flag = 0;
37:         FOR_EACH_PROCESS(p) {
38:             if(p->ppid != current->pid) {
39:                 p = p->next;
40:                 continue;
41:             }
42:             if(pid > 0) {
43:                 if(p->pid == pid) {
44:                     flag = 1;
45:                 }
46:             }
47:             if(!pid) {
48:                 if(p->pgid == current->pgid) {
49:                     flag = 1;
50:                 }
51:             }
52:             if(pid < -1) {
53:                 if(p->pgid == -pid) {
54:                     flag = 1;
55:                 }
56:             }
57:             if(pid == -1) {
58:                 flag = 1;
59:             }
60:             if(flag) {
61:                 if(p->state == PROC_STOPPED) {
62:                     if(p->exit_code) {
63:                         if(status) {
64:                             *status = (p->exit_code
<< 8) | 0x7F;

```

kernel/syscalls/wait4.c

Page 2/2

```
65:                                     }
66:                                     p->exit_code = 0;
67:                                     }
68:                                     if(ru) {
69:                                         get_rusage(p, ru);
70:                                     }
71:                                     return p->pid;
72:                                     }
73:                                     if(p->state == PROC_ZOMBIE) {
74:                                         add_rusage(p);
75:                                         if(status) {
76:                                             *status = p->exit_code;
77:                                         }
78:                                         if(ru) {
79:                                             get_rusage(p, ru);
80:                                         }
81:                                         return remove_zombie(p);
82:                                     }
83:                                     }
84:                                     p = p->next;
85:                                     flag = 0;
86:                                     }
87:                                     if(options & WNOHANG) {
88:                                         if(flag) {
89:                                             return 0;
90:                                         }
91:                                         break;
92:                                     }
93:                                     if((signum = sleep(&sys_wait4, PROC_INTERRUPTIBLE))) {
94:                                         return signum;
95:                                     }
96:                                     current->sigpending &= SIG_MASK(SIGCHLD);
97:                                     }
98:                                     return -ECHILD;
99:     }
```

kernel/syscalls/waitpid.c

Page 1/1

```
1: /*
2:  * fiwix/kernel/syscalls/waitpid.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/syscalls.h>
10: #include <fiwix/string.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #include <fiwix/process.h>
15: #endif /*__DEBUG__*/
16:
17: int sys_waitpid(__pid_t pid, int *status, int options)
18: {
19: #ifdef __DEBUG__
20:     printk("(pid %d) sys_waitpid(%d, 0x%08x, %d)\n", current->pid, pid, sta
tus ? *status : 0, options);
21: #endif /*__DEBUG__*/
22:     return sys_wait4(pid, status, options, NULL);
23: }
```

kernel/syscalls/write.c

Page 1/1

```

1: /*
2:  * fiwix/kernel/syscalls/write.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/fs.h>
9: #include <fiwix/fcntl.h>
10: #include <fiwix/errno.h>
11:
12: #ifdef __DEBUG__
13: #include <fiwix/stdio.h>
14: #include <fiwix/process.h>
15: #endif /*__DEBUG__*/
16:
17: int sys_write(unsigned int ufd, const char *buf, int count)
18: {
19:     struct inode *i;
20:     int errno;
21:
22: #ifdef __DEBUG__
23: /*      printk("(pid %d) sys_write(%d, '%s', %d)\n", current->pid, ufd, buf, cou
nt);*/
24:     printk("(pid %d) sys_write(%d, 0x%08x, %d) -> ", current->pid, ufd, buf,
count);
25: #endif /*__DEBUG__*/
26:
27:     CHECK_UFD(ufd);
28:     if((errno = check_user_area(VERIFY_READ, buf, count))) {
29:         return errno;
30:     }
31:     if(fd_table[current->fd[ufd]].flags & O_RDONLY) {
32:         return -EBADF;
33:     }
34:     if(!count) {
35:         return 0;
36:     }
37:     if(count < 0) {
38:         return -EINVAL;
39:     }
40:     i = fd_table[current->fd[ufd]].inode;
41:     if(i->fsop && i->fsop->write) {
42:         errno = i->fsop->write(i, &fd_table[current->fd[ufd]], buf, coun
t);
43: #ifdef __DEBUG__
44:         printk("%d\n", errno);
45: #endif /*__DEBUG__*/
46:         return errno;
47:     }
48:     return -EINVAL;
49: }

```

drivers/block/dma.c

Page 1/2

```

1:  /*
2:  *  fiwix/drivers/block/dma.c
3:  *
4:  *  Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  *  Distributed under the terms of the Fiwix License.
6:  */
7:
8:  #include <fiwix/asm.h>
9:  #include <fiwix/dma.h>
10: #include <fiwix/string.h>
11:
12: /*
13:  *  DMA Channel  Page  Address  Count
14:  *  -----
15:  *  0 (8 bit)      87h      0h      1h
16:  *  1 (8 bit)      83h      2h      3h
17:  *  2 (8 bit)      81h      4h      5h
18:  *  3 (8 bit)      82h      6h      7h
19:  *  4 (16 bit)     8Fh      C0h     C2h
20:  *  5 (16 bit)     8Bh      C4h     C6h
21:  *  6 (16 bit)     89h      C8h     CAh
22:  *  7 (16 bit)     8Ah      CCh     CEh
23:  */
24:
25: #define LOW_BYTE(addr)  (addr & 0x00FF)
26: #define HIGH_BYTE(addr) ((addr & 0xFF00) >> 8)
27:
28: unsigned char dma_mask[DMA_CHANNELS] =
29:     { 0x0A, 0x0A, 0x0A, 0x0A, 0xD4, 0xD4, 0xD4, 0xD4 };
30: unsigned char dma_mode[DMA_CHANNELS] =
31:     { 0x0B, 0x0B, 0x0B, 0x0B, 0xD6, 0xD6, 0xD6, 0xD6 };
32: unsigned char dma_clear[DMA_CHANNELS] =
33:     { 0x0C, 0x0C, 0x0C, 0x0C, 0xD8, 0xD8, 0xD8, 0xD8 };
34: unsigned char dma_page[DMA_CHANNELS] =
35:     { 0x87, 0x83, 0x81, 0x82, 0x8F, 0x8B, 0x89, 0x8A };
36: unsigned char dma_address[DMA_CHANNELS] =
37:     { 0x00, 0x02, 0x04, 0x06, 0xC0, 0xC4, 0xC8, 0xCC };
38: unsigned char dma_count[DMA_CHANNELS] =
39:     { 0x01, 0x03, 0x05, 0x07, 0xC2, 0xC6, 0xCA, 0xCE };
40:
41:
42: void start_dma(int channel, void *address, unsigned int count, int mode)
43: {
44:     /* setup (mask) the DMA channel */
45:     outport_b(dma_mask[channel], DMA_MASK_CHANNEL | channel);
46:
47:     /* clear any data transfers that are currently executing */
48:     outport_b(dma_clear[channel], 0);
49:
50:     /* set the specified mode */
51:     outport_b(dma_mode[channel], mode | channel);
52:
53:     /* set the offset address */
54:     outport_b(dma_address[channel], LOW_BYTE((unsigned int)address));
55:     outport_b(dma_address[channel], HIGH_BYTE((unsigned int)address));
56:
57:     /* set the physical page */
58:     outport_b(dma_page[channel], (unsigned int)address >> 16);
59:
60:     /* the true (internal) length sent to the DMA is actually length + 1 */
61:     count--;
62:
63:     /* set the length of the data */
64:     outport_b(dma_count[channel], LOW_BYTE(count));
65:     outport_b(dma_count[channel], HIGH_BYTE(count));
66:
67:     /* clear the mask */

```

drivers/block/dma.c

Page 2/2

```
68:         outport_b(dma_mask[channel], DMA_UNMASK_CHANNEL | channel);
69:     }
70:
71: int dma_register(int channel, char *dev_name)
72: {
73:     if(dma_resources[channel]) {
74:         return 1;
75:     }
76:     dma_resources[channel] = dev_name;
77:     return 0;
78: }
79:
80: int dma_unregister(int channel)
81: {
82:     if(!dma_resources[channel]) {
83:         return 1;
84:     }
85:
86:     dma_resources[channel] = NULL;
87:     return 0;
88: }
89:
90: void dma_init(void)
91: {
92:     memset_b(dma_resources, NULL, sizeof(dma_resources));
93: }
```

drivers/block/floppy.c

Page 1/15

```

1: /*
2:  * fiwix/drivers/block/floppy.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/floppy.h>
10: #include <fiwix/ioctl.h>
11: #include <fiwix/devices.h>
12: #include <fiwix/part.h>
13: #include <fiwix/fs.h>
14: #include <fiwix/buffer.h>
15: #include <fiwix/sleep.h>
16: #include <fiwix/timer.h>
17: #include <fiwix/sched.h>
18: #include <fiwix/errno.h>
19: #include <fiwix/pic.h>
20: #include <fiwix/cmos.h>
21: #include <fiwix/dma.h>
22: #include <fiwix/stdio.h>
23: #include <fiwix/string.h>
24:
25: #define WAIT_MOTOR_OFF (3 * HZ)      /* time waiting to turn the motor off */
26: #define WAIT_FDC        WAIT_MOTOR_OFF
27:
28: #define INVALID_TRACK  -1
29:
30: #define DEV_TYPE_SHIFT 2             /* right shift to match with the floppy
31:                                     type when minor > 3 */
32:
33: static int need_reset = 0;
34: static int fdc_wait_interrupt = 0;
35: static int fdc_timeout = 0;
36: static unsigned char fdc_results[MAX_FDC_RESULTS];
37: static struct resource floppy_resource = { NULL, NULL };
38:
39: static struct fddt fdd_type[] = {
40: /*
41:  * R (data rate): 0 = 500Kb/s, 2 = 250Kb/s, 3 = 1Mb/s
42:  * SPEC(IFY) 0xAF: SRT = 6ms, HUT = 240ms (500Kb/s)
43:  * SPEC(IFY) 0xD7: SRT = 6ms, HUT = 240ms (250Kb/s)
44:  * SPEC(IFY) 0xDF: SRT = 3ms, HUT = 240ms (500Kb/s)
45:  * Head Load Time 0x02: HLT = 4ms (500Kb/s), Non-DMA = 0 (DMA enabled)
46:  *
47:  *          SIZE      KB   T   S   H   G_RW  G_FM  R   SPEC  HLT   NAME
48:  *          -----
49:  *          {    0,    0,  0,  0,  0, 0x00, 0x00, 0, 0x00, 0x00, NULL
50:  *          {  720,  360, 40,  9,  2, 0x2A, 0x50, 2, 0xD7, 0x02, "360KB 5.25\""},
51:  *          { 2400, 1200, 80, 15, 2, 0x2A, 0x50, 0, 0xAF, 0x02, "1.2MB 5.25\""},
52:  *          { 1440,  720, 80,  9,  2, 0x1B, 0x54, 2, 0xD7, 0x02, "720KB 3.5\""},
53:  *          { 2880, 1440, 80, 18, 2, 0x1B, 0x54, 0, 0xAF, 0x02, "1.44MB 3.5\""},
54:  *          { 5760, 2880, 80, 36, 2, 0x38, 0x53, 3, 0xDF, 0x02, "2.88MB 3.5\""}, */
55: };
56:
57: /* buffer area used for I/O operations (1KB) */
58: char fdc_transfer_area[BPS * 2];
59:
60: struct fdd_status {
61:     char type;                /* floppy disk drive type */
62:     char motor;
63:     char recalibrated;
64:     char current_track;
65: };
66:
67: static struct fdd_status fdd_status[] = {

```

drivers/block/floppy.c

Page 2/15

```

68:         { 0, 0, 0, INVALID_TRACK },
69:         { 0, 0, 0, INVALID_TRACK },
70: };
71:
72: static unsigned char current_fdd = 0;
73: static struct fdtdt *current_fdd_type;
74: static unsigned int fdd_sizes[256];
75:
76: static struct fs_operations fdc_driver_fsop = {
77:     0,
78:     0,
79:
80:     fdc_open,
81:     fdc_close,
82:     NULL,                /* read */
83:     NULL,                /* write */
84:     fdc_ioctl,
85:     fdc_lseek,
86:     NULL,                /* readdir */
87:     NULL,                /* mmap */
88:     NULL,                /* select */
89:
90:     NULL,                /* readlink */
91:     NULL,                /* followlink */
92:     NULL,                /* bmap */
93:     NULL,                /* lockup */
94:     NULL,                /* rmdir */
95:     NULL,                /* link */
96:     NULL,                /* unlink */
97:     NULL,                /* symlink */
98:     NULL,                /* mkdir */
99:     NULL,                /* mknod */
100:    NULL,                /* truncate */
101:    NULL,                /* create */
102:    NULL,                /* rename */
103:
104:    fdc_read,
105:    fdc_write,
106:
107:    NULL,                /* read_inode */
108:    NULL,                /* write_inode */
109:    NULL,                /* ialloc */
110:    NULL,                /* ifree */
111:    NULL,                /* statfs */
112:    NULL,                /* read_superblock */
113:    NULL,                /* remount_fs */
114:    NULL,                /* write_superblock */
115:    NULL,                /* release_superblock */
116: };
117:
118: static struct device floppy_device = {
119:     "floppy",
120:     FDC_MAJOR,
121:     { 0, 0, 0, 0, 0, 0, 0, 0 },
122:     BLKSIZE_1K,
123:     &fdd_sizes,
124:     &fdc_driver_fsop,
125:     NULL
126: };
127:
128: static struct interrupt irq_config_floppy = { 0, "floppy", &irq_floppy, NULL };
129:
130: static int fdc_in(void)
131: {
132:     int n;
133:     unsigned char status;
134:

```


drivers/block/floppy.c

Page 3/15

```

135:         if(need_reset) {
136:             return -1;
137:         }
138:
139:         for(n = 0; n < 10000; n++) {
140:             status = inport_b(FDC_MSR) & (FDC_RQM | FDC_DIO);
141:             if(status == FDC_RQM) {
142:                 return 0;
143:             }
144:             if(status == (FDC_RQM | FDC_DIO)) {
145:                 return inport_b(FDC_DATA);
146:             }
147:         }
148:         need_reset = 1;
149:         printk("WARNING: %s(): fd%d: timeout on %s.\n", __FUNCTION__, current_fd
d, floppy_device.name);
150:         return -1;
151:     }
152:
153:     static void fdc_out(unsigned char value)
154:     {
155:         int n;
156:         unsigned char status;
157:
158:         if(need_reset) {
159:             return;
160:         }
161:
162:         for(n = 0; n < 10000; n++) {
163:             status = inport_b(FDC_MSR) & (FDC_RQM | FDC_DIO);
164:             if(status == FDC_RQM) {
165:                 outport_b(FDC_DATA, value);
166:                 return;
167:             }
168:         }
169:
170:         need_reset = 1;
171:         printk("WARNING: %s(): fd%d: unable to send byte 0x%x on %s.\n", __FUNCT
ION__, current_fdd, value, floppy_device.name);
172:     }
173:
174:     static void fdc_get_results(void)
175:     {
176:         int n;
177:
178:         memset_b(fdc_results, 0, sizeof(fdc_results));
179:         for(n = 0; n < MAX_FDC_RESULTS; n++) {
180:             fdc_results[n] = fdc_in();
181:         }
182:         return;
183:     }
184:
185:     static int fdc_motor_on(void)
186:     {
187:         struct callout_req creq;
188:         int errno;
189:
190:         if(fdd_status[current_fdd].motor) {
191:             return 0;
192:         }
193:
194:         /* select floppy disk drive and turn on its motor */
195:         outport_b(FDC_DOR, (FDC_DRIVE0 << current_fdd) | FDC_DMA_ENABLE | FDC_EN
ABLE | current_fdd);
196:         fdd_status[current_fdd].motor = 1;
197:         fdd_status[!current_fdd].motor = 0;
198:

```

drivers/block/floppy.c

Page 4/15

```

199:      /* fixed spin-up time of 500ms for 3.5" and 5.25" */
200:      creq.fn = fdc_timer;
201:      creq.arg = FDC_TR_MOTOR;
202:      add_callout(&creq, HZ / 2);
203:      sleep(&fdc_motor_on, PROC_UNINTERRUPTIBLE);
204:
205:      errno = 0;
206:
207:      /* check for a disk change */
208:      if(inport_b(FDC_DIR) & 0x80) {
209:          errno = 1;
210:      }
211:
212:      return errno;
213: }
214:
215: static void do_motor_off(unsigned int fdd)
216: {
217:     outport_b(FDC_DOR, FDC_DMA_ENABLE | FDC_ENABLE | fdd);
218:     fdd_status[fdd].motor = 0;
219:     fdd_status[0].motor = fdd_status[1].motor = 0;
220: }
221:
222: static void fdc_motor_off(void)
223: {
224:     struct callout_req creq;
225:
226:     creq.fn = do_motor_off;
227:     creq.arg = current_fdd;
228:     add_callout(&creq, WAIT_FDC);
229: }
230:
231: static void fdc_reset(void)
232: {
233:     int n;
234:     struct callout_req creq;
235:
236:     need_reset = 0;
237:
238:     fdc_wait_interrupt = FDC_RESET;
239:     outport_b(FDC_DOR, 0); /* enter in reset mode */
240: /* outport_b(FDC_DOR, FDC_DMA_ENABLE); */
241:     for(n = 0; n < 1000; n++) { /* recovery time */
242:         NOP();
243:     }
244:     outport_b(FDC_DOR, FDC_DMA_ENABLE | FDC_ENABLE);
245:
246:     creq.fn = fdc_timer;
247:     creq.arg = FDC_TR_DEFAULT;
248:     add_callout(&creq, WAIT_FDC);
249:     /* avoid sleep if interrupt already happened */
250:     if(fdc_wait_interrupt) {
251:         sleep(&irq_floppy, PROC_UNINTERRUPTIBLE);
252:     }
253:     if(fdc_timeout) {
254:         need_reset = 1;
255:         printk("WARNING: %s(): fd%d: timeout on %s.\n", __FUNCTION__, cu
rrent_fdd, floppy_device.name);
256:     }
257:     del_callout(&creq);
258:
259:     fdd_status[0].motor = fdd_status[1].motor = 0;
260:     fdd_status[current_fdd].recalibrated = 0;
261:
262:     /* assumes drive polling mode is ON (by default) */
263:     for(n = 0; n < 4; n++) {
264:         fdc_out(FDC_SENSEI);

```

drivers/block/floppy.c

Page 5/15

```

265:             fdc_get_results();
266:         }
267:
268:         /* keeps controller informed on the drive about to use */
269:         fdc_out(FDC_SPECIFY);
270:         fdc_out(current_fdd_type->spec);
271:         fdc_out(current_fdd_type->hlt);
272:
273:         /* set data rate */
274:         outport_b(FDC_CCR, current_fdd_type->rate);
275:     }
276:
277:     static int fdc_recalibrate(void)
278:     {
279:         struct callout_req creq;
280:
281:         if(need_reset) {
282:             return 1;
283:         }
284:
285:         fdc_wait_interrupt = FDC_RECALIBRATE;
286:         fdc_motor_on();
287:         fdc_out(FDC_RECALIBRATE);
288:         fdc_out(current_fdd);
289:
290:         if(need_reset) {
291:             return 1;
292:         }
293:
294:         creq.fn = fdc_timer;
295:         creq.arg = FDC_TR_DEFAULT;
296:         add_callout(&creq, WAIT_FDC);
297:         /* avoid sleep if interrupt already happened */
298:         if(fdc_wait_interrupt) {
299:             sleep(&irq_floppy, PROC_UNINTERRUPTIBLE);
300:         }
301:         if(fdc_timeout) {
302:             need_reset = 1;
303:             printk("WARNING: %s(): fd%d: timeout on %s.\n", __FUNCTION__, cu
rrrent_fdd, floppy_device.name);
304:             return 1;
305:         }
306:
307:         del_callout(&creq);
308:         fdc_out(FDC_SENSEI);
309:         fdc_get_results();
310:
311:         /* PCN must be 0 indicating a successful position to track 0 */
312:         if((fdc_results[ST0] & (ST0_IC | ST0_SE | ST0_UC | ST0_NR)) != ST0_RECAL
IBRATE || fdc_results[ST_PCN]) {
313:             need_reset = 1;
314:             printk("WARNING: %s(): fd%d: unable to recalibrate on %s.\n", __
FUNCTION__, current_fdd, floppy_device.name);
315:             return 1;
316:         }
317:
318:         fdd_status[current_fdd].current_track = INVALID_TRACK;
319:         fdd_status[current_fdd].recalibrated = 1;
320:         fdc_motor_off();
321:         return 0;
322:     }
323:
324:     static int fdc_seek(int track, int head)
325:     {
326:         struct callout_req creq;
327:
328:         if(need_reset) {

```

drivers/block/floppy.c

Page 6/15

```

329:         return 1;
330:     }
331:
332:     if(!fdd_status[current_fdd].recalibrated) {
333:         if(fdc_recalibrate()) {
334:             return 1;
335:         }
336:     }
337:
338:     if(fdd_status[current_fdd].current_track == track) {
339:         return 0;
340:     }
341:
342:     fdc_wait_interrupt = FDC_SEEK;
343:     fdc_motor_on();
344:     fdc_out(FDC_SEEK);
345:     fdc_out((head << 2) | current_fdd);
346:     fdc_out(track);
347:
348:     if(need_reset) {
349:         return 1;
350:     }
351:
352:     creq.fn = fdc_timer;
353:     creq.arg = FDC_TR_DEFAULT;
354:     add_callout(&creq, WAIT_FDC);
355:     /* avoid sleep if interrupt already happened */
356:     if(fdc_wait_interrupt) {
357:         sleep(&irq_floppy, PROC_UNINTERRUPTIBLE);
358:     }
359:     if(fdc_timeout) {
360:         need_reset = 1;
361:         printk("WARNING: %s(): fd%d: timeout on %s.\n", __FUNCTION__, cu
rrent_fdd, floppy_device.name);
362:         return 1;
363:     }
364:
365:     del_callout(&creq);
366:     fdc_out(FDC_SENSEI);
367:     fdc_get_results();
368:
369:     if((fdc_results[ST0] & (ST0_IC | ST0_SE | ST0_UC | ST0_NR)) != ST0_SEEK
|| fdc_results[ST_PCN] != track) {
370:         need_reset = 1;
371:         printk("WARNING: %s(): fd%d: unable to seek on %s.\n", __FUNCTIO
N__, current_fdd, floppy_device.name);
372:         return 1;
373:     }
374:
375:     fdc_motor_off();
376:     fdd_status[current_fdd].current_track = track;
377:     return 0;
378: }
379:
380: static int fdc_get_chip(void)
381: {
382:     unsigned char version, fifo, id;
383:
384:     fdc_out(FDC_VERSION);
385:     version = fdc_in();
386:     fdc_out(FDC_LOCK);
387:     fifo = fdc_in();
388:     fdc_out(FDC_PARTID);
389:     id = fdc_in();
390:
391:     if(version == 0x80) {
392:         if(fifo == 0x80) {

```

drivers/block/floppy.c

Page 7/15

```

393:         printk("(NEC D765/Intel 8272A/compatible)\n");
394:         return 0;
395:     }
396:     if(fifo == 0) {
397:         printk("(Intel 82072)\n");
398:         return 0;
399:     }
400: }
401:
402: if(version == 0x81) {
403:     printk("(Very Early Intel 82077/compatible)\n");
404:     return 0;
405: }
406:
407: if(version == 0x90) {
408:     if(fifo == 0x80) {
409:         printk("(Old Intel 82077, no FIFO)\n");
410:         return 0;
411:     }
412:     if(fifo == 0) {
413:         if(id == 0x80) {
414:             printk("(New Intel 82077)\n");
415:             return 0;
416:         }
417:         if(id == 0x41) {
418:             printk("(Intel 82078)\n");
419:             return 0;
420:         }
421:         if(id == 0x73) {
422:             printk("(National Semiconductor PC87306)\n");
423:             return 0;
424:         }
425:         printk("(Intel 82078 compatible)\n");
426:         return 0;
427:     }
428:     printk("(NEC 72065B)\n");
429:     return 0;
430: }
431:
432: if(version == 0xA0) {
433:     printk("(SMC FDC37c65C+)\n");
434:     return 0;
435: }
436: printk("(unknown controller chip)\n");
437: return 1;
438: }
439:
440: static int fdc_block2chs(__blk_t block, int blksize, int *cyl, int *head, int *s
ector)
441: {
442:     int spb = blksize / FDC_SECTSIZE;
443:
444:     *cyl = (block * spb) / (current_fdd_type->spt * current_fdd_type->heads)
;
445:     *head = ((block * spb) % (current_fdd_type->spt * current_fdd_type->head
s)) / current_fdd_type->spt;
446:     *sector = (((block * spb) % (current_fdd_type->spt * current_fdd_type->h
eads)) % current_fdd_type->spt) + 1;
447:
448:     if(*cyl >= current_fdd_type->tracks || *head >= current_fdd_type->heads
|| *sector > current_fdd_type->spt) {
449:         return 1;
450:     }
451:
452:     return 0;
453: }
454:

```

drivers/block/floppy.c

Page 8/15

```

455: static void set_current_fdd_type(int minor)
456: {
457:     current_fdd = minor & 1;
458:
459:     /* minors 0 and 1 are directly assigned */
460:     if(minor < 2) {
461:         current_fdd_type = &fdd_type[(int)fdd_status[current_fdd].type];
462:     } else {
463:         current_fdd_type = &fdd_type[minor >> DEV_TYPE_SHIFT];
464:     }
465: }
466:
467: void irq_floppy(int num, struct sigcontext *sc)
468: {
469:     if(!fdc_wait_interrupt) {
470:         printk("WARNING: %s(): fd%d: unexpected interrupt on %s.\n", __F
UNCTION__, current_fdd, floppy_device.name);
471:         need_reset = 1;
472:     } else {
473:         fdc_timeout = fdc_wait_interrupt = 0;
474:         wakeup(&irq_floppy);
475:     }
476: }
477:
478: void fdc_timer(unsigned int reason)
479: {
480:     switch(reason) {
481:         case FDC_TR_DEFAULT:
482:             fdc_timeout = 1;
483:             fdc_wait_interrupt = 0;
484:             wakeup(&irq_floppy);
485:             break;
486:         case FDC_TR_MOTOR:
487:             wakeup(&fdc_motor_on);
488:             break;
489:     }
490: }
491:
492: int fdc_open(struct inode *i, struct fd *fd_table)
493: {
494:     unsigned char minor;
495:
496:     minor = MINOR(i->rdev);
497:     if(!TEST_MINOR(floppy_device.minors, minor)) {
498:         return -ENXIO;
499:     }
500:
501:     lock_resource(&floppy_resource);
502:     set_current_fdd_type(minor);
503:     unlock_resource(&floppy_resource);
504:
505:     return 0;
506: }
507:
508: int fdc_close(struct inode *i, struct fd *fd_table)
509: {
510:     unsigned char minor;
511:
512:     minor = MINOR(i->rdev);
513:     if(!TEST_MINOR(floppy_device.minors, minor)) {
514:         return -ENXIO;
515:     }
516:
517:     lock_resource(&floppy_resource);
518:     set_current_fdd_type(minor);
519:     unlock_resource(&floppy_resource);
520:

```

drivers/block/floppy.c

Page 9/15

```

521:         return 0;
522:     }
523:
524:     int fdc_read(__dev_t dev, __blk_t block, char *buffer, int blksize)
525:     {
526:         unsigned char minor;
527:         unsigned int sectors_read;
528:         int cyl, head, sector;
529:         int retries;
530:         struct callout_req creq;
531:         struct device *d;
532:
533:         minor = MINOR(dev);
534:         if(!TEST_MINOR(floppy_device.minors, minor)) {
535:             return -ENXIO;
536:         }
537:
538:         if(!blksize) {
539:             if(!(d = get_device(BLK_DEV, dev))) {
540:                 return -EINVAL;
541:             }
542:             blksize = d->blksize;
543:         }
544:         blksize = blksize ? blksize : BLKSIZE_1K;
545:
546:         lock_resource(&floppy_resource);
547:         set_current_fdd_type(minor);
548:
549:         if(fdc_block2chs(block, blksize, &cyl, &head, &sector)) {
550:             printk("WARNING: %s(): fd%d: invalid block number %d on %s device
e %d,%d.\n", __FUNCTION__, current_fdd, block, floppy_device.name, MAJOR(dev), MINOR(de
v));
551:             unlock_resource(&floppy_resource);
552:             return -EINVAL;
553:         }
554:
555:         for(retries = 0; retries < MAX_FDC_ERR; retries++) {
556:             if(need_reset) {
557:                 fdc_reset();
558:             }
559:             if(fdc_motor_on()) {
560:                 printk("%s(): %s disk was changed in device %d,%d!\n", _
__FUNCTION__, floppy_device.name, MAJOR(dev), MINOR(dev));
561:                 invalidate_buffers(dev);
562:                 fdd_status[current_fdd].recalibrated = 0;
563:             }
564:
565:             if(fdc_seek(cyl, head)) {
566:                 printk("WARNING: %s(): fd%d: seek error on %s device %d,
%d during read operation.\n", __FUNCTION__, current_fdd, floppy_device.name, MAJOR(dev)
, MINOR(dev));
567:                 continue;
568:             }
569:
570:             start_dma(FLOPPY_DMA, fdc_transfer_area, blksize, DMA_MODE_WRITE
| DMA_MODE_SINGLE);
571:
572:             /* send READ command */
573:             fdc_wait_interrupt = FDC_READ;
574:             fdc_out(FDC_READ);
575:             fdc_out((head << 2) | current_fdd);
576:             fdc_out(cyl);
577:             fdc_out(head);
578:             fdc_out(sector);
579:             fdc_out(2); /* sector size is 512 bytes */
580:             fdc_out(current_fdd_type->spt);
581:             fdc_out(current_fdd_type->gpl1);

```

drivers/block/floppy.c

Page 10/15

```

582:         fdc_out(0xFF); /* sector size is 512 bytes */
583:
584:         if(need_reset) {
585:             printk("WARNING: %s(): fd%d: needs reset on %s device %d
,%d during read operation.\n", __FUNCTION__, current_fdd, floppy_device.name, MAJOR(dev
), MINOR(dev));
586:             continue;
587:         }
588:         creq.fn = fdc_timer;
589:         creq.arg = FDC_TR_DEFAULT;
590:         add_callout(&creq, WAIT_FDC);
591:         /* avoid sleep if interrupt already happened */
592:         if(fdc_wait_interrupt) {
593:             sleep(&irq_floppy, PROC_UNINTERRUPTIBLE);
594:         }
595:         if(fdc_timeout) {
596:             need_reset = 1;
597:             printk("WARNING: %s(): fd%d: timeout on %s device %d,%d.
\n", __FUNCTION__, current_fdd, floppy_device.name, MAJOR(dev), MINOR(dev));
598:             continue;
599:         }
600:         del_callout(&creq);
601:         fdc_get_results();
602:         if(fdc_results[ST0] & (ST0_IC | ST0_UC | ST0_NR)) {
603:             need_reset = 1;
604:             continue;
605:         }
606:         break;
607:     }
608:
609:     if(retries >= MAX_FDC_ERR) {
610:         printk("WARNING: %s(): fd%d: error on %s device %d,%d during rea
d operation,\n", __FUNCTION__, current_fdd, floppy_device.name, MAJOR(dev), MINOR(dev))
;
611:         printk("\tblock=%d, sector=%d, cylinder/head=%d/%d\n", block, se
ctor, cyl, head);
612:         unlock_resource(&floppy_resource);
613:         fdc_motor_off();
614:         return -EIO;
615:     }
616:
617:     fdc_motor_off();
618:     sectors_read = (fdc_results[ST_CYL] - cyl) * (current_fdd_type->heads *
current_fdd_type->spt);
619:     sectors_read += (fdc_results[ST_HEAD] - head) * current_fdd_type->spt;
620:     sectors_read += fdc_results[ST_SECTOR] - sector;
621:     if(sectors_read * BPS != blksize) {
622:         printk("WARNING: %s(): fd%d: read error on %s device %d,%d (%d s
ectors read).\n", __FUNCTION__, current_fdd, floppy_device.name, MAJOR(dev), MINOR(dev)
, sectors_read);
623:         printk("\tblock=%d, sector=%d, cylinder/head=%d/%d\n", block, se
ctor, cyl, head);
624:         unlock_resource(&floppy_resource);
625:         fdc_motor_off();
626:         return -EIO;
627:     }
628:
629:     memcpy_b(buffer, (void *)fdc_transfer_area, blksize);
630:
631:     unlock_resource(&floppy_resource);
632:     return sectors_read * BPS;
633: }
634:
635: int fdc_write(__dev_t dev, __blk_t block, char *buffer, int blksize)
636: {
637:     unsigned char minor;
638:     unsigned int sectors_written;

```


drivers/block/floppy.c

Page 11/15

```

639:         int cyl, head, sector;
640:         int retries;
641:         struct callout_req creq;
642:         struct device *d;
643:
644:         minor = MINOR(dev);
645:         if(!TEST_MINOR(floppy_device.minors, minor)) {
646:             return -ENXIO;
647:         }
648:
649:         if(!blksize) {
650:             if(!(d = get_device(BLK_DEV, dev))) {
651:                 return -EINVAL;
652:             }
653:             blksize = d->blksize;
654:         }
655:         blksize = blksize ? blksize : BLKSIZE_1K;
656:
657:         lock_resource(&floppy_resource);
658:         set_current_fdd_type(minor);
659:
660:         if(fdc_block2chs(block, blksize, &cyl, &head, &sector)) {
661:             printk("WARNING: %s(): fd%d: invalid block number %d on %s device
e %d,%d.\n", __FUNCTION__, current_fdd, block, floppy_device.name, MAJOR(dev), MINOR(de
v));
662:             unlock_resource(&floppy_resource);
663:             return -EINVAL;
664:         }
665:
666:         for(retries = 0; retries < MAX_FDC_ERR; retries++) {
667:             if(need_reset) {
668:                 fdc_reset();
669:             }
670:             if(fdc_motor_on()) {
671:                 printk("%s(): %s disk was changed in device %d,%d!\n", _
_FUNCTION__, floppy_device.name, MAJOR(dev), MINOR(dev));
672:                 invalidate_buffers(dev);
673:                 fdd_status[current_fdd].recalibrated = 0;
674:             }
675:
676:             if(fdc_seek(cyl, head)) {
677:                 printk("WARNING: %s(): fd%d: seek error on %s device %d,
%d during write operation.\n", __FUNCTION__, current_fdd, floppy_device.name, MAJOR(dev
), MINOR(dev));
678:                 continue;
679:             }
680:
681:             start_dma(FLOPPY_DMA, fdc_transfer_area, blksize, DMA_MODE_READ
| DMA_MODE_SINGLE);
682:             memcpy_b((void *)fdc_transfer_area, buffer, blksize);
683:
684:             /* send WRITE command */
685:             fdc_wait_interrupt = FDC_WRITE;
686:             fdc_out(FDC_WRITE);
687:             fdc_out((head << 2) | current_fdd);
688:             fdc_out(cyl);
689:             fdc_out(head);
690:             fdc_out(sector);
691:             fdc_out(2); /* sector size is 512 bytes */
692:             fdc_out(current_fdd_type->spt);
693:             fdc_out(current_fdd_type->gpl1);
694:             fdc_out(0xFF); /* sector size is 512 bytes */
695:
696:             if(need_reset) {
697:                 printk("WARNING: %s(): fd%d: needs reset on %s device %d
,%d during write operation.\n", __FUNCTION__, current_fdd, floppy_device.name, MAJOR(de
v), MINOR(dev));

```

drivers/block/floppy.c

Page 12/15

```

698:             continue;
699:         }
700:         creq.fn = fdc_timer;
701:         creq.arg = FDC_TR_DEFAULT;
702:         add_callout(&creq, WAIT_FDC);
703:         /* avoid sleep if interrupt already happened */
704:         if(fdc_wait_interrupt) {
705:             sleep(&irq_floppy, PROC_UNINTERRUPTIBLE);
706:         }
707:         if(fdc_timeout) {
708:             need_reset = 1;
709:             printk("WARNING: %s(): fd%d: timeout on %s device %d,%d.
\n", __FUNCTION__, current_fdd, floppy_device.name, MAJOR(dev), MINOR(dev));
710:             continue;
711:         }
712:         del_callout(&creq);
713:         fdc_get_results();
714:         if(fdc_results[ST1] & ST1_NW) {
715:             unlock_resource(&floppy_resource);
716:             fdc_motor_off();
717:             return -EROFS;
718:         }
719:         if(fdc_results[ST0] & (ST0_IC | ST0_UC | ST0_NR)) {
720:             need_reset = 1;
721:             continue;
722:         }
723:         break;
724:     }
725:
726:     if(retries >= MAX_FDC_ERR) {
727:         printk("WARNING: %s(): fd%d: error on %s device %d,%d during wri
te operation,\n", __FUNCTION__, current_fdd, floppy_device.name, MAJOR(dev), MINOR(dev)
);
728:         printk("\tblock=%d, sector=%d, cylinder/head=%d/%d\n", block, se
ctor, cyl, head);
729:         unlock_resource(&floppy_resource);
730:         fdc_motor_off();
731:         return -EIO;
732:     }
733:
734:     fdc_motor_off();
735:     sectors_written = (fdc_results[ST_CYL] - cyl) * (current_fdd_type->heads
* current_fdd_type->spt);
736:     sectors_written += (fdc_results[ST_HEAD] - head) * current_fdd_type->spt
;
737:     sectors_written += fdc_results[ST_SECTOR] - sector;
738:     if(sectors_written * BPS != blksize) {
739:         printk("WARNING: %s(): fd%d: write error on %s device %d,%d (%d
sectors written).\n", __FUNCTION__, current_fdd, floppy_device.name, MAJOR(dev), MINOR(
dev), sectors_written);
740:         printk("\tblock=%d, sector=%d, cylinder/head=%d/%d\n", block, se
ctor, cyl, head);
741:         unlock_resource(&floppy_resource);
742:         fdc_motor_off();
743:         return -EIO;
744:     }
745:
746:     unlock_resource(&floppy_resource);
747:     return sectors_written * BPS;
748: }
749:
750: int fdc_ioctl(struct inode *i, int cmd, unsigned long int arg)
751: {
752:     unsigned char minor;
753:     struct hd_geometry *geom;
754:     int errno;
755:

```

drivers/block/floppy.c

Page 13/15

```

756:     minor = MINOR(i->rdev);
757:     if(!TEST_MINOR(floppy_device.minors, minor)) {
758:         return -ENXIO;
759:     }
760:
761:     lock_resource(&floppy_resource);
762:     set_current_fdd_type(minor);
763:     unlock_resource(&floppy_resource);
764:
765:     switch(cmd) {
766:         case HDIO_GETGEO:
767:             if((errno = check_user_area(VERIFY_WRITE, (void *)arg, s
sizeof(struct hd_geometry)))) {
768:                 return errno;
769:             }
770:             geom = (struct hd_geometry *)arg;
771:             geom->heads = current_fdd_type->heads;
772:             geom->sectors = current_fdd_type->spt;
773:             geom->cylinders = current_fdd_type->tracks;
774:             geom->start = 0;
775:             break;
776:         case BLKRRPART:
777:             break;
778:         case BLKGETSIZE:
779:             if((errno = check_user_area(VERIFY_WRITE, (void *)arg, s
sizeof(unsigned int)))) {
780:                 return errno;
781:             }
782:             *(int *)arg = fdd_sizes[MINOR(i->rdev)] * 2;
783:             break;
784:         default:
785:             return -EINVAL;
786:     }
787:     return 0;
788: }
789:
790: int fdc_lseek(struct inode *i, __off_t offset)
791: {
792:     unsigned char minor;
793:
794:     minor = MINOR(i->rdev);
795:     if(!TEST_MINOR(floppy_device.minors, minor)) {
796:         return -ENXIO;
797:     }
798:
799:     lock_resource(&floppy_resource);
800:     set_current_fdd_type(minor);
801:     unlock_resource(&floppy_resource);
802:
803:     return offset;
804: }
805:
806: void floppy_init(void)
807: {
808:     short int cmosval, master, slave;
809:
810:     cmosval = cmos_read(CMOS_FDDTYPE);
811:     set_current_fdd_type(0);          /* sets /dev/fd0 by default */
812:
813:     /* the high nibble describes the 'master' floppy drive */
814:     master = cmosval >> 4;
815:
816:     /*
817:      * Some BIOS may return the value 0x05 (for 2.88MB floppy type) which is
818:      * not supported by Fiwix. This prevents from using an unexistent type
819:      * in the fdd_type structure if this happens.
820:      */

```

drivers/block/floppy.c

Page 14/15

```

821:         if(master > 4) {
822:             master = 4;
823:         }
824:
825:         if(master) {
826:             if(!register_irq(FLOPPY_IRQ, &irq_config_floppy)) {
827:                 enable_irq(FLOPPY_IRQ);
828:             }
829:             printk("fd0          0x%04X-0x%04X  %2d      ", FDC_SRA, FDC_CCR, FL
OPPY_IRQ);
830:             printk("%s ", fdd_type[master].name);
831:             fdd_status[0].type = fdd_status[1].type = master;
832:             SET_MINOR(floppy_device.minors, 0);
833:             SET_MINOR(floppy_device.minors, 4);
834:             SET_MINOR(floppy_device.minors, 8);
835:             SET_MINOR(floppy_device.minors, 12);
836:             SET_MINOR(floppy_device.minors, 16);
837:             fdd_sizes[0] = fdd_type[master].sizekb;
838:             fdd_sizes[4] = fdd_type[1].sizekb;
839:             fdd_sizes[8] = fdd_type[2].sizekb;
840:             fdd_sizes[12] = fdd_type[3].sizekb;
841:             fdd_sizes[16] = fdd_type[4].sizekb;
842:             fdc_reset();
843:             fdc_get_chip();
844:         }
845:
846:         /* the low nibble is for the 'slave' floppy drive */
847:         slave = cmosval & 0x0F;
848:         if(slave) {
849:             if(!master) {
850:                 if(!register_irq(FLOPPY_IRQ, &irq_config_floppy)) {
851:                     enable_irq(FLOPPY_IRQ);
852:                 }
853:             }
854:             printk("fd1          0x%04X-0x%04X  %2d      ", FDC_SRA, FDC_CCR, FL
OPPY_IRQ);
855:             printk("%s ", fdd_type[slave].name);
856:             fdd_status[1].type = slave;
857:             SET_MINOR(floppy_device.minors, 1);
858:             SET_MINOR(floppy_device.minors, 5);
859:             SET_MINOR(floppy_device.minors, 9);
860:             SET_MINOR(floppy_device.minors, 13);
861:             SET_MINOR(floppy_device.minors, 17);
862:             fdd_sizes[1] = fdd_type[slave].sizekb;
863:             fdd_sizes[5] = fdd_type[1].sizekb;
864:             fdd_sizes[9] = fdd_type[2].sizekb;
865:             fdd_sizes[13] = fdd_type[3].sizekb;
866:             fdd_sizes[17] = fdd_type[4].sizekb;
867:             if(!master) {
868:                 fdc_get_chip();
869:             } else {
870:                 printk("\n");
871:             }
872:         }
873:
874:         if(master || slave) {
875:             need_reset = 1;
876:             dma_init();
877:             if(dma_register(FLOPPY_DMA, floppy_device.name)) {
878:                 printk("WARNING: %s(): fd%d: unable to register DMA chan
nel on %s.\n", __FUNCTION__, current_fdd, floppy_device.name);
879:             } else {
880:                 if(!register_device(BLK_DEV, &floppy_device)) {
881:                     do_motor_off(current_fdd);
882:                 }
883:             }
884:         }

```

```
885: }
```

drivers/block/ide.c

Page 1/15

```

1: /*
2:  * fiwix/drivers/block/ide.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/ide.h>
10: #include <fiwix/ide_hd.h>
11: #include <fiwix/ide_cd.h>
12: #include <fiwix/devices.h>
13: #include <fiwix/sleep.h>
14: #include <fiwix/timer.h>
15: #include <fiwix/sched.h>
16: #include <fiwix/cpu.h>
17: #include <fiwix/pic.h>
18: #include <fiwix/fs.h>
19: #include <fiwix/mm.h>
20: #include <fiwix/errno.h>
21: #include <fiwix/stdio.h>
22: #include <fiwix/string.h>
23:
24: int ide0_need_reset = 0;
25: int ide0_wait_interrupt = 0;
26: int ide0_timeout = 0;
27: int ide1_need_reset = 0;
28: int ide1_wait_interrupt = 0;
29: int ide1_timeout = 0;
30:
31: struct ide ide_table[NR_IDE_CTRL] = {
32:     { IDE_PRIMARY, IDE0_BASE, IDE0_CTRL, IDE0_IRQ, -1, { NULL, NULL },
33:       {
34:           { IDE_MASTER, "hda", IDE0_MAJOR, 0, IDE_MASTER_MSF, NULL
, NULL, NULL, NULL, NULL, { NULL }, {{ NULL }} },
35:           { IDE_SLAVE, "hdb", IDE0_MAJOR, 0, IDE_SLAVE_MSF, NULL,
NULL, NULL, NULL, NULL, { NULL }, {{ NULL }} }
36:       }
37:     },
38:     { IDE_SECONDARY, IDE1_BASE, IDE1_CTRL, IDE1_IRQ, -1, {NULL, NULL },
39:       {
40:           { IDE_MASTER, "hdc", IDE1_MAJOR, 0, IDE_MASTER_MSF, NULL
, NULL, NULL, NULL, NULL, { NULL }, {{ NULL }} },
41:           { IDE_SLAVE, "hdd", IDE1_MAJOR, 0, IDE_SLAVE_MSF, NULL,
NULL, NULL, NULL, NULL, { NULL }, {{ NULL }} }
42:       }
43:     }
44: };
45:
46: static char *ide_ctrl_name[] = { "primary", "secondary" };
47: static char *ide_drv_name[] = { "master", "slave" };
48:
49: static unsigned int ide0_sizes[256];
50: static unsigned int ide1_sizes[256];
51:
52: static struct fs_operations ide_driver_fsop = {
53:     0,
54:     0,
55:
56:     ide_open,
57:     ide_close,
58:     NULL, /* read */
59:     NULL, /* write */
60:     ide_ioctl,
61:     NULL, /* lseek */
62:     NULL, /* readdir */
63:     NULL, /* mmap */

```

drivers/block/ide.c

Page 2/15

```

64:         NULL,                /* select */
65:
66:         NULL,                /* readlink */
67:         NULL,                /* followlink */
68:         NULL,                /* bmap */
69:         NULL,                /* lockup */
70:         NULL,                /* rmdir */
71:         NULL,                /* link */
72:         NULL,                /* unlink */
73:         NULL,                /* symlink */
74:         NULL,                /* mkdir */
75:         NULL,                /* mknod */
76:         NULL,                /* truncate */
77:         NULL,                /* create */
78:         NULL,                /* rename */
79:
80:         ide_read,
81:         ide_write,
82:
83:         NULL,                /* read_inode */
84:         NULL,                /* write_inode */
85:         NULL,                /* ialloc */
86:         NULL,                /* ifree */
87:         NULL,                /* statfs */
88:         NULL,                /* read_superblock */
89:         NULL,                /* remount_fs */
90:         NULL,                /* write_superblock */
91:         NULL,                /* release_superblock */
92: };
93:
94: static struct device ide0_device = {
95:     "ide0",
96:     IDE0_MAJOR,
97:     { 0, 0, 0, 0, 0, 0, 0, 0 },
98:     0,
99:     &ide0_sizes,
100:    &ide_driver_fsop,
101:    NULL
102: };
103:
104: static struct device idel_device = {
105:     "idel",
106:     IDE1_MAJOR,
107:     { 0, 0, 0, 0, 0, 0, 0, 0 },
108:     0,
109:     &idel_sizes,
110:     &ide_driver_fsop,
111:     NULL
112: };
113:
114: static struct interrupt irq_config_ide0 = { 0, "ide0", &irq_ide0, NULL };
115: static struct interrupt irq_config_idel = { 0, "idel", &irq_idel, NULL };
116:
117: static int ide_identify(struct ide *ide, int drive)
118: {
119:     short int status, *buffer;
120:     struct callout_req creq;
121:
122:     if((status = ide_drvsel(ide, drive, IDE_CHS_MODE, 0))) {
123:         /* some controllers return 0xFF to indicate a non-drive conditio
n */
124:         if(status == 0xFF) {
125:             return status;
126:         }
127:         printk("WARNING: %s(): error on device '%s'.\n", __FUNCTION__, i
de->drive[drive].dev_name);
128:         ide_error(ide, status);

```

drivers/block/ide.c

Page 3/15

```

129:         return status;
130:     }
131:
132:     if(ide->channel == IDE_PRIMARY) {
133:         ide0_wait_interrupt = ide->base;
134:         creq.fn = ide0_timer;
135:         creq.arg = 0;
136:         add_callout(&creq, WAIT_FOR_IDE);
137:         outport_b(ide->base + IDE_COMMAND, (ide->drive[drive].flags & DE
VICE_IS_ATAPI) ? ATA_IDENTIFY_PACKET : ATA_IDENTIFY);
138:         if(ide0_wait_interrupt) {
139:             sleep(&irq_ide0, PROC_UNINTERRUPTIBLE);
140:         }
141:         if(ide0_timeout) {
142:             status = inport_b(ide->base + IDE_STATUS);
143:             if((status & (IDE_STAT_RDY | IDE_STAT_DRQ)) != (IDE_STAT
_RDY | IDE_STAT_DRQ)) {
144:                 return 1;
145:             }
146:         }
147:         del_callout(&creq);
148:     }
149:     if(ide->channel == IDE_SECONDARY) {
150:         idel_wait_interrupt = ide->base;
151:         creq.fn = idel_timer;
152:         creq.arg = 0;
153:         add_callout(&creq, WAIT_FOR_IDE);
154:         outport_b(ide->base + IDE_COMMAND, (ide->drive[drive].flags & DE
VICE_IS_ATAPI) ? ATA_IDENTIFY_PACKET : ATA_IDENTIFY);
155:         if(idel_wait_interrupt) {
156:             sleep(&irq_idel, PROC_UNINTERRUPTIBLE);
157:         }
158:         if(idel_timeout) {
159:             status = inport_b(ide->base + IDE_STATUS);
160:             if((status & (IDE_STAT_RDY | IDE_STAT_DRQ)) != (IDE_STAT
_RDY | IDE_STAT_DRQ)) {
161:                 return 1;
162:             }
163:         }
164:         del_callout(&creq);
165:     }
166:
167:     status = inport_b(ide->base + IDE_STATUS);
168:     if((status & (IDE_STAT_RDY | IDE_STAT_DRQ)) != (IDE_STAT_RDY | IDE_STAT
DRQ)) {
169:         return 1;
170:     }
171:
172:     if(!(buffer = (void *)kmalloc())) {
173:         return 1;
174:     }
175:
176:     inport_sw(ide->base + IDE_DATA, (void *)buffer, IDE_HD_SECTSIZE / sizeof
(short int));
177:     memcpy_b(&ide->drive[drive].ident, (void *)buffer, sizeof(struct ide_drv
_ident));
178:     kfree((unsigned int)buffer);
179:
180:     /* some basic checks to make sure that data received makes sense */
181:     if(ide->drive[drive].ident.logic_cyls > 0xF000 &&
182:         ide->drive[drive].ident.logic_heads > 0xF000 &&
183:         ide->drive[drive].ident.logic_spt > 0xF000 &&
184:         ide->drive[drive].ident.buffer_cache > 0xF000
185:     ) {
186:         memset_b(&ide->drive[drive].ident, 0, sizeof(struct ide_drv_iden
t));
187:         return 1;

```


drivers/block/ide.c

Page 4/15

```

188:         }
189:
190:         if(ide->drive[drive].ident.gen_config == IDE_SUPPORTS_CFA) {
191:             ide->drive[drive].flags |= DEVICE_IS_CFA;
192:         }
193:
194:         if(ide->drive[drive].flags & DEVICE_IS_ATAPI) {
195:             if((ide->drive[drive].ident.gen_config >> 8) & 0x1F) == ATAPI_I
S_CDROM) {
196:                 ide->drive[drive].flags |= DEVICE_IS_CDROM;
197:             }
198:             if(ide->drive[drive].ident.gen_config & 0x3) {
199:                 printk("WARNING: %s(): packet size must be 16 bytes!\n")
;
200:             }
201:         }
202:
203:         /* only bits 0-7 are relevant */
204:         ide->drive[drive].ident.rw_multiple &= 0xFF;
205:         return 0;
206:     }
207:
208:     static void get_device_size(struct ide_drv *drive)
209:     {
210:         if(drive->ident.capabilities & IDE_HAS_LBA) {
211:             drive->lba_cyls = drive->ident.logic_cyls;
212:             drive->lba_heads = drive->ident.logic_heads;
213:             drive->lba_factor = 0;
214:
215:             while(drive->lba_cyls > 1023) {
216:                 if(drive->lba_heads < 255) {
217:                     drive->lba_cyls >>= 1;
218:                     drive->lba_heads <<= 1;
219:                 } else {
220:                     break;
221:                 }
222:                 drive->lba_factor++;
223:             }
224:             drive->nr_sects = drive->ident.tot_sectors | (drive->ident.tot_s
ectors2 << 16);
225:         }
226:
227:         /* some old disk drives (ATA or ATA2) don't specify total sectors */
228:         if(!(drive->ident.capabilities & IDE_HAS_LBA)) {
229:             if(drive->nr_sects == 0) {
230:                 drive->nr_sects = drive->ident.logic_cyls * drive->ident
.logic_heads * drive->ident.logic_spt;
231:             }
232:         }
233:
234:     }
235:
236:     static int get_udma(struct ide *ide, int drive)
237:     {
238:         int udma;
239:
240:         if(ide->drive[drive].ident.fields_validity & IDE_HAS_UDMA) {
241:             if((ide->drive[drive].ident.ultradma >> 13) & 1) {
242:                 udma = 5;
243:             } else if((ide->drive[drive].ident.ultradma >> 12) & 1) {
244:                 udma = 4;
245:             } else if((ide->drive[drive].ident.ultradma >> 11) & 1) {
246:                 udma = 3;
247:             } else if((ide->drive[drive].ident.ultradma >> 10) & 1) {
248:                 udma = 2;
249:             } else if((ide->drive[drive].ident.ultradma >> 9) & 1) {
250:                 udma = 1;

```

drivers/block/ide.c

Page 5/15

```

251:         } else {
252:             udma = 0;
253:         }
254:     } else {
255:         udma = -1;
256:     }
257:     return udma;
258: }
259:
260: static void ide_results(struct ide *ide, int drive)
261: {
262:     unsigned int cyl, hds, sect;
263:     __loff_t capacity;
264:     int udma;
265:     int udma_speed[] = { 16, 25, 33, 44, 66, 100 };
266:
267:     cyl = ide->drive[drive].ident.logic_cyls;
268:     hds = ide->drive[drive].ident.logic_heads;
269:     sect = ide->drive[drive].ident.logic_spt;
270:
271:     udma = get_udma(ide, drive);
272:     /*
273:      * After knowing if the device is UDMA capable we could choose between
274:      * the PIO transfer mode or the UDMA transfer mode.
275:      * FIXME: Currently only PIO mode is supported.
276:      */
277:
278:     capacity = (__loff_t)ide->drive[drive].nr_sects * BPS;
279:     capacity = capacity / 1024 / 1024;
280:
281:     printk("%s      0x%04X-0x%04X   %2d   ", ide->drive[drive].dev_name, i
de->base, ide->base + IDE_BASE_LEN, ide->irq);
282:     swap_asc_word(ide->drive[drive].ident.model_number, 40);
283:     printk("%s %s ", ide_ctrl_name[ide->channel], ide_drv_name[ide->drive[dr
ive].drive]);
284:
285:     if(!(ide->drive[drive].flags & DEVICE_IS_ATAPI)) {
286:         printk("ATA");
287:     } else {
288:         printk("ATAPI");
289:     }
290:
291:     if(ide->drive[drive].flags & DEVICE_IS_CFA) {
292:         printk(" CFA");
293:     }
294:
295:     if(ide->drive[drive].flags & DEVICE_IS_DISK) {
296:         printk(" DISK drive %dMB\n", (unsigned int)capacity);
297:         printk("                                model=%s\n", ide->drive[
drive].ident.model_number);
298:         if(ide->drive[drive].nr_sects < IDE_MIN_LBA) {
299:             printk("                                CHS=%d/%d/%d", c
yl, hds, sect);
300:         } else {
301:             ide->drive[drive].flags |= DEVICE_REQUIRES_LBA;
302:             printk("                                sectors=%d", ide
->drive[drive].nr_sects);
303:         }
304:         printk(" cache=%dKB", ide->drive[drive].ident.buffer_cache >> 1)
;
305:     }
306:
307:     if(ide->drive[drive].flags & DEVICE_IS_CDROM) {
308:         printk(" CDROM drive\n");
309:         printk("                                model=%s\n", ide->drive[
drive].ident.model_number);
310:         printk("                                cache=%dKB", ide->drive[

```

drivers/block/ide.c

Page 6/15

```

drive].ident.buffer_cache >> 1);
311:     }
312:
313:     if(udma >= 0) {
314:         printk(" UDMA%d(%d) ", udma, udma_speed[udma]);
315:     }
316:     if(ide->drive[drive].ident.capabilities & IDE_HAS_LBA) {
317:         ide->drive[drive].flags |= DEVICE_REQUIRES_LBA;
318:         printk(" LBA");
319:     }
320:
321:     printk("\n");
322:
323:     if(ide->drive[drive].ident.rw_multiple > 1) {
324:         ide->drive[drive].flags |= DEVICE_HAS_RW_MULTIPLE;
325:     }
326:
327:     /*
328:     printk("\n");
329:     printk("%s -> %s\n", ide->drive[drive].dev_name, ide->drive[drive].flags
& DEVICE_IS_ATAPI ? "ATAPI" : "ATA");
330:     printk("general conf = %d (%b) (0x%x)\n", ide->drive[drive].ident.gen_c
onfig, ide->drive[drive].ident.gen_config, ide->drive[drive].ident.gen_config);
331:     printk("logic_cyls = %d (%b)\n", ide->drive[drive].ident.logic_cyls,
ide->drive[drive].ident.logic_cyls);
332:     printk("reserved2 = %d (%b)\n", ide->drive[drive].ident.reserved2, i
de->drive[drive].ident.reserved2);
333:     printk("logic_heads = %d (%b)\n", ide->drive[drive].ident.logic_heads,
ide->drive[drive].ident.logic_heads);
334:     printk("retired4 = %d (%b)\n", ide->drive[drive].ident.retired4, id
e->drive[drive].ident.retired4);
335:     printk("retired5 = %d (%b)\n", ide->drive[drive].ident.retired5, id
e->drive[drive].ident.retired5);
336:     printk("logic_spt = %d (%b)\n", ide->drive[drive].ident.logic_spt, i
de->drive[drive].ident.logic_spt);
337:     printk("retired7 = %d (%b)\n", ide->drive[drive].ident.retired7, id
e->drive[drive].ident.retired7);
338:     printk("retired8 = %d (%b)\n", ide->drive[drive].ident.retired8, id
e->drive[drive].ident.retired8);
339:     printk("retired9 = %d (%b)\n", ide->drive[drive].ident.retired9, id
e->drive[drive].ident.retired9);
340:     printk("serial number = '%s'\n", ide->drive[drive].ident.serial_number);
341:     printk("vendor spec20 = %d (%b)\n", ide->drive[drive].ident.vendor_spec2
0, ide->drive[drive].ident.vendor_spec20);
342:     printk("buffer cache = %d (%b)\n", ide->drive[drive].ident.buffer_cache
, ide->drive[drive].ident.buffer_cache);
343:     printk("vendor spec22 = %d (%b)\n", ide->drive[drive].ident.vendor_spec2
2, ide->drive[drive].ident.vendor_spec22);
344:     printk("firmware rev = '%s'\n", ide->drive[drive].ident.firmware_rev);
345:     printk("model number = '%s'\n", ide->drive[drive].ident.model_number);
346:     printk("rw multiple = %d (%b)\n", ide->drive[drive].ident.rw_multiple,
ide->drive[drive].ident.rw_multiple);
347:     printk("reserved48 = %d (%b)\n", ide->drive[drive].ident.reserved48,
ide->drive[drive].ident.reserved48);
348:     printk("capabilities = %d (%b)\n", ide->drive[drive].ident.capabilities
, ide->drive[drive].ident.capabilities);
349:     printk("reserved50 = %d (%b)\n", ide->drive[drive].ident.reserved50,
ide->drive[drive].ident.reserved50);
350:     printk("pio mode = %d (%b)\n", ide->drive[drive].ident.pio_mode, id
e->drive[drive].ident.pio_mode);
351:     printk("dma mode = %d (%b)\n", ide->drive[drive].ident.dma_mode, id
e->drive[drive].ident.dma_mode);
352:     printk("fields validi = %d (%b)\n", ide->drive[drive].ident.fields_valid
ity, ide->drive[drive].ident.fields_validity);
353:     printk("cur log cyls = %d (%b)\n", ide->drive[drive].ident.cur_log_cyls
, ide->drive[drive].ident.cur_log_cyls);
354:     printk("cur log heads = %d (%b)\n", ide->drive[drive].ident.cur_log_head

```

drivers/block/ide.c

Page 7/15

```

s, ide->drive[drive].ident.cur_log_heads);
355:     printk("cur log spt   = %d (%b)\n", ide->drive[drive].ident.cur_log_spt,
ide->drive[drive].ident.cur_log_spt);
356:     printk("cur capacity = %d (%b)\n", ide->drive[drive].ident.cur_capacity
| (ide->drive[drive].ident.cur_capacity2 << 16), ide->drive[drive].ident.cur_capacity
| (ide->drive[drive].ident.cur_capacity2 << 16));
357:     printk("mult sect set = %d (%b)\n", ide->drive[drive].ident.mult_sect_se
t, ide->drive[drive].ident.mult_sect_set);
358:     printk("tot sectors   = %d (%b)\n", ide->drive[drive].ident.tot_sectors
| (ide->drive[drive].ident.tot_sectors2 << 16), ide->drive[drive].ident.tot_sectors | (
ide->drive[drive].ident.tot_sectors2 << 16));
359:     printk("singleword dma= %d (%b)\n", ide->drive[drive].ident.singleword_d
ma, ide->drive[drive].ident.singleword_dma);
360:     printk("multiword dma = %d (%b)\n", ide->drive[drive].ident.multiword_dm
a, ide->drive[drive].ident.multiword_dma);
361:     printk("adv pio modes = %d (%b)\n", ide->drive[drive].ident.adv_pio_mode
s, ide->drive[drive].ident.adv_pio_modes);
362:     printk("min multiword = %d (%b)\n", ide->drive[drive].ident.min_multiwor
d, ide->drive[drive].ident.min_multiword);
363:     printk("rec multiword = %d (%b)\n", ide->drive[drive].ident.rec_multiwor
d, ide->drive[drive].ident.rec_multiword);
364:     printk("min pio wo fc = %d (%b)\n", ide->drive[drive].ident.min_pio_wo_fc
, ide->drive[drive].ident.min_pio_wo_fc);
365:     printk("min pio w fc  = %d (%b)\n", ide->drive[drive].ident.min_pio_w_fc
, ide->drive[drive].ident.min_pio_w_fc);
366:     printk("reserved69   = %d (%b)\n", ide->drive[drive].ident.reserved69,
ide->drive[drive].ident.reserved69);
367:     printk("reserved70   = %d (%b)\n", ide->drive[drive].ident.reserved70,
ide->drive[drive].ident.reserved70);
368:     printk("reserved71   = %d (%b)\n", ide->drive[drive].ident.reserved71,
ide->drive[drive].ident.reserved71);
369:     printk("reserved72   = %d (%b)\n", ide->drive[drive].ident.reserved72,
ide->drive[drive].ident.reserved72);
370:     printk("reserved73   = %d (%b)\n", ide->drive[drive].ident.reserved73,
ide->drive[drive].ident.reserved73);
371:     printk("reserved74   = %d (%b)\n", ide->drive[drive].ident.reserved74,
ide->drive[drive].ident.reserved74);
372:     printk("queue depth  = %d (%b)\n", ide->drive[drive].ident.queue_depth,
ide->drive[drive].ident.queue_depth);
373:     printk("reserved76   = %d (%b)\n", ide->drive[drive].ident.reserved76,
ide->drive[drive].ident.reserved76);
374:     printk("reserved77   = %d (%b)\n", ide->drive[drive].ident.reserved77,
ide->drive[drive].ident.reserved77);
375:     printk("reserved78   = %d (%b)\n", ide->drive[drive].ident.reserved78,
ide->drive[drive].ident.reserved78);
376:     printk("reserved79   = %d (%b)\n", ide->drive[drive].ident.reserved79,
ide->drive[drive].ident.reserved79);
377:     printk("major version = %d (%b)\n", ide->drive[drive].ident.majorver, id
e->drive[drive].ident.majorver);
378:     printk("minor version = %d (%b)\n", ide->drive[drive].ident.minorver, id
e->drive[drive].ident.minorver);
379:     printk("cmdset1     = %d (%b)\n", ide->drive[drive].ident.cmdset1, ide
->drive[drive].ident.cmdset1);
380:     printk("cmdset2     = %d (%b)\n", ide->drive[drive].ident.cmdset2, ide
->drive[drive].ident.cmdset2);
381:     printk("cmdsfs ext   = %d (%b)\n", ide->drive[drive].ident.cmdsf_ext, i
de->drive[drive].ident.cmdsf_ext);
382:     printk("cmdsfs enable1 = %d (%b)\n", ide->drive[drive].ident.cmdsf_enable
1, ide->drive[drive].ident.cmdsf_enable1);
383:     printk("cmdsfs enable2 = %d (%b)\n", ide->drive[drive].ident.cmdsf_enable
2, ide->drive[drive].ident.cmdsf_enable2);
384:     printk("cmdsfs default = %d (%b)\n", ide->drive[drive].ident.cmdsf_defaul
t, ide->drive[drive].ident.cmdsf_default);
385:     printk("ultra dma    = %d (%b)\n", ide->drive[drive].ident.ultradma, id
e->drive[drive].ident.ultradma);
386:     printk("reserved89   = %d (%b)\n", ide->drive[drive].ident.reserved89,
ide->drive[drive].ident.reserved89);

```

drivers/block/ide.c

Page 8/15

```

387:         printk("reserved90    = %d (%b)\n", ide->drive[drive].ident.reserved90,
ide->drive[drive].ident.reserved90);
388:         printk("current apm     = %d (%b)\n", ide->drive[drive].ident.curapm, ide-
>drive[drive].ident.curapm);
389:         */
390:     }
391:
392:     static int ide_get_status(struct ide *ide)
393:     {
394:         int n, retries, status;
395:
396:         status = 0;
397:         SET_IDE_RDY_RETR(retries);
398:
399:         for(n = 0; n < retries; n++) {
400:             status = inport_b(ide->ctrl + IDE_ALT_STATUS);
401:             if(!(status & IDE_STAT_BSY)) {
402:                 return 0;
403:             }
404:             ide_delay();
405:         }
406:
407:         inport_b(ide->base + IDE_STATUS);          /* clear any pending interrupt */
408:         return status;
409:     }
410:
411: void irq_ide0(int num, struct sigcontext *sc)
412: {
413:     if(!ide0_wait_interrupt) {
414:         printk("WARNING: %s(): unexpected interrupt!\n", __FUNCTION__);
415:         ide0_need_reset = 1;
416:     } else {
417:         ide0_timeout = ide0_wait_interrupt = 0;
418:         wakeup(&irq_ide0);
419:     }
420: }
421:
422: void irq_idel(int num, struct sigcontext *sc)
423: {
424:     if(!idel_wait_interrupt) {
425:         printk("WARNING: %s(): unexpected interrupt!\n", __FUNCTION__);
426:         idel_need_reset = 1;
427:     } else {
428:         idel_timeout = idel_wait_interrupt = 0;
429:         wakeup(&irq_idel);
430:     }
431: }
432:
433: void ide0_timer(unsigned int arg)
434: {
435:     ide0_timeout = 1;
436:     ide0_wait_interrupt = 0;
437:     wakeup(&irq_ide0);
438: }
439:
440: void idel_timer(unsigned int arg)
441: {
442:     idel_timeout = 1;
443:     idel_wait_interrupt = 0;
444:     wakeup(&irq_idel);
445: }
446:
447: void ide_error(struct ide *ide, int status)
448: {
449:     int error;
450:

```

drivers/block/ide.c

Page 9/15

```

451:         if(status & IDE_STAT_ERR) {
452:             error = inport_b(ide->base + IDE_ERROR);
453:             if(error) {
454:                 printk("error=0x%x [", error);
455:             }
456:             if(error & IDE_ERR_AMNF) {
457:                 printk("address mark not found, ");
458:             }
459:             if(error & IDE_ERR_TKONF) {
460:                 printk("track 0 not found (no media) or media error, ");
461:             }
462:             if(error & IDE_ERR_ABRT) {
463:                 printk("command aborted, ");
464:             }
465:             if(error & IDE_ERR_MCR) {
466:                 printk("media change requested, ");
467:             }
468:             if(error & IDE_ERR_IDNF) {
469:                 printk("id mark not found, ");
470:             }
471:             if(error & IDE_ERR_MC) {
472:                 printk("media changer, ");
473:             }
474:             if(error & IDE_ERR_UNC) {
475:                 printk("uncorrectable data, ");
476:             }
477:             if(error & IDE_ERR_BBK) {
478:                 printk("bad block, ");
479:             }
480:             printk("]");
481:         }
482:         if(status & IDE_STAT_DWF) {
483:             printk("device fault, ");
484:         }
485:         if(status & IDE_STAT_BSY) {
486:             printk("device busy, ");
487:         }
488:         printk("\n");
489:     }
490:
491: void ide_delay(void)
492: {
493:     int n;
494:
495:     for(n = 0; n < 10000; n++) {
496:         NOP();
497:     }
498: }
499:
500: void ide_wait400ns(struct ide *ide)
501: {
502:     int n;
503:
504:     /* wait 400ns */
505:     for(n = 0; n < 4; n++) {
506:         inport_b(ide->ctrl + IDE_ALT_STATUS);
507:     }
508: }
509:
510: int ide_drtsel(struct ide *ide, int drive, int mode, unsigned char lba24_head)
511: {
512:     int n, status;
513:     int selected;
514:
515:     selected = (mode + (drive << 4)) | lba24_head;
516:
517:     /* just return if the drive is already setup with the same parameters */

```

drivers/block/ide.c

Page 10/15

```

518:         if(selected == ide->current) {
519:             return 0;
520:         }
521:
522:         status = 0;
523:
524:         for(n = 0; n < MAX_IDE_ERR; n++) {
525:             if((status = ide_get_status(ide))) {
526:                 continue;
527:             }
528:             break;
529:         }
530:         if(status) {
531:             return status;
532:         }
533:
534:         ide->current = selected;
535:         outport_b(ide->base + IDE_DRVHD, selected);
536:         ide_wait400ns(ide);
537:
538:         for(n = 0; n < MAX_IDE_ERR; n++) {
539:             if((status = ide_get_status(ide))) {
540:                 continue;
541:             }
542:             break;
543:         }
544:         return status;
545:     }
546:
547: int ide_softreset(struct ide *ide)
548: {
549:     int error;
550:     unsigned short int dev_type;
551:
552:     error = 0;
553:
554:     outport_b(ide->base + IDE_DRVHD, IDE_CHS_MODE);
555:     ide_delay();
556:
557:     outport_b(ide->ctrl + IDE_DEV_CTRL, IDE_DEVCTR_SRST | IDE_DEVCTR_NIEN);
558:     ide_delay();
559:     outport_b(ide->ctrl + IDE_DEV_CTRL, 0);
560:     ide_delay();
561:
562:     outport_b(ide->base + IDE_DRVHD, IDE_CHS_MODE);
563:     ide_delay();
564:     if(ide_get_status(ide)) {
565:         printk("WARNING: %s(): reset error on IDE(%d:0).\n", __FUNCTION_
_, ide->channel);
566:         error = 1;
567:     } else {
568:         /* find out the device type */
569:         if(inport_b(ide->base + IDE_SECCNT) == 1 && inport_b(ide->base +
IDE_SECNUM) == 1) {
570:             dev_type = (inport_b(ide->base + IDE_HCYL) << 8) | inpor
t_b(ide->base + IDE_LCYL);
571:             switch(dev_type) {
572:                 case 0xEB14:
573:                     ide->drive[IDE_MASTER].flags |= DEVICE_I
S_ATAPI;
574:                     break;
575:                 case 0x0:
576:                 default:
577:                     ide->drive[IDE_MASTER].flags |= DEVICE_I
S_DISK;
578:             }
579:         }

```

drivers/block/ide.c

Page 11/15

```

580:         }
581:
582:         outport_b(ide->base + IDE_DRVHD, IDE_CHS_MODE + (1 << 4));
583:         ide_delay();
584:         if(ide_get_status(ide)) {
585:             printk("WARNING: %s(): reset error on IDE(%d:1).\n", __FUNCTION_
_, ide->channel);
586:             outport_b(ide->base + IDE_DRVHD, IDE_CHS_MODE);
587:             ide_delay();
588:             ide_get_status(ide);
589:             error |= (1 << 4);
590:         }
591:
592:         outport_b(ide->ctrl + IDE_DEV_CTRL, 0);
593:         ide_delay();
594:         if(error > 1) {
595:             return error;
596:         }
597:
598:         /* find out the device type */
599:         if(inport_b(ide->base + IDE_SECCNT) == 1 && inport_b(ide->base + IDE_SEC
NUM) == 1) {
600:             dev_type = (inport_b(ide->base + IDE_HCYL) << 8) | inport_b(ide-
>base + IDE_LCYL);
601:             switch(dev_type) {
602:                 case 0xEB14:
603:                     ide->drive[IDE_SLAVE].flags |= DEVICE_IS_ATAPI;
604:                     break;
605:                 case 0x0:
606:                 default:
607:                     ide->drive[IDE_SLAVE].flags |= DEVICE_IS_DISK;
608:             }
609:         }
610:
611:         return error;
612:     }
613:
614:     struct ide * get_ide_controller(__dev_t dev)
615:     {
616:         int controller;
617:
618:         if(MAJOR(dev) == IDE0_MAJOR) {
619:             controller = IDE_PRIMARY;
620:         } else {
621:             if(MAJOR(dev) == IDE1_MAJOR) {
622:                 controller = IDE_SECONDARY;
623:             } else {
624:                 return NULL;
625:             }
626:         }
627:         return &ide_table[controller];
628:     }
629:
630:     int get_ide_drive(__dev_t dev)
631:     {
632:         int drive;
633:
634:         drive = MINOR(dev);
635:         if(drive) {
636:             if(drive & (1 << IDE_SLAVE_MSF)) {
637:                 drive = IDE_SLAVE;
638:             } else {
639:                 drive = IDE_MASTER;
640:             }
641:         }
642:         return drive;
643:     }

```


drivers/block/ide.c

Page 12/15

```

644:
645: int ide_open(struct inode *i, struct fd *fd_table)
646: {
647:     int drive;
648:     struct ide *ide;
649:
650:     if (!(ide = get_ide_controller(i->rdev))) {
651:         return -EINVAL;
652:     }
653:
654:     if (!get_device(BLK_DEV, i->rdev)) {
655:         return -ENXIO;
656:     }
657:
658:     drive = get_ide_drive(i->rdev);
659:     if (ide->drive[drive].fsop && ide->drive[drive].fsop->open) {
660:         return ide->drive[drive].fsop->open(i, fd_table);
661:     }
662:     return -EINVAL;
663: }
664:
665: int ide_close(struct inode *i, struct fd *fd_table)
666: {
667:     int drive;
668:     struct ide *ide;
669:
670:     if (!(ide = get_ide_controller(i->rdev))) {
671:         return -EINVAL;
672:     }
673:
674:     if (!get_device(BLK_DEV, i->rdev)) {
675:         return -ENXIO;
676:     }
677:
678:     drive = get_ide_drive(i->rdev);
679:     if (ide->drive[drive].fsop && ide->drive[drive].fsop->close) {
680:         return ide->drive[drive].fsop->close(i, fd_table);
681:     }
682:     return -EINVAL;
683: }
684:
685: int ide_read(__dev_t dev, __blk_t block, char *buffer, int blksize)
686: {
687:     int drive;
688:     struct ide *ide;
689:
690:     if (!(ide = get_ide_controller(dev))) {
691:         printk("%s(): no ide controller!\n", __FUNCTION__);
692:         return -EINVAL;
693:     }
694:
695:     if (!get_device(BLK_DEV, dev)) {
696:         return -ENXIO;
697:     }
698:
699:     drive = get_ide_drive(dev);
700:     if (ide->drive[drive].fsop && ide->drive[drive].fsop->read_block) {
701:         return ide->drive[drive].fsop->read_block(dev, block, buffer, bl
ksize);
702:     }
703:     printk("WARNING: %s(): device %d,%d does not have the read_block() metho
d!\n", __FUNCTION__, MAJOR(dev), MINOR(dev));
704:     return -EINVAL;
705: }
706:
707: int ide_write(__dev_t dev, __blk_t block, char *buffer, int blksize)
708: {

```

drivers/block/ide.c

Page 13/15

```

709:         int drive;
710:         struct ide *ide;
711:
712:         if(!(ide = get_ide_controller(dev))) {
713:             printk("%s(): no ide controller!\n", __FUNCTION__);
714:             return -EINVAL;
715:         }
716:
717:         if(!get_device(BLK_DEV, dev)) {
718:             return -ENXIO;
719:         }
720:
721:         drive = get_ide_drive(dev);
722:         if(ide->drive[drive].fsop && ide->drive[drive].fsop->write_block) {
723:             return ide->drive[drive].fsop->write_block(dev, block, buffer, b
lksize);
724:         }
725:         printk("WARNING: %s(): device %d,%d does not have the write_block() meth
od!\n", __FUNCTION__, MAJOR(dev), MINOR(dev));
726:         return -EINVAL;
727:     }
728:
729: int ide_ioctl(struct inode *i, int cmd, unsigned long int arg)
730: {
731:     int drive;
732:     struct ide *ide;
733:
734:     if(!(ide = get_ide_controller(i->rdev))) {
735:         return -EINVAL;
736:     }
737:
738:     if(!get_device(BLK_DEV, i->rdev)) {
739:         return -ENXIO;
740:     }
741:
742:     drive = get_ide_drive(i->rdev);
743:     if(ide->drive[drive].fsop && ide->drive[drive].fsop->ioctl) {
744:         return ide->drive[drive].fsop->ioctl(i, cmd, arg);
745:     }
746:     return -EINVAL;
747: }
748:
749: void ide_init(void)
750: {
751:     int devices, errno;
752:     struct ide *ide;
753:
754:     if(!register_irq(IDE0_IRQ, &irq_config_ide0)) {
755:         enable_irq(IDE0_IRQ);
756:     }
757:     devices = 0;
758:
759:     ide = &ide_table[IDE_PRIMARY];
760:     errno = ide_softreset(ide);
761:     if(!(errno & 1)) {
762:         if(!(ide_identify(ide, IDE_MASTER))) {
763:             get_device_size(&ide->drive[IDE_MASTER]);
764:             ide_results(ide, IDE_MASTER);
765:             SET_MINOR(ide0_device.minors, 0);
766:             register_device(BLK_DEV, &ide0_device);
767:             if(ide->drive[IDE_MASTER].flags & DEVICE_IS_DISK) {
768:                 if(!ide_hd_init(ide, IDE_MASTER)) {
769:                     devices++;
770:                 }
771:             }
772:             if(ide->drive[IDE_MASTER].flags & DEVICE_IS_CDROM) {
773:                 if(!ide_cd_init(ide, IDE_MASTER)) {

```

drivers/block/ide.c

Page 14/15

```

774:                                     devices++;
775:                                     }
776:                                 }
777:                            }
778:    }
779:    if(!(errno & 0x10)) {
780:        if(!(ide_identify(ide, IDE_SLAVE))) {
781:            get_device_size(&ide->drive[IDE_SLAVE]);
782:            ide_results(ide, IDE_SLAVE);
783:            SET_MINOR(ide0_device.minors, 1 << IDE_SLAVE_MSF);
784:            if(!devices) {
785:                register_device(BLK_DEV, &ide0_device);
786:            }
787:            if(ide->drive[IDE_SLAVE].flags & DEVICE_IS_DISK) {
788:                if(!ide_hd_init(ide, IDE_SLAVE)) {
789:                    devices++;
790:                }
791:            }
792:            if(ide->drive[IDE_SLAVE].flags & DEVICE_IS_CDROM) {
793:                if(!ide_cd_init(ide, IDE_SLAVE)) {
794:                    devices++;
795:                }
796:            }
797:        }
798:    }
799:    if(!devices) {
800:        disable_irq(IDE0_IRQ);
801:        unregister_irq(IDE0_IRQ, &irq_config_ide0);
802:    }
803:
804:    if(!register_irq(IDE1_IRQ, &irq_config_idel)) {
805:        enable_irq(IDE1_IRQ);
806:    }
807:    devices = 0;
808:    ide = &ide_table[IDE_SECONDARY];
809:    errno = ide_softreset(ide);
810:    if(!(errno & 1)) {
811:        if(!(ide_identify(ide, IDE_MASTER))) {
812:            get_device_size(&ide->drive[IDE_MASTER]);
813:            ide_results(ide, IDE_MASTER);
814:            SET_MINOR(idel_device.minors, 0);
815:            register_device(BLK_DEV, &idel_device);
816:            if(ide->drive[IDE_MASTER].flags & DEVICE_IS_DISK) {
817:                if(!ide_hd_init(ide, IDE_MASTER)) {
818:                    devices++;
819:                }
820:            }
821:            if(ide->drive[IDE_MASTER].flags & DEVICE_IS_CDROM) {
822:                if(!ide_cd_init(ide, IDE_MASTER)) {
823:                    devices++;
824:                }
825:            }
826:        }
827:    }
828:    if(!(errno & 0x10)) {
829:        if(!(ide_identify(ide, IDE_SLAVE))) {
830:            get_device_size(&ide->drive[IDE_SLAVE]);
831:            ide_results(ide, IDE_SLAVE);
832:            SET_MINOR(idel_device.minors, 1 << IDE_SLAVE_MSF);
833:            if(!devices) {
834:                register_device(BLK_DEV, &idel_device);
835:            }
836:            if(ide->drive[IDE_SLAVE].flags & DEVICE_IS_DISK) {
837:                if(!ide_hd_init(ide, IDE_SLAVE)) {
838:                    devices++;
839:                }
840:            }

```

drivers/block/ide.c

Page 15/15

```
841:                 if(ide->drive[IDE_SLAVE].flags & DEVICE_IS_CDROM) {
842:                     if(!ide_cd_init(ide, IDE_SLAVE)) {
843:                         devices++;
844:                     }
845:                 }
846:             }
847:         }
848:         if(!devices) {
849:             disable_irq(IDE1_IRQ);
850:             unregister_irq(IDE1_IRQ, &irq_config_ide1);
851:         }
852: }
```

drivers/block/ide_cd.c

Page 1/9

```

1: /*
2:  * fiwix/drivers/block/ide_cd.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/buffer.h>
10: #include <fiwix/ide.h>
11: #include <fiwix/ide_cd.h>
12: #include <fiwix/ioctl.h>
13: #include <fiwix/devices.h>
14: #include <fiwix/sleep.h>
15: #include <fiwix/timer.h>
16: #include <fiwix/sched.h>
17: #include <fiwix/cpu.h>
18: #include <fiwix/fs.h>
19: #include <fiwix/part.h>
20: #include <fiwix/process.h>
21: #include <fiwix/mm.h>
22: #include <fiwix/errno.h>
23: #include <fiwix/stdio.h>
24: #include <fiwix/string.h>
25:
26: /* default size of 1GB is enough to read a whole CDROM */
27: #define CDROM_DEFAULT_SIZE      (1024 * 1024)    /* in KBs */
28:
29: static struct fs_operations ide_cd_driver_fsop = {
30:     0,
31:     0,
32:
33:     ide_cd_open,
34:     ide_cd_close,
35:     NULL,                /* read */
36:     NULL,                /* write */
37:     ide_cd_ioctl,
38:     NULL,                /* lseek */
39:     NULL,                /* readdir */
40:     NULL,                /* mmap */
41:     NULL,                /* select */
42:
43:     NULL,                /* readlink */
44:     NULL,                /* followlink */
45:     NULL,                /* bmap */
46:     NULL,                /* lockup */
47:     NULL,                /* rmdir */
48:     NULL,                /* link */
49:     NULL,                /* unlink */
50:     NULL,                /* symlink */
51:     NULL,                /* mkdir */
52:     NULL,                /* mknod */
53:     NULL,                /* truncate */
54:     NULL,                /* create */
55:     NULL,                /* rename */
56:
57:     ide_cd_read,
58:     NULL,                /* write_block */
59:
60:     NULL,                /* read_inode */
61:     NULL,                /* write_inode */
62:     NULL,                /* ialloc */
63:     NULL,                /* ifree */
64:     NULL,                /* statfs */
65:     NULL,                /* read_superblock */
66:     NULL,                /* remount_fs */
67:     NULL,                /* write_superblock */

```

drivers/block/ide_cd.c

Page 2/9

```

68:         NULL                                     /* release_superblock */
69: };
70:
71: static char *sense_key_err[] = {
72:     "NO SENSE",
73:     "RECOVERED ERROR",
74:     "NOT READY",
75:     "MEDIUM ERROR",
76:     "HARDWARE ERROR",
77:     "ILLEGAL REQUEST",
78:     "UNIT ATTENTION",
79:     "DATA PROTECT",
80:     "RESERVED",
81:     "RESERVED",
82:     "RESERVED",
83:     "ABORTED COMMAND",
84:     "MISCOMPARE",
85:     "RESERVED"
86: };
87:
88: enum {
89:     RS_NO_SENSE,
90:     RS_RECOVERED_ERROR,
91:     RS_NOT_READY,
92:     RS_MEDIUM_ERROR,
93:     RS_HARDWARE_ERROR,
94:     RS_ILLEGAL_REQUEST,
95:     RS_UNIT_ATTENTION,
96:     RS_DATA_PROTECT,
97:     RS_RESERVED1,
98:     RS_RESERVED2,
99:     RS_RESERVED3,
100:    RS_ABORTED_COMMAND,
101:    RS_MISCOMPARE,
102:    RS_RESERVED4
103: };
104:
105: static int send_packet_command(unsigned char *pkt, struct ide *ide, int drive, int
nt blksize)
106: {
107:     int n, retries, status;
108:
109:     outport_b(ide->ctrl + IDE_DEV_CTRL, 0);
110:     ide_delay();
111:     outport_b(ide->base + IDE_DRVHD, IDE_CHS_MODE);
112:     ide_delay();
113:     if(ide_drvsel(ide, drive, IDE_CHS_MODE, 0)) {
114:         printk("WARNING: %s(): %s: drive not ready to receive PACKET com
mand.\n", __FUNCTION__, ide->drive[drive].dev_name);
115:         return 1;
116:     }
117:
118:     CLI();
119:     outport_b(ide->base + IDE_FEATURES, 0);
120:     outport_b(ide->base + IDE_SECCNT, 0);
121:     outport_b(ide->base + IDE_SECNUM, 0);
122:     outport_b(ide->base + IDE_LCYL, blksize & 0xFF);
123:     outport_b(ide->base + IDE_HCYL, blksize >> 8);
124:     outport_b(ide->base + IDE_DRVHD, drive << 4);
125:     outport_b(ide->base + IDE_COMMAND, ATA_PACKET);
126:     ide_wait400ns(ide);
127:
128: /*
129:  * NOTE: Some devices prior to ATA/ATAPI-4 assert INTRQ if enabled at this
130:  * point. See IDENTIFY PACKET DEVICE, word 0, bits 5-6 to determine if an
131:  * interrupt will occur.
132:  */

```

drivers/block/ide_cd.c

Page 3/9

```

133:         SET_IDE_RDY_RETR(retries);
134:
135:         for(n = 0; n < retries; n++) {
136:             status = inport_b(ide->base + IDE_STATUS);
137:             if((status & (IDE_STAT_DRQ | IDE_STAT_BSY)) == IDE_STAT_DRQ) {
138:                 break;
139:             }
140:             ide_delay();
141:         }
142:         if(n >= retries) {
143:             printk("WARNING: %s(): %s: drive not ready to receive command pa
cket (retries = %d).\n", __FUNCTION__, ide->drive[drive].dev_name, n);
144:             return 1;
145:         }
146:
147:         outport_sw(ide->base + IDE_DATA, pkt, 12 / sizeof(short int));
148:         return 0;
149:     }
150:
151: static int atapi_read_data(__dev_t dev, char *data, struct ide *ide, int blksize
, int offset)
152: {
153:     int errno, status;
154:     char *buffer;
155:     int retries, bytes;
156:     struct callout_req creq;
157:
158:     status = 0;
159:
160:     for(retries = 0; retries < MAX_IDE_ERR; retries++) {
161:         if(ide->channel == IDE_PRIMARY) {
162:             ide0_wait_interrupt = ide->base;
163:             creq.fn = ide0_timer;
164:             creq.arg = 0;
165:             add_callout(&creq, WAIT_FOR_IDE);
166:             sleep(&irq_ide0, PROC_UNINTERRUPTIBLE);
167:             if(ide0_timeout) {
168:                 status = inport_b(ide->base + IDE_STATUS);
169:                 if((status & (IDE_STAT_RDY | IDE_STAT_DRQ)) != (
IDE_STAT_RDY | IDE_STAT_DRQ)) {
170:                     continue;
171:                 }
172:             }
173:             del_callout(&creq);
174:         }
175:         if(ide->channel == IDE_SECONDARY) {
176:             idel_wait_interrupt = ide->base;
177:             creq.fn = idel_timer;
178:             creq.arg = 0;
179:             add_callout(&creq, WAIT_FOR_IDE);
180:             sleep(&irq_idel, PROC_UNINTERRUPTIBLE);
181:             if(idel_timeout) {
182:                 status = inport_b(ide->base + IDE_STATUS);
183:                 if((status & (IDE_STAT_RDY | IDE_STAT_DRQ)) != (
IDE_STAT_RDY | IDE_STAT_DRQ)) {
184:                     continue;
185:                 }
186:             }
187:             del_callout(&creq);
188:         }
189:         status = inport_b(ide->base + IDE_STATUS);
190:         if(status & IDE_STAT_ERR) {
191:             continue;
192:         }
193:
194:         if((status & (IDE_STAT_DRQ | IDE_STAT_BSY)) == 0) {
195:             break;

```

drivers/block/ide_cd.c

Page 4/9

```

196:         }
197:
198:         bytes = (inport_b(ide->base + IDE_HCYL) << 8) + inport_b(ide->ba
se + IDE_LCYL);
199:         if(!bytes || bytes > blksize) {
200:             break;
201:         }
202:
203:         bytes = MAX(bytes, IDE_CD_SECTSIZE);    /* read more than 2048 b
ytes is not supported */
204:         buffer = data + offset;
205:         inport_sw(ide->base + IDE_DATA, (void *)buffer, bytes / sizeof(s
hort int));
206:     }
207:
208:     if(status & IDE_STAT_ERR) {
209:         errno = inport_b(ide->base + IDE_ERROR);
210:         printk("WARNING: %s(): error on cdrom device %d,%d, status=0x%x
error=0x%x,\n", __FUNCTION__, MAJOR(dev), MINOR(dev), status, errno);
211:         return 1;
212:     }
213:
214:     if(retries >= MAX_IDE_ERR) {
215:         printk("WARNING: %s(): timeout on cdrom device %d,%d, status=0x%
x.\n", __FUNCTION__, MAJOR(dev), MINOR(dev), status);
216:         /* a reset may be required at this moment */
217:         return 1;
218:     }
219:     return 0;
220: }
221:
222: static int atapi_cmd_testunit(struct ide *ide, int drive)
223: {
224:     unsigned char pkt[12];
225:
226:     pkt[0] = ATAPI_TEST_UNIT;
227:     pkt[1] = NULL;
228:     pkt[2] = NULL;
229:     pkt[3] = NULL;
230:     pkt[4] = NULL;
231:     pkt[5] = NULL;
232:     pkt[6] = NULL;
233:     pkt[7] = NULL;
234:     pkt[8] = NULL;
235:     pkt[9] = NULL;
236:     pkt[10] = NULL;
237:     pkt[11] = NULL;
238:     return send_packet_command(pkt, ide, drive, 0);
239: }
240:
241: static int atapi_cmd_reqsense(struct ide *ide, int drive)
242: {
243:     unsigned char pkt[12];
244:
245:     pkt[0] = ATAPI_REQUEST_SENSE;
246:     pkt[1] = NULL;
247:     pkt[2] = NULL;
248:     pkt[3] = NULL;
249:     pkt[4] = 252;    /* this command can send up to 252 bytes */
250:     pkt[5] = NULL;
251:     pkt[6] = NULL;
252:     pkt[7] = NULL;
253:     pkt[8] = NULL;
254:     pkt[9] = NULL;
255:     pkt[10] = NULL;
256:     pkt[11] = NULL;
257:     return send_packet_command(pkt, ide, drive, 0);

```


drivers/block/ide_cd.c

Page 5/9

```

258: }
259:
260: static int atapi_cmd_startstop(int action, struct ide *ide, int drive)
261: {
262:     unsigned char pkt[12];
263:
264:     pkt[0] = ATAPI_START_STOP;
265:     pkt[1] = NULL;
266:     pkt[2] = NULL;
267:     pkt[3] = NULL;
268:     pkt[4] = action;
269:     pkt[5] = NULL;
270:     pkt[6] = NULL;
271:     pkt[7] = NULL;
272:     pkt[8] = NULL;
273:     pkt[9] = NULL;
274:     pkt[10] = NULL;
275:     pkt[11] = NULL;
276:     return send_packet_command(pkt, ide, drive, 0);
277: }
278:
279: static int atapi_cmd_mediumrm(int action, struct ide *ide, int drive)
280: {
281:     unsigned char pkt[12];
282:
283:     pkt[0] = ATAPI_MEDIUM_REMOVAL;
284:     pkt[1] = NULL;
285:     pkt[2] = NULL;
286:     pkt[3] = NULL;
287:     pkt[4] = action;
288:     pkt[5] = NULL;
289:     pkt[6] = NULL;
290:     pkt[7] = NULL;
291:     pkt[8] = NULL;
292:     pkt[9] = NULL;
293:     pkt[10] = NULL;
294:     pkt[11] = NULL;
295:     return send_packet_command(pkt, ide, drive, 0);
296: }
297:
298: static int request_sense(char *buffer, __dev_t dev, struct ide *ide, int drive)
299: {
300:     int errcode;
301:     int sense_key, sense_asc;
302:
303:     errcode = inport_b(ide->base + IDE_ERROR);
304:     sense_key = (errcode & 0xF0) >> 4;
305:     printk("\tSense Key code indicates a '%s' condition.\n", sense_key_err[sense_key & 0xF]);
306:     errcode = atapi_cmd_reqsense(ide, drive);
307:     printk("reqsense() returned %d\n", errcode);
308:     errcode = atapi_read_data(dev, buffer, ide, BLKSIZE_2K, 0);
309:     printk("atapi_read_data() returned %d\n", errcode);
310:     errcode = (int)(buffer[0] & 0x7F);
311:     sense_key = (int)(buffer[2] & 0xF);
312:     sense_asc = (int)(buffer[12] & 0xFF);
313:     printk("errcode = %x\n", errcode);
314:     printk("sense_key = %x\n", sense_key);
315:     printk("sense_asc = %x\n", sense_asc);
316:     return errcode;
317: }
318:
319: void ide_cd_timer(unsigned int arg)
320: {
321:     wakeup(&ide_cd_open);
322: }
323:

```

drivers/block/ide_cd.c

Page 6/9

```

324: int ide_cd_open(struct inode *i, struct fd *fd_table)
325: {
326:     int drive;
327:     char *buffer;
328:     int errcode;
329:     int sense_key, sense_asc;
330:     int retries;
331:     struct ide *ide;
332:
333:     if(!(ide = get_ide_controller(i->rdev))) {
334:         return -EINVAL;
335:     }
336:
337:     drive = get_ide_drive(i->rdev);
338:
339:     lock_resource(&ide->resource);
340:
341:     if(!(buffer = (void *)kmalloc())) {
342:         unlock_resource(&ide->resource);
343:         return -ENOMEM;
344:     }
345:
346:     if((errcode = atapi_cmd_testunit(ide, drive)) {
347:         printk("WARNING: %s(): cdrom device %d,%d is not ready for TEST_
UNIT, error %d.\n", __FUNCTION__, MAJOR(i->rdev), MINOR(i->rdev), errcode);
348:         request_sense(buffer, i->rdev, ide, drive);
349:     }
350:
351:     for(retries = 0; retries < MAX_CD_ERR; retries++) {
352:         if(!(errcode = atapi_cmd_startstop(CD_LOAD, ide, drive))) {
353:             break;
354:         }
355:         printk("WARNING: %s(): cdrom device %d,%d is not ready for CD_IO
AD, error %d.\n", __FUNCTION__, MAJOR(i->rdev), MINOR(i->rdev), errcode);
356:         atapi_read_data(i->rdev, buffer, ide, BLKSIZE_2K, 0);
357:         errcode = request_sense(buffer, i->rdev, ide, drive);
358:         sense_key = (errcode & 0xF0) >> 4;
359:         /* trying to eject on slim drives may lead to an illegal request
*/
360:         if(!sense_key || sense_key == RS_ILLEGAL_REQUEST) {
361:             break;
362:         }
363:         if(errcode == 0x70 || errcode == 0x71) {
364:             sense_key = (int)(buffer[2] & 0xF);
365:             sense_asc = (int)(buffer[12] & 0xFF);
366:             if(sense_key == RS_NOT_READY && sense_asc == ASC_NO_MEDI
UM) {
367:                 kfree((unsigned int)buffer);
368:                 unlock_resource(&ide->resource);
369:                 return -ENOMEDIUM;
370:             }
371:         }
372:     }
373:
374:     if(retries == MAX_CD_ERR) {
375:         if(sense_key == RS_NOT_READY) {
376:             kfree((unsigned int)buffer);
377:             unlock_resource(&ide->resource);
378:             return -ENOMEDIUM;
379:         }
380:     }
381:
382:     if(atapi_cmd_mediumrm(CD_LOCK_MEDIUM, ide, drive)) {
383:         printk("WARNING: %s(): error on cdrom device %d,%d while trying
to lock medium.\n", __FUNCTION__, MAJOR(i->rdev), MINOR(i->rdev), ATAPI_MEDIUM_REMOVAL)
;
384:         request_sense(buffer, i->rdev, ide, drive);

```

drivers/block/ide_cd.c

Page 7/9

```

385:         }
386:
387:         /* this line just to catch interrupt */
388:         atapi_read_data(i->rdev, buffer, ide, BLKSIZE_2K, 0);
389:         kfree((unsigned int)buffer);
390:
391:         unlock_resource(&ide->resource);
392:         return 0;
393:     }
394:
395: int ide_cd_close(struct inode *i, struct fd *fd_table)
396: {
397:     int drive;
398:     char *buffer;
399:     struct ide *ide;
400:
401:     if (!(ide = get_ide_controller(i->rdev))) {
402:         return -EINVAL;
403:     }
404:
405:     if (!(buffer = (void *)kmalloc())) {
406:         return -ENOMEM;
407:     }
408:
409:     drive = get_ide_drive(i->rdev);
410:
411:     /* FIXME: only if device usage == 0 */
412:     invalidate_buffers(i->rdev);
413:
414:     if (atapi_cmd_mediumrm(CD_UNLOCK_MEDIUM, ide, drive)) {
415:         printk("WARNING: %s(): error on cdrom device %d,%d during 0x%x c
ommand.\n", __FUNCTION__, MAJOR(i->rdev), MINOR(i->rdev), ATAPI_MEDIUM_REMOVAL);
416:     }
417:
418:     /* this line just to catch interrupt */
419:     atapi_read_data(i->rdev, buffer, ide, BLKSIZE_2K, 0);
420:     kfree((unsigned int)buffer);
421:
422:     return 0;
423: }
424:
425: int ide_cd_read(__dev_t dev, __blk_t block, char *buffer, int blksize)
426: {
427:     int drive;
428:     int sectors_to_read;
429:     int n, retries;
430:     unsigned char pkt[12];
431:     struct ide *ide;
432:
433:     if (!(ide = get_ide_controller(dev))) {
434:         return -EINVAL;
435:     }
436:
437:     drive = get_ide_drive(dev);
438:     blksize = BLKSIZE_2K;
439:     sectors_to_read = blksize / IDE_CD_SECTSIZE;
440:
441:     pkt[0] = ATAPI_READ10;
442:     pkt[1] = NULL;
443:     pkt[2] = (block >> 24) & 0xFF;
444:     pkt[3] = (block >> 16) & 0xFF;
445:     pkt[4] = (block >> 8) & 0xFF;
446:     pkt[5] = block & 0xFF;
447:     pkt[6] = NULL;
448:     pkt[7] = (sectors_to_read >> 8) & 0xFF;
449:     pkt[8] = sectors_to_read & 0xFF;
450:     pkt[9] = NULL;

```

drivers/block/ide_cd.c

Page 8/9

```

451:         pkt[10] = NULL;
452:         pkt[11] = NULL;
453:
454:         lock_resource(&ide->resource);
455:         for(n = 0; n < sectors_to_read; n++, block++) {
456:             for(retries = 0; retries < MAX_CD_ERR; retries++) {
457:                 if(send_packet_command(pkt, ide, drive, blksize)) {
458:                     printk("\tblock=%d, offset=%d\n", block, block *
blksize);
459:
460:                     unlock_resource(&ide->resource);
461:                     return -EIO;
462:                 }
463:                 if(atapi_read_data(dev, buffer, ide, blksize, n * IDE_CD
_SECTSIZE)) {
464:                     int errcode;
465:                     int sense_key;
466:                     errcode = inport_b(ide->base + IDE_ERROR);
467:                     sense_key = (errcode & 0xF0) >> 4;
468:                     printk("\tSense Key code indicates a '%s' condit
ion.\n", sense_key_err[sense_key & 0xF]);
469:                     if(sense_key) {
470:                         continue;
471:                     }
472:                     break;
473:                 }
474:                 if(retries == MAX_CD_ERR) {
475:                     printk("\tblock=%d, offset=%d\n", block, block * blksize
);
476:                     unlock_resource(&ide->resource);
477:                     return -EIO;
478:                 }
479:             }
480:         }
481:         unlock_resource(&ide->resource);
482:         return sectors_to_read * IDE_CD_SECTSIZE;
483:     }
484:
485: int ide_cd_ioctl(struct inode *i, int cmd, unsigned long int arg)
486: {
487:     struct ide *ide;
488:
489:     if(!(ide = get_ide_controller(i->rdev))) {
490:         return -EINVAL;
491:     }
492:
493:     switch(cmd) {
494:         default:
495:             return -EINVAL;
496:             break;
497:     }
498:
499:     return 0;
500: }
501:
502: int ide_cd_init(struct ide *ide, int drive)
503: {
504:     struct device *d;
505:     unsigned char minor;
506:
507:     ide->drive[drive].fsop = &ide_cd_driver_fsop;
508:
509:     minor = !ide->drive[drive].minor_shift ? 0 : 1 << ide->drive[drive].mino
r_shift;
510:
511:     if(!(d = get_device(BLK_DEV, MKDEV(ide->drive[drive].major, minor)))) {
512:         return -EINVAL;

```

drivers/block/ide_cd.c

Page 9/9

```
513:         }
514:         SET_MINOR(d->minors, minor);
515:         ((unsigned int *)d->device_data)[minor] = CDROM_DEFAULT_SIZE;
516:
517:         return 0;
518: }
```

drivers/block/ide_hd.c

Page 1/9

```

1:  /*
2:  *  fiwix/drivers/block/ide_hd.c
3:  *
4:  *  Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  *  Distributed under the terms of the Fiwix License.
6:  */
7:
8:  #include <fiwix/asm.h>
9:  #include <fiwix/buffer.h>
10: #include <fiwix/ide.h>
11: #include <fiwix/ide_hd.h>
12: #include <fiwix/ioctl.h>
13: #include <fiwix/devices.h>
14: #include <fiwix/sleep.h>
15: #include <fiwix/timer.h>
16: #include <fiwix/sched.h>
17: #include <fiwix/cpu.h>
18: #include <fiwix/fs.h>
19: #include <fiwix/part.h>
20: #include <fiwix/process.h>
21: #include <fiwix/mm.h>
22: #include <fiwix/errno.h>
23: #include <fiwix/stdio.h>
24: #include <fiwix/string.h>
25:
26: static struct fs_operations ide_hd_driver_fsop = {
27:     0,
28:     0,
29:
30:     ide_hd_open,
31:     ide_hd_close,
32:     NULL,                /* read */
33:     NULL,                /* write */
34:     ide_hd_ioctl,
35:     NULL,                /* lseek */
36:     NULL,                /* readdir */
37:     NULL,                /* mmap */
38:     NULL,                /* select */
39:
40:     NULL,                /* readlink */
41:     NULL,                /* followlink */
42:     NULL,                /* bmap */
43:     NULL,                /* lockup */
44:     NULL,                /* rmdir */
45:     NULL,                /* link */
46:     NULL,                /* unlink */
47:     NULL,                /* symlink */
48:     NULL,                /* mkdir */
49:     NULL,                /* mknod */
50:     NULL,                /* truncate */
51:     NULL,                /* create */
52:     NULL,                /* rename */
53:
54:     ide_hd_read,
55:     ide_hd_write,
56:
57:     NULL,                /* read_inode */
58:     NULL,                /* write_inode */
59:     NULL,                /* ialloc */
60:     NULL,                /* ifree */
61:     NULL,                /* statfs */
62:     NULL,                /* read_superblock */
63:     NULL,                /* remount_fs */
64:     NULL,                /* write_superblock */
65:     NULL,                /* release_superblock */
66: };
67:

```

drivers/block/ide_hd.c

Page 2/9

```

68: static void assign_minors(__dev_t rdev, struct ide *ide, struct partition *part)
69: {
70:     int n;
71:     int drive, minor;
72:     struct device *d;
73:
74:     minor = 0;
75:     drive = get_ide_drive(rdev);
76:
77:     if(!(d = get_device(BLK_DEV, rdev))) {
78:         printk("WARNING: %s(): unable to assign minors to device %d,%d.\n
n", __FUNCTION__, MAJOR(rdev), MINOR(rdev));
79:         return;
80:     }
81:
82:     for(n = 0; n < NR_PARTITIONS; n++) {
83:         if(drive == IDE_MASTER) {
84:             minor = (1 << ide->drive[drive].minor_shift) + n;
85:         }
86:         if(drive == IDE_SLAVE) {
87:             minor = (1 << ide->drive[drive].minor_shift) + n + 1;
88:         }
89:         CLEAR_MINOR(d->minors, minor);
90:         if(part[n].type) {
91:             SET_MINOR(d->minors, minor);
92:             ((unsigned int *)d->device_data)[minor] = part[n].nr_sec
ts / 2;
93:         }
94:     }
95: }
96:
97: static __off_t block2sector(__blk_t block, int blksize, struct partition *part,
int minor)
98: {
99:     __off_t sector;
100:
101:     sector = block * (blksize / IDE_HD_SECTSIZE);
102:     if(minor) {
103:         sector += part[minor - 1].startsect;
104:     }
105:     return sector;
106: }
107:
108: static void sector2chs(__off_t offset, int *cyl, int *head, int *sector, struct
ide_drv_ident *ident)
109: {
110:     *cyl = offset / (ident->logic_spt * ident->logic_heads);
111:     *head = (offset / ident->logic_spt) % ident->logic_heads;
112:     *sector = (offset % ident->logic_spt) + 1;
113: }
114:
115: int ide_hd_open(struct inode *i, struct fd *fd_table)
116: {
117:     return 0;
118: }
119:
120: int ide_hd_close(struct inode *i, struct fd *fd_table)
121: {
122:     sync_buffers(i->rdev);
123:     return 0;
124: }
125:
126: int ide_hd_read(__dev_t dev, __blk_t block, char *buffer, int blksize)
127: {
128:     int minor;
129:     int drive;
130:     int sectors_to_read, cmd;

```

drivers/block/ide_hd.c

Page 3/9

```

131:         int n, status, r, retries;
132:         int cyl, head, sector;
133:         __off_t offset;
134:         struct ide *ide;
135:         struct ide_drv_ident *ident;
136:         struct partition *part;
137:         struct callout_req creq;
138:
139:         if(!(ide = get_ide_controller(dev))) {
140:             return -EINVAL;
141:         }
142:
143:         minor = MINOR(dev);
144:         if((drive = get_ide_drive(dev))) {
145:             minor &= ~(1 << IDE_SLAVE_MSF);
146:         }
147:
148:         SET_IDE_RDY_RETR(retries);
149:
150:         blksize = blksize ? blksize : BLKSIZE_1K;
151:         sectors_to_read = MIN(blksize, PAGE_SIZE) / IDE_HD_SECTSIZE;
152:
153:         ident = &ide->drive[drive].ident;
154:         part = ide->drive[drive].part_table;
155:         offset = block2sector(block, blksize, part, minor);
156:
157:         CLI();
158:         lock_resource(&ide->resource);
159:
160:         n = 0;
161:
162:         while(n < sectors_to_read) {
163:             if(ide->drive[drive].flags & DEVICE_HAS_RW_MULTIPLE) {
164:                 outport_b(ide->base + IDE_SECCNT, sectors_to_read);
165:                 cmd = ATA_READ_MULTIPLE_PIO;
166:             } else {
167:                 outport_b(ide->base + IDE_SECCNT, 1);
168:                 cmd = ATA_READ_PIO;
169:             }
170:
171:             if(ide->drive[drive].flags & DEVICE_REQUIRES_LBA) {
172:                 outport_b(ide->base + IDE_SECNUM, offset & 0xFF);
173:                 outport_b(ide->base + IDE_LCYL, (offset >> 8) & 0xFF);
174:                 outport_b(ide->base + IDE_HCYL, (offset >> 16) & 0xFF);
175:                 if(ide_drvsel(ide, drive, IDE_LBA_MODE, (offset >> 24) &
0x0F)) {
176:                     printk("WARNING: %s(): %s: drive not ready.\n",
__FUNCTION__, ide->drive[drive].dev_name);
177:                     unlock_resource(&ide->resource);
178:                     return -EIO;
179:                 }
180:             } else {
181:                 sector2chs(offset, &cyl, &head, &sector, ident);
182:                 outport_b(ide->base + IDE_SECNUM, sector);
183:                 outport_b(ide->base + IDE_LCYL, cyl);
184:                 outport_b(ide->base + IDE_HCYL, (cyl >> 8));
185:                 if(ide_drvsel(ide, drive, IDE_CHS_MODE, head)) {
186:                     printk("WARNING: %s(): %s: drive not ready.\n",
__FUNCTION__, ide->drive[drive].dev_name);
187:                     unlock_resource(&ide->resource);
188:                     return -EIO;
189:                 }
190:             }
191:             outport_b(ide->base + IDE_COMMAND, cmd);
192:             if(ide->channel == IDE_PRIMARY) {
193:                 ide0_wait_interrupt = ide->base;
194:                 creq.fn = ide0_timer;

```


drivers/block/ide_hd.c

Page 4/9

```

195:         creq.arg = 0;
196:         add_callout(&creq, WAIT_FOR_IDE);
197:         if(ide0_wait_interrupt) {
198:             sleep(&irq_ide0, PROC_UNINTERRUPTIBLE);
199:         }
200:         if(ide0_timeout) {
201:             printk("WARNING: %s(): %s: timeout on hard disk
dev %d,%d during read.\n", __FUNCTION__, ide->drive[drive].dev_name, MAJOR(dev), MINOR(
dev));
202:         } else {
203:             del_callout(&creq);
204:         }
205:     }
206:     if(ide->channel == IDE_SECONDARY) {
207:         idel_wait_interrupt = ide->base;
208:         creq.fn = idel_timer;
209:         creq.arg = 0;
210:         add_callout(&creq, WAIT_FOR_IDE);
211:         if(idel_wait_interrupt) {
212:             sleep(&irq_idel, PROC_UNINTERRUPTIBLE);
213:         }
214:         if(idel_timeout) {
215:             printk("WARNING: %s(): %s: timeout on hard disk
dev %d,%d during read.\n", __FUNCTION__, ide->drive[drive].dev_name, MAJOR(dev), MINOR(
dev));
216:         } else {
217:             del_callout(&creq);
218:         }
219:     }
220:     for(r = 0; r < retries; r++) {
221:         status = inport_b(ide->base + IDE_STATUS);
222:         if(!(status & IDE_STAT_BSY) && (status & IDE_STAT_DRQ))
{
223:             break;
224:         }
225:         ide_delay();
226:     }
227:     if(status & IDE_STAT_ERR) {
228:         printk("WARNING: %s(): %s: error on hard disk dev %d,%d
during read.\n", __FUNCTION__, ide->drive[drive].dev_name, MAJOR(dev), MINOR(dev));
229:         printk("\tstatus=0x%x ", status);
230:         ide_error(ide, status);
231:         printk("\tblock %d, sector %d.\n", block, offset);
232:         inport_b(ide->base + IDE_STATUS);          /* clear any pen
ding interrupt */
233:         unlock_resource(&ide->resource);
234:         return -EIO;
235:     }
236:
237:     if(cmd == ATA_READ_MULTIPLE_PIO) {
238:         inport_sw(ide->base + IDE_DATA, (void *)buffer, (IDE_HD_
SECTSIZE * sectors_to_read) / sizeof(short int));
239:         break;
240:     }
241:     inport_sw(ide->base + IDE_DATA, (void *)buffer, IDE_HD_SECTSIZE
/ sizeof(short int));
242:     inport_b(ide->ctrl + IDE_ALT_STATUS);          /* ignore results */
243:     inport_b(ide->base + IDE_STATUS);             /* clear any pending int
errupt */
244:     n++;
245:     offset++;
246:     buffer += IDE_HD_SECTSIZE;
247: }
248: inport_b(ide->ctrl + IDE_ALT_STATUS);          /* ignore results */
249: inport_b(ide->base + IDE_STATUS);             /* clear any pending interrupt */
/
250:     unlock_resource(&ide->resource);

```

drivers/block/ide_hd.c

Page 5/9

```

251:         return sectors_to_read * IDE_HD_SECTSIZE;
252: }
253:
254: int ide_hd_write(__dev_t dev, __blk_t block, char *buffer, int blksize)
255: {
256:     int minor;
257:     int drive;
258:     int sectors_to_write, cmd;
259:     int n, status, r, retries;
260:     int cyl, head, sector;
261:     __off_t offset;
262:     struct ide *ide;
263:     struct ide_drv_ident *ident;
264:     struct partition *part;
265:     struct callout_req creq;
266:
267:     if(!(ide = get_ide_controller(dev))) {
268:         return -EINVAL;
269:     }
270:
271:     minor = MINOR(dev);
272:     if((drive = get_ide_drive(dev)) {
273:         minor &= ~(1 << IDE_SLAVE_MSF);
274:     }
275:
276:     SET_IDE_RDY_RETR(retries);
277:
278:     blksize = blksize ? blksize : BLKSIZE_1K;
279:     sectors_to_write = MIN(blksize, PAGE_SIZE) / IDE_HD_SECTSIZE;
280:
281:     ident = &ide->drive[drive].ident;
282:     part = ide->drive[drive].part_table;
283:     offset = block2sector(block, blksize, part, minor);
284:
285:     CLI();
286:     lock_resource(&ide->resource);
287:
288:     n = 0;
289:
290:     while(n < sectors_to_write) {
291:         if(ide->drive[drive].flags & DEVICE_HAS_RW_MULTIPLE) {
292:             outport_b(ide->base + IDE_SECCNT, sectors_to_write);
293:             cmd = ATA_WRITE_MULTIPLE_PIO;
294:         } else {
295:             outport_b(ide->base + IDE_SECCNT, 1);
296:             cmd = ATA_WRITE_PIO;
297:         }
298:
299:         if(ide->drive[drive].flags & DEVICE_REQUIRES_LBA) {
300:             outport_b(ide->base + IDE_SECNUM, offset & 0xFF);
301:             outport_b(ide->base + IDE_LCYL, (offset >> 8) & 0xFF);
302:             outport_b(ide->base + IDE_HCYL, (offset >> 16) & 0xFF);
303:             if(ide_drvsel(ide, drive, IDE_LBA_MODE, (offset >> 24) &
0x0F)) {
304:                 printk("WARNING: %s(): %s: drive not ready.\n",
__FUNCTION__, ide->drive[drive].dev_name);
305:                 unlock_resource(&ide->resource);
306:                 return -EIO;
307:             }
308:         } else {
309:             sector2chs(offset, &cyl, &head, &sector, ident);
310:             outport_b(ide->base + IDE_SECNUM, sector);
311:             outport_b(ide->base + IDE_LCYL, cyl);
312:             outport_b(ide->base + IDE_HCYL, (cyl >> 8));
313:             if(ide_drvsel(ide, drive, IDE_CHS_MODE, head)) {
314:                 printk("WARNING: %s(): %s: drive not ready.\n",
__FUNCTION__, ide->drive[drive].dev_name);

```

drivers/block/ide_hd.c

Page 6/9

```

315:                unlock_resource(&ide->resource);
316:                return -EIO;
317:            }
318:        }
319:        outport_b(ide->base + IDE_COMMAND, cmd);
320:        for(r = 0; r < retries; r++) {
321:            status = inport_b(ide->base + IDE_STATUS);
322:            if(!(status & IDE_STAT_BSY) && (status & IDE_STAT_DRQ))
{
323:                break;
324:            }
325:            ide_delay();
326:        }
327:        if(status & IDE_STAT_ERR) {
328:            printk("WARNING: %s(): %s: error on hard disk dev %d,%d
during write.\n", __FUNCTION__, ide->drive[drive].dev_name, MAJOR(dev), MINOR(dev));
329:            printk("\tstatus=0x%x ", status);
330:            ide_error(ide, status);
331:            printk("\tblock %d, sector %d.\n", block, offset);
332:            inport_b(ide->base + IDE_STATUS);          /* clear any pen
ding interrupt */
333:            unlock_resource(&ide->resource);
334:            return -EIO;
335:        }
336:
337:        if(cmd == ATA_WRITE_MULTIPLE_PIO) {
338:            outport_sw(ide->base + IDE_DATA, (void *)buffer, (IDE_HD
_SECTSIZE * sectors_to_write) / sizeof(short int));
339:        } else {
340:            outport_sw(ide->base + IDE_DATA, (void *)buffer, IDE_HD_
SECTSIZE / sizeof(short int));
341:        }
342:        if(ide->channel == IDE_PRIMARY) {
343:            ide0_wait_interrupt = ide->base;
344:            creq.fn = ide0_timer;
345:            creq.arg = 0;
346:            add_callout(&creq, WAIT_FOR_IDE);
347:            if(ide0_wait_interrupt) {
348:                sleep(&irq_ide0, PROC_UNINTERRUPTIBLE);
349:            }
350:            if(ide0_timeout) {
351:                printk("WARNING: %s(): %s: timeout on hard disk
dev %d,%d during write.\n", __FUNCTION__, ide->drive[drive].dev_name, MAJOR
(dev), MINOR
(dev));
352:            } else {
353:                del_callout(&creq);
354:            }
355:        }
356:        if(ide->channel == IDE_SECONDARY) {
357:            idel_wait_interrupt = ide->base;
358:            creq.fn = idel_timer;
359:            creq.arg = 0;
360:            add_callout(&creq, WAIT_FOR_IDE);
361:            if(idel_wait_interrupt) {
362:                sleep(&irq_idel, PROC_UNINTERRUPTIBLE);
363:            }
364:            if(idel_timeout) {
365:                printk("WARNING: %s(): %s: timeout on hard disk
dev %d,%d during write.\n", __FUNCTION__, ide->drive[drive].dev_name, MAJOR
(dev), MINOR
(dev));
366:            } else {
367:                del_callout(&creq);
368:            }
369:        }
370:
371:        inport_b(ide->ctrl + IDE_ALT_STATUS);          /* ignore results */
372:        inport_b(ide->base + IDE_STATUS);              /* clear any pending int

```

drivers/block/ide_hd.c

Page 7/9

```

errupt */
373:         if(cmd == ATA_WRITE_MULTIPLE_PIO) {
374:             break;
375:         }
376:         n++;
377:         offset++;
378:         buffer += IDE_HD_SECTSIZE;
379:     }
380:     inport_b(ide->ctrl + IDE_ALT_STATUS);    /* ignore results */
381:     inport_b(ide->base + IDE_STATUS);        /* clear any pending interrupt */
/
382:     unlock_resource(&ide->resource);
383:     return sectors_to_write * IDE_HD_SECTSIZE;
384: }
385:
386: int ide_hd_ioctl(struct inode *i, int cmd, unsigned long int arg)
387: {
388:     int minor;
389:     int drive;
390:     struct ide *ide;
391:     struct ide_drv_ident *ident;
392:     struct partition *part;
393:     struct hd_geometry *geom;
394:     int errno;
395:
396:     if(!(ide = get_ide_controller(i->rdev))) {
397:         return -EINVAL;
398:     }
399:
400:     minor = MINOR(i->rdev);
401:     drive = get_ide_drive(i->rdev);
402:     if(drive) {
403:         minor &= ~(1 << IDE_SLAVE_MSF);
404:     }
405:
406:     ident = &ide->drive[drive].ident;
407:     part = ide->drive[drive].part_table;
408:
409:     switch(cmd) {
410:         case HDIO_GETGEO:
411:             if((errno = check_user_area(VERIFY_WRITE, (void *)arg, s
sizeof(struct hd_geometry)))) {
412:                 return errno;
413:             }
414:             geom = (struct hd_geometry *)arg;
415:             geom->cylinders = ident->logic_cyls;
416:             geom->heads = (char)ident->logic_heads;
417:             geom->sectors = (char)ident->logic_spt;
418:             geom->start = 0;
419:             if(minor) {
420:                 geom->start = part[minor - 1].startsect;
421:             }
422:             break;
423:         case BLKGETSIZE:
424:             if((errno = check_user_area(VERIFY_WRITE, (void *)arg, s
sizeof(unsigned int)))) {
425:                 return errno;
426:             }
427:             if(!minor) {
428:                 *(int *)arg = (unsigned int)ide->drive[drive].nr
_sects;
429:             } else {
430:                 *(int *)arg = (unsigned int)ide->drive[drive].pa
rt_table[minor - 1].nr_sects;
431:             }
432:             break;
433:         case BLKFLSBUF:

```

drivers/block/ide_hd.c

Page 8/9

```

434:             sync_buffers(i->rdev);
435:             invalidate_buffers(i->rdev);
436:             break;
437:         case BLKRRPART:
438:             read_msdos_partition(i->rdev, part);
439:             assign_minors(i->rdev, ide, part);
440:             break;
441:         default:
442:             return -EINVAL;
443:             break;
444:     }
445:
446:     return 0;
447: }
448:
449: int ide_hd_init(struct ide *ide, int drive)
450: {
451:     int n;
452:     __dev_t rdev;
453:     struct device *d;
454:     struct partition *part;
455:
456:     rdev = 0;
457:     ide->drive[drive].fsop = &ide_hd_driver_fsop;
458:     part = ide->drive[drive].part_table;
459:
460:     if(ide->channel == IDE_PRIMARY) {
461:         if(drive == IDE_MASTER) {
462:             rdev = MKDEV(IDE0_MAJOR, drive);
463:             ide->drive[drive].minor_shift = IDE_MASTER_MSF;
464:             if(!(d = get_device(BLK_DEV, rdev))) {
465:                 return -EINVAL;
466:             }
467:             ((unsigned int *)d->device_data)[0] = ide->drive[drive].
nr_sects / 2;
468:         } else {
469:             rdev = MKDEV(IDE0_MAJOR, 1 << IDE_SLAVE_MSF);
470:             ide->drive[drive].minor_shift = IDE_SLAVE_MSF;
471:             if(!(d = get_device(BLK_DEV, rdev))) {
472:                 return -EINVAL;
473:             }
474:             ((unsigned int *)d->device_data)[1 << IDE_SLAVE_MSF] = i
de->drive[drive].nr_sects / 2;
475:         }
476:     } else if(ide->channel == IDE_SECONDARY) {
477:         if(drive == IDE_MASTER) {
478:             rdev = MKDEV(IDE1_MAJOR, drive);
479:             ide->drive[drive].minor_shift = IDE_MASTER_MSF;
480:             if(!(d = get_device(BLK_DEV, rdev))) {
481:                 return -EINVAL;
482:             }
483:             ((unsigned int *)d->device_data)[0] = ide->drive[drive].
nr_sects / 2;
484:         } else {
485:             rdev = MKDEV(IDE1_MAJOR, 1 << IDE_SLAVE_MSF);
486:             ide->drive[drive].minor_shift = IDE_SLAVE_MSF;
487:             if(!(d = get_device(BLK_DEV, rdev))) {
488:                 return -EINVAL;
489:             }
490:             ((unsigned int *)d->device_data)[1 << IDE_SLAVE_MSF] = i
de->drive[drive].nr_sects / 2;
491:         }
492:     } else {
493:         printk("WARNING: %s(): invalid drive number %d.\n", __FUNCTION__
, drive);
494:         return 1;
495:     }

```

drivers/block/ide_hd.c

Page 9/9

```
496:
497:     read_msdos_partition(rdev, part);
498:     assign_minors(rdev, ide, part);
499:     printk("                                partition summary: ");
500:     for(n = 0; n < NR_PARTITIONS; n++) {
501:         /* status values other than 0x00 and 0x80 are invalid */
502:         if(part[n].status && part[n].status != 0x80) {
503:             continue;
504:         }
505:         if(part[n].type) {
506:             printk("%s%d ", ide->drive[drive].dev_name, n + 1);
507:         }
508:     }
509:     printk("\n");
510:
511:     outport_b(ide->ctrl + IDE_DEV_CTRL, IDE_DEVCTR_NIEN);
512:     if(ide->drive[drive].flags & DEVICE_HAS_RW_MULTIPLE) {
513:         /* read/write in 4KB blocks (8 sectors) by default */
514:         outport_b(ide->base + IDE_SECCNT, PAGE_SIZE / IDE_HD_SECTSIZE);
515:         outport_b(ide->base + IDE_COMMAND, ATA_SET_MULTIPLE_MODE);
516:         ide_wait400ns(ide);
517:         while(inport_b(ide->base + IDE_STATUS) & IDE_STAT_BSY);
518:     }
519:     outport_b(ide->ctrl + IDE_DEV_CTRL, IDE_DEVCTR_DRQ);
520:
521:     return 0;
522: }
```

drivers/block/Makefile

Page 1/1

```
1: # fiwix/drivers/block/Makefile
2: #
3: # Copyright 2018, Jordi Sanfeliu. All rights reserved.
4: # Distributed under the terms of the Fiwix License.
5: #
6:
7: .S.o:
8:     $(CC) -traditional -I$(INCLUDE) -c -o $@ $<
9: .c.o:
10:     $(CC) $(CFLAGS) -c -o $@ $<
11:
12: OBJS = dma.o floppy.o part.o ide.o ide_hd.o ide_cd.o ramdisk.o
13:
14: block:  $(OBJS)
15:           $(LD) $(LDFLAGS) -r $(OBJS) -o block.o
16:
17: clean:
18:           rm -f *.o
19:
```

drivers/block/part.c

Page 1/1

```
1: /*
2:  * fiwix/drivers/block/part.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/ide.h>
9: #include <fiwix/ide_hd.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/part.h>
12: #include <fiwix/mm.h>
13: #include <fiwix/errno.h>
14: #include <fiwix/stdio.h>
15: #include <fiwix/string.h>
16:
17: int read_msdos_partition(__dev_t dev, struct partition *part)
18: {
19:     char *buffer;
20:
21:     if(!(buffer = (void *)kmalloc())) {
22:         return -ENOMEM;
23:     }
24:
25:     if(ide_hd_read(dev, PARTITION_BLOCK, buffer, BLKSIZE_1K) <= 0) {
26:         printk("WARNING: %s(): unable to read partition block in device
%d,%d.\n", __FUNCTION__, MAJOR(dev), MINOR(dev));
27:         kfree((unsigned int)buffer);
28:         return -EIO;
29:     }
30:
31:     memcpy_b(part, (void *) (buffer + MBR_CODE_SIZE), sizeof(struct partition
) * NR_PARTITIONS);
32:     kfree((unsigned int)buffer);
33:     return 0;
34: }
```


drivers/block/ramdisk.c

Page 1/3

```

1:  /*
2:  *  fiwix/drivers/block/ramdisk.c
3:  *
4:  *  Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  *  Distributed under the terms of the Fiwix License.
6:  */
7:
8:  #include <fiwix/kernel.h>
9:  #include <fiwix/ramdisk.h>
10: #include <fiwix/ioctl.h>
11: #include <fiwix/devices.h>
12: #include <fiwix/part.h>
13: #include <fiwix/fs.h>
14: #include <fiwix/errno.h>
15: #include <fiwix/mm.h>
16: #include <fiwix/stdio.h>
17: #include <fiwix/string.h>
18:
19: struct ramdisk ramdisk_table[RAMDISK_MINORS];
20: static unsigned int rd_sizes[256];
21:
22: static struct fs_operations ramdisk_driver_fsop = {
23:     0,
24:     0,
25:
26:     ramdisk_open,
27:     ramdisk_close,
28:     NULL,                /* read */
29:     NULL,                /* write */
30:     ramdisk_ioctl,
31:     ramdisk_lseek,
32:     NULL,                /* readdir */
33:     NULL,                /* mmap */
34:     NULL,                /* select */
35:
36:     NULL,                /* readlink */
37:     NULL,                /* followlink */
38:     NULL,                /* bmap */
39:     NULL,                /* lockup */
40:     NULL,                /* rmdir */
41:     NULL,                /* link */
42:     NULL,                /* unlink */
43:     NULL,                /* symlink */
44:     NULL,                /* mkdir */
45:     NULL,                /* mknod */
46:     NULL,                /* truncate */
47:     NULL,                /* create */
48:     NULL,                /* rename */
49:
50:     ramdisk_read,
51:     ramdisk_write,
52:
53:     NULL,                /* read_inode */
54:     NULL,                /* write_inode */
55:     NULL,                /* ialloc */
56:     NULL,                /* ifree */
57:     NULL,                /* statfs */
58:     NULL,                /* read_superblock */
59:     NULL,                /* remount_fs */
60:     NULL,                /* write_superblock */
61:     NULL,                /* release_superblock */
62: };
63:
64: static struct device ramdisk_device = {
65:     "ramdisk",
66:     RAMDISK_MAJOR,
67:     { 0, 0, 0, 0, 0, 0, 0, 0, 0 },

```

drivers/block/ramdisk.c

Page 2/3

```

68:         BLKSIZE_1K,
69:         &rd_sizes,
70:         &ramdisk_driver_fsop,
71:         NULL
72: };
73:
74: static struct ramdisk * get_ramdisk(int minor)
75: {
76:     if(TEST_MINOR(ramdisk_device.minors, minor)) {
77:         return &ramdisk_table[minor];
78:     }
79:     return NULL;
80: }
81:
82: int ramdisk_open(struct inode *i, struct fd *fd_table)
83: {
84:     if(!get_ramdisk(MINOR(i->rdev))) {
85:         return -ENXIO;
86:     }
87:     return 0;
88: }
89:
90: int ramdisk_close(struct inode *i, struct fd *fd_table)
91: {
92:     if(!get_ramdisk(MINOR(i->rdev))) {
93:         return -ENXIO;
94:     }
95:     return 0;
96: }
97:
98: int ramdisk_read(__dev_t dev, __blk_t block, char *buffer, int blksize)
99: {
100:     int size;
101:     __off_t offset;
102:     struct ramdisk *ramdisk;
103:
104:     if(!(ramdisk = get_ramdisk(MINOR(dev)))) {
105:         return -ENXIO;
106:     }
107:
108:     size = rd_sizes[MINOR(dev)] * 1024;
109:     offset = block * blksize;
110:     if(offset > size) {
111:         printk("%s(): block %d is beyond the size of the ramdisk.\n", __
FUNCTION__, block);
112:         return -EIO;
113:     }
114:     blksize = MIN(blksize, size - offset);
115:     memcpy_b((void *)buffer, ramdisk->addr + offset, blksize);
116:     return blksize;
117: }
118:
119: int ramdisk_write(__dev_t dev, __blk_t block, char *buffer, int blksize)
120: {
121:     int size;
122:     __off_t offset;
123:     struct ramdisk *ramdisk;
124:
125:     if(!(ramdisk = get_ramdisk(MINOR(dev)))) {
126:         return -ENXIO;
127:     }
128:
129:     size = rd_sizes[MINOR(dev)] * 1024;
130:     offset = block * blksize;
131:     if(offset > size) {
132:         printk("%s(): block %d is beyond the size of the ramdisk.\n", __
FUNCTION__, block);

```

drivers/block/ramdisk.c

Page 3/3

```

133:             return -EIO;
134:         }
135:         blksize = MIN(blksize, size - offset);
136:         memcpy_b(ramdisk->addr + offset, buffer, blksize);
137:         return blksize;
138:     }
139:
140: int ramdisk_ioctl(struct inode *i, int cmd, unsigned long int arg)
141: {
142:     struct hd_geometry *geom;
143:     int errno;
144:
145:     if(!get_ramdisk(MINOR(i->rdev))) {
146:         return -ENXIO;
147:     }
148:
149:     switch(cmd) {
150:         case HDIO_GETGEO:
151:             if((errno = check_user_area(VERIFY_WRITE, (void *)arg, s
izeof(struct hd_geometry)))) {
152:                 return errno;
153:             }
154:             geom = (struct hd_geometry *)arg;
155:             geom->heads = 63;
156:             geom->sectors = 16;
157:             geom->cylinders = rd_sizes[MINOR(i->rdev)] * 1024 / BPS;
158:             geom->cylinders /= (geom->heads * geom->sectors);
159:             geom->start = 0;
160:             break;
161:         case BLKRRPART:
162:             break;
163:         case BLKGETSIZE:
164:             if((errno = check_user_area(VERIFY_WRITE, (void *)arg, s
izeof(unsigned int)))) {
165:                 return errno;
166:             }
167:             *(int *)arg = rd_sizes[MINOR(i->rdev)] * 2;
168:             break;
169:         default:
170:             return -EINVAL;
171:     }
172:     return 0;
173: }
174:
175: int ramdisk_lseek(struct inode *i, __off_t offset)
176: {
177:     return offset;
178: }
179:
180: void ramdisk_init(void)
181: {
182:     int n;
183:     struct ramdisk *ramdisk;
184:
185:     if(!_noramdisk) {
186:         for(n = 0; n < RAMDISK_MINORS; n++) {
187:             SET_MINOR(ramdisk_device.minors, n);
188:             rd_sizes[n] = _ramdisksize;
189:             ramdisk = get_ramdisk(n);
190:             printk("ram%d      0x%08X-0x%08X %d RAMdisk(s) of %dKB s
ize, %dKB blocksize\n", n, ramdisk->addr, ramdisk->addr + (_ramdisksize * 1024), RAMDIS
K_MINORS, _ramdisksize, BLKSIZE_1K / 1024);
191:         }
192:         register_device(BLK_DEV, &ramdisk_device);
193:     }
194: }

```

drivers/char/console.c

Page 1/18

```

1:  /*
2:  * fiwix/drivers/char/console.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8:  #include <fiwix/asm.h>
9:  #include <fiwix/kernel.h>
10: #include <fiwix/ctype.h>
11: #include <fiwix/console.h>
12: #include <fiwix/devices.h>
13: #include <fiwix/tty.h>
14: #include <fiwix/keyboard.h>
15: #include <fiwix/sleep.h>
16: #include <fiwix/pit.h>
17: #include <fiwix/timer.h>
18: #include <fiwix/process.h>
19: #include <fiwix/mm.h>
20: #include <fiwix/sched.h>
21: #include <fiwix/kd.h>
22: #include <fiwix/stdio.h>
23: #include <fiwix/string.h>
24: #include <fiwix/fbcon.h>
25:
26: #define CSI_J_CUR2END    0    /* clear from cursor to end of screen */
27: #define CSI_J_STA2CUR    1    /* clear from start of screen to cursor */
28: #define CSI_J_SCREEN    2    /* clear entire screen */
29:
30: #define CSI_K_CUR2END    0    /* clear from cursor to end of line */
31: #define CSI_K_STA2CUR    1    /* clear from start of line to cursor */
32: #define CSI_K_LINE      2    /* clear entire line */
33:
34: #define CSE              vc->esc = 0    /* Code Set End */
35:
36: /* VT100 ID string generated by <ESC>Z or <ESC>[c */
37: #define VT100ID          "\033[?1;2c"
38:
39: /* VT100 report status generated by <ESC>[5n */
40: #define DEVICE_OK        "\033[0n"
41: #define DEVICE_NOT_OK    "\033[3n"
42:
43: #define SCREEN_SIZE      (video.columns * video.lines)
44: #define VC_BUF_LINES     (video.lines * SCREENS_LOG)
45:
46:
47: short int current_cons;
48:
49: struct video_parms video;
50: struct vconsole vc[NR_VCONSOLES + 1];
51:
52: static struct fs_operations tty_driver_fsop = {
53:     0,
54:     0,
55:
56:     tty_open,
57:     tty_close,
58:     tty_read,
59:     tty_write,
60:     tty_ioctl,
61:     tty_lseek,
62:     NULL,          /* readdir */
63:     NULL,          /* mmap */
64:     tty_select,
65:
66:     NULL,          /* readlink */
67:     NULL,          /* followlink */

```

drivers/char/console.c

Page 2/18

```

68:         NULL,                /* bmap */
69:         NULL,                /* lookup */
70:         NULL,                /* rmdir */
71:         NULL,                /* link */
72:         NULL,                /* unlink */
73:         NULL,                /* symlink */
74:         NULL,                /* mkdir */
75:         NULL,                /* mknod */
76:         NULL,                /* truncate */
77:         NULL,                /* create */
78:         NULL,                /* rename */
79:
80:         NULL,                /* read_block */
81:         NULL,                /* write_block */
82:
83:         NULL,                /* read_inode */
84:         NULL,                /* write_inode */
85:         NULL,                /* ialloc */
86:         NULL,                /* ifree */
87:         NULL,                /* statfs */
88:         NULL,                /* read_superblock */
89:         NULL,                /* remount_fs */
90:         NULL,                /* write_superblock */
91:         NULL,                /* release_superblock */
92: };
93:
94: static struct device tty_device = {
95:     "vconsole",
96:     VCONSOLES_MAJOR,
97:     { 0, 0, 0, 0, 0, 0, 0, 0 },
98:     0,
99:     NULL,
100:    &tty_driver_fsop,
101:    NULL
102: };
103:
104: static struct device console_device = {
105:     "console",
106:     SYSCON_MAJOR,
107:     { 0, 0, 0, 0, 0, 0, 0, 0 },
108:     0,
109:     NULL,
110:    &tty_driver_fsop,
111:    NULL
112: };
113:
114: unsigned short int ansi_color_table[] = {
115:     COLOR_BLACK,
116:     COLOR_RED,
117:     COLOR_GREEN,
118:     COLOR_BROWN,
119:     COLOR_BLUE,
120:     COLOR_MAGENTA,
121:     COLOR_CYAN,
122:     COLOR_WHITE
123: };
124:
125: static int is_vconsole(__dev_t dev)
126: {
127:     if (MAJOR(dev) == VCONSOLES_MAJOR && MINOR(dev) <= NR_VCONSOLES) {
128:         return 1;
129:     }
130:
131:     return 0;
132: }
133:
134: static void adjust(struct vconsole *vc, int x, int y)

```

drivers/char/console.c

Page 3/18

```

135: {
136:     if(x < 0) {
137:         x = 0;
138:     }
139:     if(x >= vc->columns) {
140:         x = vc->columns - 1;
141:     }
142:     if(y < 0) {
143:         y = 0;
144:     }
145:     if(y >= vc->lines) {
146:         y = vc->lines - 1;
147:     }
148:     vc->x = x;
149:     vc->y = y;
150: }
151:
152: static void cr(struct vconsole *vc)
153: {
154:     vc->x = 0;
155: }
156:
157: static void lf(struct vconsole *vc)
158: {
159:     if(vc->y == vc->lines) {
160:         video.scroll_screen(vc, 0, SCROLL_UP);
161:     } else {
162:         vc->y++;
163:     }
164: }
165:
166: static void ri(struct vconsole *vc)
167: {
168:     video.scroll_screen(vc, 0, SCROLL_DOWN);
169: }
170:
171: static void csi_J(struct vconsole *vc, int mode)
172: {
173:     int from, count;
174:
175:     switch(mode) {
176:         case CSI_J_CUR2END:      /* Erase Down <ESC>[J */
177:             from = (vc->y * vc->columns) + vc->x;
178:             count = vc->columns - vc->x;
179:             video.write_screen(vc, from, count, vc->color_attr);
180:             from = ((vc->y + 1) * vc->columns);
181:             count = SCREEN_SIZE - from;
182:             break;
183:         case CSI_J_STA2CUR:      /* Erase Up <ESC>[1J */
184:             from = vc->y * vc->columns;
185:             count = vc->x + 1;
186:             video.write_screen(vc, from, count, vc->color_attr);
187:             from = 0;
188:             count = vc->y * vc->columns;
189:             break;
190:         case CSI_J_SCREEN:      /* Erase Screen <ESC>[2J */
191:             from = 0;
192:             count = SCREEN_SIZE;
193:             break;
194:         default:
195:             return;
196:     }
197:     video.write_screen(vc, from, count, vc->color_attr);
198: }
199:
200: static void csi_K(struct vconsole *vc, int mode)
201: {

```

drivers/char/console.c

Page 4/18

```

202:     int from, count;
203:
204:     switch(mode) {
205:         case CSI_K_CUR2END:      /* Erase End of Line <ESC>[K */
206:             from = (vc->y * vc->columns) + vc->x;
207:             count = vc->columns - vc->x;
208:             break;
209:         case CSI_K_STA2CUR:      /* Erase Start of Line <ESC>[1K */
210:             from = vc->y * vc->columns;
211:             count = vc->x + 1;
212:             break;
213:         case CSI_K_LINE:         /* Erase Line <ESC>[2K */
214:             from = vc->y * vc->columns;
215:             count = vc->columns;
216:             break;
217:         default:
218:             return;
219:     }
220:     video.write_screen(vc, from, count, vc->color_attr);
221: }
222:
223: static void csi_X(struct vconsole *vc, int count)
224: {
225:     int from;
226:
227:     from = (vc->y * vc->columns) + vc->x;
228:     count = count > (vc->columns - vc->x) ? vc->columns - vc->x : count;
229:     video.write_screen(vc, from, count, vc->color_attr);
230: }
231:
232: static void csi_L(struct vconsole *vc, int count)
233: {
234:     if(count > (vc->lines - vc->top)) {
235:         count = vc->lines - vc->top;
236:     }
237:     while(count--) {
238:         video.scroll_screen(vc, vc->y, SCROLL_DOWN);
239:     }
240: }
241:
242: static void csi_M(struct vconsole *vc, int count)
243: {
244:     if(count > (vc->lines - vc->top)) {
245:         count = vc->lines - vc->top;
246:     }
247:     while(count--) {
248:         video.scroll_screen(vc, vc->y, SCROLL_UP);
249:     }
250: }
251:
252: static void csi_P(struct vconsole *vc, int count)
253: {
254:     if(count > vc->columns) {
255:         count = vc->columns;
256:     }
257:     while(count--) {
258:         video.delete_char(vc);
259:     }
260: }
261:
262: static void csi_at(struct vconsole *vc, int count)
263: {
264:     if(count > vc->columns) {
265:         count = vc->columns;
266:     }
267:     while(count--) {
268:         video.insert_char(vc);

```

drivers/char/console.c

Page 5/18

```

269:         }
270:     }
271:
272:     static void default_color_attr(struct vconsole *vc)
273:     {
274:         vc->color_attr = DEF_MODE;
275:         vc->bold = 0;
276:         vc->underline = 0;
277:         vc->blink = 0;
278:         vc->reverse = 0;
279:     }
280:
281:     /* Select Graphic Rendition */
282:     static void csi_m(struct vconsole *vc)
283:     {
284:         int n;
285:
286:         if(vc->reverse) {
287:             vc->color_attr = ((vc->color_attr & 0x7000) >> 4) | ((vc->color_
attr & 0x0700) << 4) | (vc->color_attr & 0x8800);
288:         }
289:
290:         for(n = 0; n < vc->nparms; n++) {
291:             switch(vc->parms[n]) {
292:                 case SGR_DEFAULT:
293:                     default_color_attr(vc);
294:                     break;
295:                 case SGR_BOLD:
296:                     vc->bold = 1;
297:                     break;
298:                 case SGR_BLINK:
299:                     vc->blink = 1;
300:                     break;
301:                 case SGR_REVERSE:
302:                     vc->reverse = 1;
303:                     break;
304:                 /* normal intensity */
305:                 case 21:
306:                 case 22:
307:                     vc->bold = 0;
308:                     break;
309:                 case SGR_BLINK_OFF:
310:                     vc->blink = 0;
311:                     break;
312:                 case SGR_REVERSE_OFF:
313:                     vc->reverse = 0;
314:                     break;
315:                 case SGR_BLACK_FG:
316:                 case SGR_RED_FG:
317:                 case SGR_GREEN_FG:
318:                 case SGR_BROWN_FG:
319:                 case SGR_BLUE_FG:
320:                 case SGR_MAGENTA_FG:
321:                 case SGR_CYAN_FG:
322:                 case SGR_WHITE_FG:
323:                     vc->color_attr = (vc->color_attr & 0xF8FF) | (an
si_color_table[vc->parms[n] - 30]);
324:                     break;
325:                 case SGR_DEFAULT_FG_U_ON:
326:                 case SGR_DEFAULT_FG_U_OFF:
327:                     /* not supported yet */
328:                     break;
329:                 case SGR_BLACK_BG:
330:                 case SGR_RED_BG:
331:                 case SGR_GREEN_BG:
332:                 case SGR_BROWN_BG:
333:                 case SGR_BLUE_BG:

```


drivers/char/console.c

Page 6/18

```

334:             case SGR_MAGENTA_BG:
335:             case SGR_CYAN_BG:
336:             case SGR_WHITE_BG:
337:                 vc->color_attr = (vc->color_attr & 0x8FFF) | ((a
nsi_color_table[vc->parms[n] - 40]) << 4);
338:                 break;
339:             case SGR_DEFAULT_BG:
340:                 /* not supported yet */
341:                 break;
342:         }
343:     }
344:     if(vc->bold) {
345:         vc->color_attr |= 0x0800;
346:     } else {
347:         vc->color_attr &= ~0x0800;
348:     }
349:     if(vc->blink) {
350:         vc->color_attr |= 0x8000;
351:     } else {
352:         vc->color_attr &= ~0x8000;
353:     }
354:     if(vc->reverse) {
355:         vc->color_attr = ((vc->color_attr & 0x7000) >> 4) | ((vc->color_
attr & 0x0700) << 4) | (vc->color_attr & 0x8800);
356:     }
357: }
358:
359: static void init_vt(struct vconsole *vc)
360: {
361:     vc->vt_mode.mode = VT_AUTO;
362:     vc->vt_mode.waitv = 0;
363:     vc->vt_mode.relsig = 0;
364:     vc->vt_mode.acqsig = 0;
365:     vc->vt_mode.frsig = 0;
366:     vc->vc_mode = KD_TEXT;
367:     vc->tty->pid = 0;
368:     vc->switchto_tty = -1;
369: }
370:
371: static void insert_seq(struct tty *tty, char *buf, int count)
372: {
373:     while(count-- > 0) {
374:         tty_queue_putchar(tty, &tty->read_q, *(buf++));
375:     }
376:     tty->input(tty);
377: }
378:
379: static void vcbuf_scroll_up(void)
380: {
381:     memcpy_w(vcbuf, vcbuf + video.columns, (VC_BUF_SIZE - video.columns) * 2
);
382: }
383:
384: static void vcbuf_refresh(struct vconsole *vc)
385: {
386:     short int *screen;
387:
388:     screen = (short int *)vc->screen;
389:     memset_w(vcbuf, BLANK_MEM, VC_BUF_SIZE);
390:     memcpy_w(vcbuf, screen, SCREEN_SIZE);
391: }
392:
393: static void echo_char(struct vconsole *vc, unsigned char *buf, unsigned int coun
t)
394: {
395:     unsigned char ch;
396:     unsigned long int flags;

```

drivers/char/console.c

Page 7/18

```

397:
398:     SAVE_FLAGS(flags); CLI();
399:     if(vc->flags & CONSOLE_HAS_FOCUS) {
400:         if(video.buf_top) {
401:             video.restore_screen(vc);
402:             video.show_cursor(vc, ON);
403:             video.buf_top = 0;
404:         }
405:     }
406:
407:     while(count--) {
408:         ch = *buf++;
409:         if(ch == NULL) {
410:             continue;
411:
412:         } else if(ch == '\b') {
413:             if(vc->x) {
414:                 vc->x--;
415:             }
416:
417:         } else if(ch == '\a') {
418:             vconsole_beep();
419:
420:         } else if(ch == '\r') {
421:             cr(vc);
422:
423:         } else if(ch == '\n') {
424:             cr(vc);
425:             vc->y++;
426:             if(vc->flags & CONSOLE_HAS_FOCUS) {
427:                 video.buf_y++;
428:             }
429:
430:         } else if(ch == '\t') {
431:             while(vc->x < (vc->columns - 1)) {
432:                 if(vc->tty->tab_stop[+vc->x]) {
433:                     break;
434:                 }
435:             }
436:             /* vc->x += TAB_SIZE - (vc->x % TAB_SIZE); */
437:             vc->check_x = 1;
438:
439:         } else {
440:             if((vc->x == vc->columns - 1) && vc->check_x) {
441:                 vc->x = 0;
442:                 vc->y++;
443:                 if(vc->flags & CONSOLE_HAS_FOCUS) {
444:                     video.buf_y++;
445:                 }
446:             }
447:             if(vc->y >= vc->lines) {
448:                 video.scroll_screen(vc, 0, SCROLL_UP);
449:                 vc->y--;
450:             }
451:             video.put_char(vc, ch);
452:             if(vc->x < vc->columns - 1) {
453:                 vc->check_x = 0;
454:                 vc->x++;
455:             } else {
456:                 vc->check_x = 1;
457:             }
458:         }
459:         if(vc->y >= vc->lines) {
460:             video.scroll_screen(vc, 0, SCROLL_UP);
461:             vc->y--;
462:         }
463:         if(vc->flags & CONSOLE_HAS_FOCUS) {

```

drivers/char/console.c

Page 8/18

```

464:             if(video.buf_y >= VC_BUF_LINES) {
465:                 vcbuf_scroll_up();
466:                 video.buf_y--;
467:             }
468:         }
469:     }
470:     video.update_curpos(vc);
471:     RESTORE_FLAGS(flags);
472: }
473:
474: void vconsole_reset(struct tty *tty)
475: {
476:     int n;
477:     struct vconsole *vc;
478:
479:     vc = (struct vconsole *)tty->driver_data;
480:
481:     vc->top = 0;
482:     vc->lines = video.lines;
483:     vc->columns = video.columns;
484:     vc->check_x = 0;
485:     vc->led_status = 0;
486:     set_leds(vc->led_status);
487:     vc->scrlock = vc->numlock = vc->capslock = 0;
488:     vc->esc = vc->sbracket = vc->semicolon = vc->question = 0;
489:     vc->parmv1 = vc->parmv2 = 0;
490:     vc->nparms = 0;
491:     memset_b(vc->parms, 0, sizeof(vc->parms));
492:     default_color_attr(vc);
493:     vc->insert_mode = 0;
494:     vc->saved_x = vc->saved_y = 0;
495:
496:     for(n = 0; n < MAX_TAB_COLS; n++) {
497:         if(!(n % TAB_SIZE)) {
498:             vc->tty->tab_stop[n] = 1;
499:         } else {
500:             vc->tty->tab_stop[n] = 0;
501:         }
502:     }
503:
504:     termios_reset(tty);
505:     vc->tty->winsize.ws_row = vc->lines - vc->top;
506:     vc->tty->winsize.ws_col = vc->columns;
507:     vc->tty->winsize.ws_xpixel = 0;
508:     vc->tty->winsize.ws_ypixel = 0;
509:     vc->tty->lnext = 0;
510:
511:     init_vt(vc);
512:     vc->flags &= ~CONSOLE_BLANKED;
513:     video.update_curpos(vc);
514: }
515:
516: void vconsole_write(struct tty *tty)
517: {
518:     int n;
519:     unsigned char ch;
520:     int numeric;
521:     struct vconsole *vc;
522:
523:     vc = (struct vconsole *)tty->driver_data;
524:
525:     if(vc->flags & CONSOLE_HAS_FOCUS) {
526:         if(video.buf_top) {
527:             video.restore_screen(vc);
528:             video.buf_top = 0;
529:             video.show_cursor(vc, ON);
530:             video.update_curpos(vc);

```

drivers/char/console.c

Page 9/18

```

531:         }
532:     }
533:
534:     ch = numeric = 0;
535:
536:     while(!vc->scrlock && tty->write_q.count > 0) {
537:         ch = tty_queue_getchar(&tty->write_q);
538:
539:         if(vc->esc) {
540:             if(vc->sbracket) {
541:                 if(IS_NUMERIC(ch)) {
542:                     numeric = 1;
543:                     if(vc->semicolon) {
544:                         vc->parmv2 *= 10;
545:                         vc->parmv2 += ch - '0';
546:                     } else {
547:                         vc->parmv1 *= 10;
548:                         vc->parmv1 += ch - '0';
549:                     }
550:                     vc->parms[vc->nparms] *= 10;
551:                     vc->parms[vc->nparms] += ch - '0';
552:                     continue;
553:                 }
554:                 switch(ch) {
555:                     case ';':
556:                         vc->semicolon = 1;
557:                         vc->parmv2 = 0;
558:                         vc->nparms++;
559:                         continue;
560:                     case '?':
561:                         vc->question = 1;
562:                         continue;
563:                     case '@': /* Insert Character(s) <
ESC>[ n @ */
564:                         vc->parmv1 = !vc->parmv1 ? 1 : v
c->parmv1;
565:                         csi_at(vc, vc->parmv1);
566:                         CSE;
567:                         continue;
568:                     case 'A': /* Cursor Up <ESC>[ n A
*/
569:                         vc->parmv1 = !vc->parmv1 ? 1 : v
c->parmv1;
570:                         adjust(vc, vc->x, vc->y - vc->pa
rmv1);
571:                         CSE;
572:                         continue;
573:                     case 'B': /* Cursor Down <ESC>[ n
B */
574:                         vc->parmv1 = !vc->parmv1 ? 1 : v
c->parmv1;
575:                         adjust(vc, vc->x, vc->y + vc->pa
rmv1);
576:                         CSE;
577:                         continue;
578:                     case 'C': /* Cursor Forward <ESC>[
n C */
579:                         vc->parmv1 = !vc->parmv1 ? 1 : v
c->parmv1;
580:                         adjust(vc, vc->x + vc->parmv1, v
c->y);
581:                         CSE;
582:                         continue;
583:                     case 'D': /* Cursor Backward <ESC>
[ n D */
584:                         vc->parmv1 = !vc->parmv1 ? 1 : v
c->parmv1;

```

drivers/char/console.c

Page 10/18

```

585:                                adjust(vc, vc->x - vc->parmv1, v
c->y);
586:                                CSE;
587:                                continue;
588:                                case 'E': /* Cursor Next Line(s) <
ESC>[ n E */
589:                                vc->parmv1 = !vc->parmv1 ? 1 : v
c->parmv1;
590:                                adjust(vc, 0, vc->y + vc->parmv1
);
591:                                CSE;
592:                                continue;
593:                                case 'F': /* Cursor Previous Line(
s) <ESC>[ n F */
594:                                vc->parmv1 = !vc->parmv1 ? 1 : v
c->parmv1;
595:                                adjust(vc, 0, vc->y - vc->parmv1
);
596:                                CSE;
597:                                continue;
598:                                case 'G': /* Cursor Horizontal Pos
ition <ESC>[ n G */
599:                                case '`':
600:                                vc->parmv1 = vc->parmv1 ? vc->pa
rmv1 - 1 : vc->parmv1;
601:                                adjust(vc, vc->parmv1, vc->y);
602:                                CSE;
603:                                continue;
604:                                case 'H': /* Cursor Home <ESC>[ RO
W ; COLUMN H */
605:                                case 'f': /* Horizontal Vertical P
osition <ESC>[ ROW ; COLUMN f */
606:                                vc->parmv1 = vc->parmv1 ? vc->pa
rmv1 - 1 : vc->parmv1;
607:                                vc->parmv2 = vc->parmv2 ? vc->pa
rmv2 - 1 : vc->parmv2;
608:                                adjust(vc, vc->parmv2, vc->parmv
1);
609:                                CSE;
610:                                continue;
611:                                case 'I': /* Cursor Forward Tabula
tion <ESC>[ n I */
612:                                vc->parmv1 = !vc->parmv1 ? 1 : v
c->parmv1;
613:                                while(vc->parmv1--){
614:                                    while(vc->x < (vc->column
s - 1)) {
615:                                        if(vc->tty->tab_
stop[++vc->x]) {
616:                                            break;
617:                                        }
618:                                    }
619:                                }
620:                                adjust(vc, vc->x, vc->y);
621:                                CSE;
622:                                continue;
623:                                case 'J': /* Erase (Down/Up/Screen
) <ESC>[J */
624:                                csi_J(vc, vc->parmv1);
625:                                CSE;
626:                                continue;
627:                                case 'K': /* Erase (End of/Start o
f) Line <ESC>[K */
628:                                csi_K(vc, vc->parmv1);
629:                                CSE;
630:                                continue;
631:                                case 'L': /* Insert Line(s) <ESC>[

```

drivers/char/console.c

Page 11/18

```

n L */
632:                                vc->parmv1 = !vc->parmv1 ? 1 : v
c->parmv1;
633:                                csi_L(vc, vc->parmv1);
634:                                CSE;
635:                                continue;
636:                                case 'M': /* Delete Line(s) <ESC>[
n M */
637:                                vc->parmv1 = !vc->parmv1 ? 1 : v
c->parmv1;
638:                                csi_M(vc, vc->parmv1);
639:                                CSE;
640:                                continue;
641:                                case 'P': /* Delete Character(s) <
ESC>[ n P */
642:                                vc->parmv1 = !vc->parmv1 ? 1 : v
c->parmv1;
643:                                csi_P(vc, vc->parmv1);
644:                                CSE;
645:                                continue;
646:                                case 'S': /* Scroll Up <ESC>[ n S
*/
647:                                vc->parmv1 = !vc->parmv1 ? 1 : v
c->parmv1;
648:                                while(vc->parmv1--) {
649:                                    video.scroll_screen(vc,
0, SCROLL_UP);
650:                                }
651:                                CSE;
652:                                continue;
653:                                case 'T': /* Scroll Down <ESC>[ n
T */
654:                                vc->parmv1 = !vc->parmv1 ? 1 : v
c->parmv1;
655:                                while(vc->parmv1--) {
656:                                    video.scroll_screen(vc,
0, SCROLL_DOWN);
657:                                }
658:                                CSE;
659:                                continue;
660:                                case 'X': /* Erase Character(s) <E
SC>[ n X */
661:                                vc->parmv1 = !vc->parmv1 ? 1 : v
c->parmv1;
662:                                csi_X(vc, vc->parmv1);
663:                                CSE;
664:                                continue;
665:                                case 'c': /* Query Device Code <ES
C>[c */
666:                                if(!numeric) {
667:                                    insert_seq(tty, VT100ID,
7);
668:                                }
669:                                CSE;
670:                                continue;
671:                                case 'd': /* Cursor Vertical Posit
ion <ESC>[ n d */
672:                                vc->parmv1 = vc->parmv1 ? vc->pa
rmv1 - 1 : vc->parmv1;
673:                                adjust(vc, vc->x, vc->parmv1);
674:                                CSE;
675:                                continue;
676:                                case 'g':
677:                                    switch(vc->parmv1) {
678:                                        case 0: /* Clear Tab <ES
C>[g */
679:                                            vc->tty->tab_sto

```

drivers/char/console.c

Page 12/18

```

p[vc->x] = 0;
680:                                     break;
681:                                     case 3: /* Clear All Tab
s <ESC>[3g */
682:                                     case 5: /* Clear All Tab
s <ESC>[5g */
683:                                     for(n = 0; n < M
AX_TAB_COLS; n++)
684:                                     vc->tty-
>tab_stop[n] = 0;
685:                                     break;
686:                                     }
687:                                     CSE;
688:                                     continue;
689: case 'h':
690:     if(vc->question) {
691:         switch(vc->parmv1) {
692:             /* DEC modes */
693:             case 25: /* Swit
ch Cursor Visible <ESC>[?25h */
694:                                     video.sh
ow_cursor(vc, ON);
695:                                     break;
696:                                     case 4:
697:                                     vc->inse
rt_mode = ON; /* not used */
698:                                     break;
699:                                     }
700:                                     }
701:                                     CSE;
702:                                     continue;
703: case 'l':
704:     if(vc->question) {
705:         switch(vc->parmv1) {
706:             /* DEC modes */
707:             case 25: /* Swit
ch Cursor Invisible <ESC>[?25l */
708:                                     video.sh
ow_cursor(vc, OFF);
709:                                     break;
710:                                     case 4:
711:                                     vc->inse
rt_mode = OFF; /* not used */
712:                                     break;
713:                                     }
714:                                     }
715:                                     CSE;
716:                                     continue;
717: case 'm': /* Select Graphic Rendit
ion <ESC> n ... m */
718:                                     vc->nparms++;
719:                                     csi_m(vc);
720:                                     CSE;
721:                                     continue;
722: case 'n':
723:     if(!vc->question) {
724:         switch(vc->parmv1) {
725:             case 5: /* Query
Device Status <ESC>[5n */
726:                                     insert_s
eq(tty, DEVICE_OK, 4);
727:                                     break;
728:                                     case 6: /* Query
Cursor Position <ESC>[6n */
729:                                     {
730:                                     }
char curpos[8];

```

drivers/char/console.c

Page 13/18

```

731:
char len;
732:
len = sprintk(curpos, "\033[%d;%dR", vc->y, vc->x);
733:
insert_seq(tty, curpos, len);
734:
735:
736:
737:
738:
739:
740:
s <ESC>[r / <ESC>[{start};{end}r */
741:
742:
743:
744:
745:
;
746:
747:
->parmv2 <= video.lines) {
748:
;
749:
750:
751:
752:
753:
754:
/
755:
756:
757:
758:
759:
u */
760:
761:
762:
763:
764:
765:
766:
767:
768:
769:
770:
771:
772:
773:
774:
775:
776:
->parms));
777:
778:
ESC>7 */
779:
780:
781:
782:
783:
s <ESC>8 */
784:
785:

```

```

}
break;
}
}
CSE;
continue;
case 'r': /* Top and Bottom Margin
    if(!vc->parmv1) {
        vc->parmv1++;
    }
    if(!vc->parmv2) {
        vc->parmv2 = video.lines
    }
    if(vc->parmv1 < vc->parmv2 && vc
        vc->top = vc->parmv1 - 1
        vc->lines = vc->parmv2;
        adjust(vc, 0, 0);
    }
    CSE;
    continue;
case 's': /* Save Cursor <ESC>[s *
    vc->saved_x = vc->x;
    vc->saved_y = vc->y;
    CSE;
    continue;
case 'u': /* Restore Cursor <ESC>[
    vc->x = vc->saved_x;
    vc->y = vc->saved_y;
    CSE;
    continue;
default:
    CSE;
    break;
}
} else {
    switch(ch) {
        case '[':
            vc->sbracket = 1;
            vc->semicolon = 0;
            vc->question = 0;
            vc->parmv1 = vc->parmv2 = 0;
            vc->nparms = 0;
            memset_b(vc->parms, 0, sizeof(vc
            continue;
        case '7': /* Save Cursor & Attrs <
            vc->saved_x = vc->x;
            vc->saved_y = vc->y;
            CSE;
            continue;
        case '8': /* Restore Cursor & Attr
            vc->x = vc->saved_x;
            vc->y = vc->saved_y;

```


drivers/char/console.c

Page 14/18

```

786:                                     CSE;
787:                                     continue;
788:     case 'D': /* Scroll Up <ESC>D */
789:         lf(vc);
790:         CSE;
791:         continue;
792:     case 'E': /* Move To Next Line <ES
C>E */
793:         cr(vc);
794:         lf(vc);
795:         CSE;
796:         continue;
797:     case 'H': /* Set Tab <ESC>H */
798:         vc->tty->tab_stop[vc->x] = 1;
799:         CSE;
800:         continue;
801:     case 'M': /* Scroll Down <ESC>M */
802:         ri(vc);
803:         CSE;
804:         continue;
805:     case 'Z': /* Identify Terminal <ES
C>Z */
806:         insert_seq(tty, VT100ID, 7);
807:         CSE;
808:         continue;
809:     case 'c': /* Reset Device <ESC>c *
/
810:         vconsole_reset(vc->tty);
811:         vc->x = vc->y = 0;
812:         csi_J(vc, CSI_J_SCREEN);
813:         CSE;
814:         continue;
815:     default:
816:         CSE;
817:         break;
818:     }
819: }
820: }
821: switch(ch) {
822:     case '\033':
823:         vc->esc = 1;
824:         vc->sbracket = 0;
825:         vc->semicolon = 0;
826:         vc->question = 0;
827:         vc->parmv1 = vc->parmv2 = 0;
828:         continue;
829:     default:
830:         echo_char(vc, &ch, 1);
831:         continue;
832: }
833: }
834: if(ch) {
835:     if(vc->vc_mode != KD_GRAPHICS) {
836:         video.update_curpos(vc);
837:     }
838:     wakeup(&do_select);
839:     wakeup(&tty_write);
840: }
841: }
842:
843: void vconsole_select(int new_cons)
844: {
845:     new_cons++;
846:     if(current_cons != new_cons) {
847:         if(vc[current_cons].vt_mode.mode == VT_PROCESS) {
848:             if(!kill_pid(vc[current_cons].tty->pid, vc[current_cons]
.vt_mode.acqsig)) {

```

drivers/char/console.c

Page 15/18

```

849:             vc[current_cons].switchto_tty = new_cons;
850:             return;
851:         }
852:         init_vt (&vc[current_cons]);
853:     }
854:     if(vc[current_cons].vc_mode == KD_GRAPHICS) {
855:         return;
856:     }
857:     vconsole_select_final(new_cons);
858: }
859: }
860:
861: void vconsole_select_final(int new_cons)
862: {
863:     if(current_cons != new_cons) {
864:         if(vc[new_cons].vt_mode.mode == VT_PROCESS) {
865:             if(kill_pid(vc[new_cons].tty->pid, vc[new_cons].vt_mode.
acqsig)) {
866:                 init_vt (&vc[new_cons]);
867:             }
868:         }
869:         if(video.buf_top) {
870:             video.buf_top = 0;
871:             video.show_cursor(&vc[current_cons], ON);
872:             video.update_curpos (&vc[current_cons]);
873:         }
874:         vc[current_cons].vidmem = NULL;
875:         vc[current_cons].flags &= ~CONSOLE_HAS_FOCUS;
876:         vc[new_cons].vidmem = (unsigned char *)video.address;
877:         vc[new_cons].flags |= CONSOLE_HAS_FOCUS;
878:         current_cons = new_cons;
879:         video.update_curpos (&vc[current_cons]);
880:         video.restore_screen (&vc[current_cons]);
881:         set_leds (vc[current_cons].led_status);
882:
883:         video.buf_y = vc[current_cons].y;
884:         video.buf_top = 0;
885:         vcbuf_refresh (&vc[current_cons]);
886:         video.show_cursor (&vc[current_cons], COND);
887:         video.cursor_blink ((unsigned int)&vc[current_cons]);
888:     }
889: }
890:
891: void unblank_screen(struct vconsole *vc)
892: {
893:     if(!(vc->flags & CONSOLE_BLANKED)) {
894:         return;
895:     }
896:     video.restore_screen(vc);
897:     vc->flags &= ~CONSOLE_BLANKED;
898:     video.show_cursor(vc, ON);
899: }
900:
901: void vconsole_start(struct tty *tty)
902: {
903:     struct vconsole *vc;
904:
905:     vc = (struct vconsole *)tty->driver_data;
906:     if(!vc->scrlock) {
907:         return;
908:     }
909:     vc->led_status &= ~SCR_LBIT;
910:     vc->scrlock = 0;
911:     set_leds (vc->led_status);
912: }
913:
914: void vconsole_stop(struct tty *tty)

```

drivers/char/console.c

Page 16/18

```

915: {
916:     struct vconsole *vc;
917:
918:     vc = (struct vconsole *)tty->driver_data;
919:     if(vc->scrlock) {
920:         return;
921:     }
922:     vc->led_status |= SCRLBIT;
923:     vc->scrlock = 1;
924:     set_leds(vc->led_status);
925: }
926:
927: void vconsole_beeper(void)
928: {
929:     struct callout_req creq;
930:
931:     pit_beeper_on();
932:     creq.fn = pit_beeper_off;
933:     creq.arg = 0;
934:     add_callout(&creq, HZ / 8);
935: }
936:
937: void vconsole_deltab(struct tty *tty)
938: {
939:     int col, n;
940:     unsigned char count;
941:     struct vconsole *vc;
942:     struct cblock *cb;
943:     unsigned char ch;
944:
945:     vc = (struct vconsole *)tty->driver_data;
946:     cb = tty->cooked_q.head;
947:     col = count = 0;
948:
949:     while(cb) {
950:         for(n = 0; n < cb->end_off; n++) {
951:             if(n >= cb->start_off) {
952:                 ch = cb->data[n];
953:                 if(ch == '\t') {
954:                     while(!vc->tty->tab_stop[++col]);
955:                 } else {
956:                     col++;
957:                     if(ISCNTRL(ch) && !ISSPACE(ch) && tty->t
ermios.c_lflag & ECHOCTL) {
958:                         col++;
959:                     }
960:                 }
961:                 col %= vc->columns;
962:             }
963:         }
964:         cb = cb->next;
965:     }
966:     count = vc->x - col;
967:
968:     while(count--) {
969:         tty_queue_putchar(tty, &tty->write_q, '\b');
970:     }
971: }
972:
973: void console_flush_log_buf(char *buffer, unsigned int count)
974: {
975:     char *b;
976:     struct tty *tty;
977:
978:     if(!(tty = get_tty(_syscondev))) {
979:         _syscondev = MKDEV(VCONSOLES_MAJOR, 0);
980:         tty = get_tty(_syscondev);

```

drivers/char/console.c

Page 17/18

```

981:         }
982:         b = buffer;
983:
984:         while(count) {
985:             if(tty_queue_putchar(tty, &tty->write_q, *b) < 0) {
986:                 tty->output(tty);
987:                 continue;
988:             }
989:             count--;
990:             b++;
991:         }
992:         tty->output(tty);
993:     }
994:
995: void console_init(void)
996: {
997:     int n;
998:     struct tty *tty;
999:
1000:    if(video.flags & VPF_VGA) {
1001:        printk("console 0x%04X-0x%04X - %s (%d virtual consoles)
\n", video.port, video.port + 1, video.signature, NR_VCONSOLES);
1002:    }
1003:    if(video.flags & VPF_VESAFB) {
1004:        printk("console - color frame buffer, scre
en=%dx%d, font=%dx%d\n", video.columns, video.lines, video.fb_char_width, video.fb_char
_height);
1005:        printk("\t\t\t\t(%d virtual consoles)\n", NR_VCONSOLES);
1006:    }
1007:
1008:    for(n = 1; n <= NR_VCONSOLES; n++) {
1009:        if(!register_tty(MKDEV(VCONSOLES_MAJOR, n))) {
1010:            tty = get_tty(MKDEV(VCONSOLES_MAJOR, n));
1011:            tty->driver_data = (void *)&vc[n];
1012:            tty->stop = vconsole_stop;
1013:            tty->start = vconsole_start;
1014:            tty->deltab = vconsole_deltab;
1015:            tty->reset = vconsole_reset;
1016:            tty->input = do_cook;
1017:            tty->output = vconsole_write;
1018:            vc[n].tty = tty;
1019:            if(video.flags & VPF_VGA) {
1020:                vc[n].screen = (short int *)kmallocl();
1021:            }
1022:            if(video.flags & VPF_VESAFB) {
1023:                vc[n].screen = vc_screen[n];
1024:            }
1025:            vc[n].vidmem = NULL;
1026:            memset_w(vc[n].screen, BLANK_MEM, SCREEN_SIZE);
1027:            vconsole_reset(tty);
1028:        }
1029:    }
1030:
1031:    current_cons = 1;
1032:    video.show_cursor(&vc[current_cons], ON);
1033:    vc[current_cons].vidmem = (unsigned char *)video.address;
1034:    vc[current_cons].flags |= CONSOLE_HAS_FOCUS;
1035:
1036:    if(video.flags & VPF_VGA) {
1037:        memcpy_w(vc[current_cons].screen, video.address, SCREEN_SIZE);
1038:    }
1039:
1040:    video.get_curpos(&vc[current_cons]);
1041:    video.update_curpos(&vc[current_cons]);
1042:    video.buf_y = vc[current_cons].y;
1043:    video.buf_top = 0;
1044:

```

drivers/char/console.c

Page 18/18

```
1045:     SET_MINOR(console_device.minors, 0);
1046:     SET_MINOR(console_device.minors, 1);
1047:     for(n = 0; n <= NR_VCONSOLES; n++) {
1048:         SET_MINOR(tty_device.minors, n);
1049:     }
1050:
1051:     register_device(CHR_DEV, &console_device);
1052:     register_device(CHR_DEV, &tty_device);
1053:
1054:     if(is_vconsole(_syscondev)) {
1055:         register_console(console_flush_log_buf);
1056:     }
1057: }
```

drivers/char/defkeymap.c

```

1: /*
2:  * fiwix/drivers/char/defkeymap.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/keyboard.h>
9: #include <fiwix/string.h>
10:
11: #define BS      127      /* backspace */
12:
13: __key_t keymap[NR_MODIFIERS * NR_SCODES] = {
14: /*
15:  * Standard US keyboard (default keymap) with 16 modifiers
16:  *
17:  *
18:  *
19:  *
20:  * ===== */
21: /* 00 - NULL */
22: /* 01 - ESC */
23: /* 02 - 1 */
24: /* 03 - 2 */
25: /* 04 - 3 */
26: /* 05 - 4 */
27: /* 06 - 5 */
28: /* 07 - 6 */
29: /* 08 - 7 */
30: /* 09 - 8 */
31: /* 10 - 9 */
32: /* 11 - 0 */
33: /* 12 - _ */
34: /* 13 - += */
35: /* 14 - BS */
36: /* 15 - TAB */
37: /* 16 - q */
38: /* 17 - w */
39: /* 18 - e */
40: /* 19 - r */
41: /* 20 - t */

```

drivers/char/defkeymap.c			Page 2/5		
42:	/* 21 - y */	L('y'), L('Y'), L('y'), L('Y'), C('Y'), 0, 0,	A('y'), 0, 0,	0, 0, 0,	0, 0, 0,
43:	/* 22 - u */	L('u'), L('U'), L('u'), L('U'), C('U'), 0, 0,	A('u'), 0, 0,	0, 0, 0,	0, 0, 0,
44:	/* 23 - i */	L('i'), L('I'), L('i'), L('I'), C('I'), 0, 0,	A('i'), 0, 0,	0, 0, 0,	0, 0, 0,
45:	/* 24 - o */	L('o'), L('O'), L('o'), L('O'), C('O'), 0, 0,	A('o'), 0, 0,	0, 0, 0,	0, 0, 0,
46:	/* 25 - p */	L('p'), L('P'), L('p'), L('P'), C('P'), 0, 0,	A('p'), 0, 0,	0, 0, 0,	0, 0, 0,
47:	/* 26 - [{	'[', '{', NULL, NULL, C('['), 0, 0,	A('['), 0, 0,	0, 0, 0,	0, 0, 0,
48:	/* 27 -] }	']', '}', '~', NULL, C(']'), 0, 0,	A(']'), 0, 0,	0, 0, 0,	0, 0, 0,
49:	/* 28 - CR	CR, CR, CR, CR, CR, 0, 0,	A(CR), 0, 0,	0, 0, 0,	0, 0, 0,
50:	/* 29 - LCTRL	LCTRL, LCTRL, LCTRL, LCTRL, LCTRL, 0, 0,	LCTRL, 0, 0,	0, 0, 0,	0, 0, 0,
51:	/* 30 - a	L('a'), L('A'), L('a'), L('A'), C('A'), 0, 0,	A('a'), 0, 0,	0, 0, 0,	0, 0, 0,
52:	/* 31 - s	L('s'), L('S'), L('s'), L('S'), C('S'), 0, 0,	A('s'), 0, 0,	0, 0, 0,	0, 0, 0,
53:	/* 32 - d	L('d'), L('D'), L('d'), L('D'), C('D'), 0, 0,	A('d'), 0, 0,	0, 0, 0,	0, 0, 0,
54:	/* 33 - f	L('f'), L('F'), L('f'), L('F'), C('F'), 0, 0,	A('f'), 0, 0,	0, 0, 0,	0, 0, 0,
55:	/* 34 - g	L('g'), L('G'), L('g'), L('G'), C('G'), 0, 0,	A('g'), 0, 0,	0, 0, 0,	0, 0, 0,
56:	/* 35 - h	L('h'), L('H'), L('h'), L('H'), C('H'), 0, 0,	A('h'), 0, 0,	0, 0, 0,	0, 0, 0,
57:	/* 36 - j	L('j'), L('J'), L('j'), L('J'), C('J'), 0, 0,	A('j'), 0, 0,	0, 0, 0,	0, 0, 0,
58:	/* 37 - k	L('k'), L('K'), L('k'), L('K'), C('K'), 0, 0,	A('k'), 0, 0,	0, 0, 0,	0, 0, 0,
59:	/* 38 - l	L('l'), L('L'), L('l'), L('L'), C('L'), 0, 0,	A('l'), 0, 0,	0, 0, 0,	0, 0, 0,
60:	/* 39 - ; :	';', ':', NULL, NULL, NULL, 0, 0,	A(';'), 0, 0,	0, 0, 0,	0, 0, 0,
61:	/* 40 - \"	'\\', '\"', NULL, NULL, C('G'), 0, 0,	A('\\'), 0, 0,	0, 0, 0,	0, 0, 0,
62:	/* 41 - ` ~	'`', '~', NULL, NULL, NULL, 0, 0,	A('`'), 0, 0,	0, 0, 0,	0, 0, 0,
63:	/* 42 - LSHF	LSHIFT, LSHIFT, LSHIFT, LSHIFT, LSHIFT, 0, 0,	LSHIFT, 0, 0,	0, 0, 0,	0, 0, 0,
64:	/* 43 - \	'\\', ' ', NULL, NULL, C('\\'), 0, 0,	A('\\'), 0, 0,	0, 0, 0,	0, 0, 0,
65:	/* 44 - z	L('z'), L('Z'), L('z'), L('Z'), C('Z'), 0, 0,	A('z'), 0, 0,	0, 0, 0,	0, 0, 0,
66:	/* 45 - x	L('x'), L('X'), L('x'), L('X'), C('X'), 0, 0,	A('x'), 0, 0,	0, 0, 0,	0, 0, 0,
67:	/* 46 - c	L('c'), L('C'), L('c'), L('C'), C('C'), 0, 0,	A('c'), 0, 0,	0, 0, 0,	0, 0, 0,
68:	/* 47 - v	L('v'), L('V'), L('v'), L('V'), C('V'), 0, 0,	A('v'), 0, 0,	0, 0, 0,	0, 0, 0,
69:	/* 48 - b	L('b'), L('B'), L('b'), L('B'), C('B'), 0, 0,	A('b'), 0, 0,	0, 0, 0,	0, 0, 0,
70:	/* 49 - n	L('n'), L('N'), L('n'), L('N'), C('N'), 0, 0,	A('n'), 0, 0,	0, 0, 0,	0, 0, 0,
71:	/* 50 - m	L('m'), L('M'), L('m'), L('M'), C('M'), 0, 0,	A('m'), 0, 0,	0, 0, 0,	0, 0, 0,
72:	/* 51 - , <	','', '<', NULL, NULL, NULL, 0, 0,	A(','), 0, 0,	0, 0, 0,	0, 0, 0,
73:	/* 52 - . >	'.'', '>', NULL, NULL, NULL, 0, 0,	A('.'), 0, 0,	0, 0, 0,	0, 0, 0,
74:	/* 53 - / ?	SLASH, '?', NULL, NULL, BS, 0, 0,	A('/'), 0, 0,	0, 0, 0,	0, 0, 0,
75:	/* 54 - RSHF	RSHIFT, RSHIFT, RSHIFT, RSHIFT, RSHIFT, 0, 0,	RSHIFT, 0, 0,	0, 0, 0,	0, 0, 0,

drivers/char/defkeymap.c											Page 3/5
0,	RSHIFT,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
76:	/* 55 - *	*/	ASTSK,	ASTSK,	ASTSK,	ASTSK,	ASTSK,	ASTSK,	0,	0,	0,
0,	ASTSK,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
77:	/* 56 - ALT	*/	ALT,	ALT,	ALT,	ALT,	ALT,	ALT,	0,	0,	0,
0,	ALT,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
78:	/* 57 - SPC	*/	' ',	' ',	NULL,	NULL,	NULL,	NULL,	0,	0,	0,
0,	A(' '),	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
79:	/* 58 - CAPS	*/	CAPS,	CAPS,	CAPS,	CAPS,	CAPS,	CAPS,	0,	0,	0,
0,	CAPS,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
80:	/* 59 - F1	*/	F1,	SF1,	NULL,	NULL,	F1,	0,	0,	0,	0,
0,	AF1,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
81:	/* 60 - F2	*/	F2,	SF2,	NULL,	NULL,	F2,	0,	0,	0,	0,
0,	AF2,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
82:	/* 61 - F3	*/	F3,	SF3,	NULL,	NULL,	F3,	0,	0,	0,	0,
0,	AF3,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
83:	/* 62 - F4	*/	F4,	SF4,	NULL,	NULL,	F4,	0,	0,	0,	0,
0,	AF4,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
84:	/* 63 - F5	*/	F5,	SF5,	NULL,	NULL,	F5,	0,	0,	0,	0,
0,	AF5,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
85:	/* 64 - F6	*/	F6,	SF6,	NULL,	NULL,	F6,	0,	0,	0,	0,
0,	AF6,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
86:	/* 65 - F7	*/	F7,	SF7,	NULL,	NULL,	F7,	0,	0,	0,	0,
0,	AF7,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
87:	/* 66 - F8	*/	F8,	SF8,	NULL,	NULL,	F8,	0,	0,	0,	0,
0,	AF8,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
88:	/* 67 - F9	*/	F9,	SF9,	NULL,	NULL,	F9,	0,	0,	0,	0,
0,	AF9,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
89:	/* 68 - F10	*/	F10,	SF10,	NULL,	NULL,	F10,	0,	0,	0,	0,
0,	AF10,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
90:	/* 69 - NUMS	*/	NUMS,	NUMS,	NUMS,	NUMS,	NUMS,	0,	0,	0,	0,
0,	NUMS,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
91:	/* 70 - SCRL	*/	SCRL,	SCRL3,	SCRL2,	NULL,	SCRL4,	0,	0,	0,	0,
0,	SCRL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
92:	/* 71 - HOME/7	*/	HOME,	HOME,	HOME,	HOME,	HOME,	0,	0,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
93:	/* 72 - UP /8	*/	UP,	UP,	UP,	UP,	UP,	0,	0,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
94:	/* 73 - PGUP/9	*/	PGUP,	PGUP,	PGUP,	PGUP,	PGUP,	0,	0,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
95:	/* 74 - MINUS	*/	MINUS,	MINUS,	MINUS,	MINUS,	MINUS,	0,	0,	0,	0,
0,	MINUS,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
96:	/* 75 - LEFT/4	*/	LEFT,	LEFT,	LEFT,	LEFT,	LEFT,	0,	0,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
97:	/* 76 - MID /5	*/	MID,	MID,	MID,	MID,	MID,	0,	0,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
98:	/* 77 - RIGH/6	*/	RIGHT,	RIGHT,	RIGHT,	RIGHT,	RIGHT,	0,	0,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
99:	/* 78 - PLUS	*/	PLUS,	PLUS,	PLUS,	PLUS,	PLUS,	0,	0,	0,	0,
0,	PLUS,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
100:	/* 79 - END /1	*/	END,	END,	END,	END,	END,	0,	0,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
101:	/* 80 - DOWN/2	*/	DOWN,	DOWN,	DOWN,	DOWN,	DOWN,	0,	0,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
102:	/* 81 - PGDN/3	*/	PGDN,	PGDN,	PGDN,	PGDN,	PGDN,	0,	0,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
103:	/* 82 - INS /0	*/	INS,	INS,	INS,	INS,	INS,	0,	0,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
104:	/* 83 - DEL /.	*/	DEL,	DEL,	DEL,	DEL,	DEL,	0,	0,	0,	0,
0,	DEL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
105:	/* 84 - SYSRQ	*/	SYSRQ,	NULL,	NULL,	NULL,	NULL,	0,	0,	0,	0,
0,	SYSRQ,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
106:	/* 85 -	*/	NULL,	NULL,	NULL,	NULL,	NULL,	0,	0,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
107:	/* 86 - <>	*/	'<',	'>',	' ',	NULL,	NULL,	0,	0,	0,	0,
0,	A('<'),	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
108:	/* 87 - F11	*/	SF1,	SF1,	NULL,	NULL,	F11,	0,	0,	0,	0,
0,	AF11,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,

drivers/char/defkeymap.c										Page 4/5		
109:	/*	88	-	F12	*/	SF2,	SF2,	NULL,	NULL,	F12,	0,	0,
0,	AF12,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
110:	/*	89	-		*/	NULL,	NULL,	NULL,	NULL,	NULL,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
111:	/*	90	-		*/	NULL,	NULL,	NULL,	NULL,	NULL,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
112:	/*	91	-		*/	NULL,	NULL,	NULL,	NULL,	NULL,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
113:	/*	92	-		*/	NULL,	NULL,	NULL,	NULL,	NULL,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
114:	/*	93	-		*/	NULL,	NULL,	NULL,	NULL,	NULL,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
115:	/*	94	-		*/	NULL,	NULL,	NULL,	NULL,	NULL,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
116:	/*	95	-		*/	NULL,	NULL,	NULL,	NULL,	NULL,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
117:	/*	96	-	EOENTER	*/	EOENTER,	EOENTER,	EOENTER,	EOENTER,	EOENTER,	0,	0,
0,	EOENTER,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
118:	/*	97	-	RCTRL	*/	RCTRL,	RCTRL,	RCTRL,	RCTRL,	RCTRL,	0,	0,
0,	RCTRL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
119:	/*	98	-	EOSLASH	*/	EOSLASH,	EOSLASH,	EOSLASH,	EOSLASH,	EOSLASH,	0,	0,
0,	EOSLASH,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
120:	/*	99	-		*/	NULL,	NULL,	NULL,	NULL,	C('\\"),	0,	0,
0,	C('\\"),	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
121:	/*	100	-	ALTGR	*/	ALTGR,	ALTGR,	ALTGR,	ALTGR,	ALTGR,	0,	0,
0,	ALTGR,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
122:	/*	101	-		*/	NULL,	NULL,	NULL,	NULL,	NULL,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
123:	/*	102	-	EOHOME	*/	EOHOME,	EOHOME,	EOHOME,	EOHOME,	EOHOME,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
124:	/*	103	-	EOUP	*/	EOUP,	EOUP,	EOUP,	EOUP,	EOUP,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
125:	/*	104	-	EOPGUP	*/	EOPGUP,	EOPGUP,	EOPGUP,	EOPGUP,	EOPGUP,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
126:	/*	105	-	EOLEFT	*/	EOLEFT,	EOLEFT,	EOLEFT,	EOLEFT,	EOLEFT,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
127:	/*	106	-	EORIGHT	*/	EORIGHT,	EORIGHT,	EORIGHT,	EORIGHT,	EORIGHT,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
128:	/*	107	-	EOEND	*/	EOEND,	EOEND,	EOEND,	EOEND,	EOEND,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
129:	/*	108	-	EODOWN	*/	EODOWN,	EODOWN,	EODOWN,	EODOWN,	EODOWN,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
130:	/*	109	-	EOPGDN	*/	EOPGDN,	EOPGDN,	EOPGDN,	EOPGDN,	EOPGDN,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
131:	/*	110	-	EOINS	*/	EOINS,	EOINS,	EOINS,	EOINS,	EOINS,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
132:	/*	111	-	EODEL	*/	EODEL,	EODEL,	EODEL,	EODEL,	EODEL,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
133:	/*	112	-		*/	NULL,	NULL,	NULL,	NULL,	NULL,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
134:	/*	113	-		*/	NULL,	NULL,	NULL,	NULL,	NULL,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
135:	/*	114	-		*/	NULL,	NULL,	NULL,	NULL,	NULL,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
136:	/*	115	-		*/	NULL,	NULL,	NULL,	NULL,	NULL,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
137:	/*	116	-		*/	NULL,	NULL,	NULL,	NULL,	NULL,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
138:	/*	117	-		*/	NULL,	NULL,	NULL,	NULL,	NULL,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
139:	/*	118	-		*/	NULL,	NULL,	NULL,	NULL,	NULL,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
140:	/*	119	-		*/	NULL,	NULL,	NULL,	NULL,	NULL,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
141:	/*	120	-		*/	NULL,	NULL,	NULL,	NULL,	NULL,	0,	0,
0,	NULL,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,	0,
142:	/*	121	-		*/	NULL,	NULL,	NULL,	NULL,	NULL,	0,	0,

drivers/char/defkeymap.c

Page 5/5

```
0,      NULL,  0,    0,    0,    0,    0,    0,    0,    0,    0,
143: /* 122 -    */
0,      NULL,  0,    0,    0,    0,    0,    0,    0,    0,    0,
144: /* 123 -    */
0,      NULL,  0,    0,    0,    0,    0,    0,    0,    0,    0,
145: /* 124 -    */
0,      NULL,  0,    0,    0,    0,    0,    0,    0,    0,    0,
146: /* 125 -    */
0,      NULL,  0,    0,    0,    0,    0,    0,    0,    0,    0,
147: /* 126 -    */
0,      NULL,  0,    0,    0,    0,    0,    0,    0,    0,    0,
148: /* 127 -    */
0,      NULL,  0,    0,    0,    0,    0,    0,    0,    0,    0,
149: };
```

drivers/char/fb.c

Page 1/3

```

1: /*
2:  * fiwix/drivers/char/fb.c
3:  *
4:  * Copyright 2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/kernel.h>
10: #include <fiwix/fb.h>
11: #include <fiwix/devices.h>
12: #include <fiwix/fs.h>
13: #include <fiwix/errno.h>
14: #include <fiwix/stdio.h>
15: #include <fiwix/string.h>
16: #include <fiwix/fb.h>
17: #include <fiwix/mm.h>
18:
19: static struct fs_operations fb_driver_fsop = {
20:     0,
21:     0,
22:
23:     fb_open,
24:     fb_close,
25:     fb_read,
26:     fb_write,
27:     fb_ioctl,
28:     fb_lseek,
29:     NULL,                /* readdir */
30:     NULL,
31:     NULL,                /* select */
32:
33:     NULL,                /* readlink */
34:     NULL,                /* followlink */
35:     NULL,                /* bmap */
36:     NULL,                /* lockup */
37:     NULL,                /* rmdir */
38:     NULL,                /* link */
39:     NULL,                /* unlink */
40:     NULL,                /* symlink */
41:     NULL,                /* mkdir */
42:     NULL,                /* mknod */
43:     NULL,                /* truncate */
44:     NULL,                /* create */
45:     NULL,                /* rename */
46:
47:     NULL,                /* read_block */
48:     NULL,                /* write_block */
49:
50:     NULL,                /* read_inode */
51:     NULL,                /* write_inode */
52:     NULL,                /* ialloc */
53:     NULL,                /* ifree */
54:     NULL,                /* statfs */
55:     NULL,                /* read_superblock */
56:     NULL,                /* remount_fs */
57:     NULL,                /* write_superblock */
58:     NULL,                /* release_superblock */
59: };
60:
61: static struct device fb_device = {
62:     "fb",
63:     FB_MAJOR,
64:     { 0, 0, 0, 0, 0, 0, 0, 0 },
65:     0,
66:     NULL,
67:     &fb_driver_fsop,

```

drivers/char/fb.c

Page 2/3

```

68:         NULL
69:     };
70:
71:     struct video_parms video;
72:
73:     int fb_open(struct inode *i, struct fd *fd_table)
74:     {
75:         return 0;
76:     }
77:
78:     int fb_close(struct inode *i, struct fd *fd_table)
79:     {
80:         return 0;
81:     }
82:
83:     int fb_read(struct inode *i, struct fd *fd_table, char *buffer, __size_t count)
84:     {
85:         unsigned int addr;
86:
87:         if(fd_table->offset >= video.memsize) {
88:             return 0;
89:         }
90:
91:         addr = (unsigned int)video.address + fd_table->offset;
92:         count = MIN(count, video.memsize - fd_table->offset);
93:         memcpy_b(buffer, (void *)addr, count);
94:         fd_table->offset += count;
95:         return count;
96:     }
97:
98:     int fb_write(struct inode *i, struct fd *fd_table, const char *buffer, __size_t
count)
99:     {
100:         unsigned int addr;
101:
102:         if(fd_table->offset >= video.memsize) {
103:             return -ENOSPC;
104:         }
105:
106:         addr = (unsigned int)video.address + fd_table->offset;
107:         count = MIN(count, video.memsize - fd_table->offset);
108:         memcpy_b((void *)addr, buffer, count);
109:         fd_table->offset += count;
110:         return count;
111:     }
112:
113:     int fb_ioctl(struct inode *i, int cmd, unsigned long int arg)
114:     {
115:         return -EINVAL;
116:     }
117:
118:     int fb_lseek(struct inode *i, __off_t offset)
119:     {
120:         return offset;
121:     }
122:
123:     void fb_init(void)
124:     {
125:         unsigned int limit;
126:
127:         SET_MINOR(fb_device.minors, 0);
128:         limit = (unsigned int)video.address + video.memsize;
129:
130:         printk("fb0          0x%08X-0x%08X type=%s %X.%X resolution=%dx%dx%d size=%
dMB\n",
131:             video.address,
132:             limit,

```

drivers/char/fb.c

Page 3/3

```
133:         video.signature,
134:         video.fb_version >> 8,
135:         video.fb_version & 0xFF,
136:         video.fb_width,
137:         video.fb_height,
138:         video.fb_bpp,
139:         video.memsize / 1024 / 1024
140:     );
141:     if(register_device(CHR_DEV, &fb_device)) {
142:         printk("ERROR: %s(): unable to register fb device.\n", __FUNCTION
N__);
143:         return;
144:     }
145: }
```

drivers/char/keyboard.c

Page 1/12

```

1: /*
2:  * fiwix/drivers/char/keyboard.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/kernel.h>
10: #include <fiwix/limits.h>
11: #include <fiwix/keyboard.h>
12: #include <fiwix/console.h>
13: #include <fiwix/vgacon.h>
14: #include <fiwix/pic.h>
15: #include <fiwix/signal.h>
16: #include <fiwix/process.h>
17: #include <fiwix/sleep.h>
18: #include <fiwix/kd.h>
19: #include <fiwix/sysrq.h>
20: #include <fiwix/stdio.h>
21: #include <fiwix/string.h>
22:
23: #define KB_DATA          0x60      /* I/O data port */
24: #define KBC_COMMAND     0x64      /* command/control port */
25: #define KBC_STATUS      0x64      /* status register port */
26:
27: /*
28:  * PS/2 System Control Port A
29:  * -----
30:  * bit 7 -> fixed disk activity led
31:  * bit 6 -> fixed disk activity led
32:  * bit 5 -> reserved
33:  * bit 4 -> watchdog timer status
34:  * bit 3 -> security lock latch
35:  * bit 2 -> reserved
36:  * bit 1 -> alternate gate A20
37:  * bit 0 -> alternate hot reset
38:  */
39: #define PS2_SYSCtrl_A   0x92      /* PS/2 system control port A (write) */
40:
41: #define KB_CMD_RESET    0xFF      /* keyboard reset */
42: #define KB_CMD_ENABLE   0xF4      /* keyboard enable scanning */
43: #define KB_CMD_DISABLE  0xF5      /* keyboard disable scanning */
44: #define KB_CMD_IDENTIFY 0xF2      /* keyboard identify (for PS/2 only) */
45: #define KB_CMD_ECHO     0xEE      /* echo (for diagnostics only) */
46:
47: #define KBC_CMD_RECV_CONFIG 0x20   /* read controller's config byte */
48: #define KBC_CMD_SEND_CONFIG 0x60   /* write controller's config byte */
49: #define KBC_CMD_SELF_TEST  0xAA    /* self-test command */
50: #define KBC_CMD_PS2_1_TEST 0xAB    /* first PS/2 interface test command */
51: #define KBC_CMD_PS2_2_TEST 0xA9    /* second PS/2 interface test command */
52: #define KBC_CMD_DISABLE_PS2_1 0xAD  /* disable first PS/2 port */
53: #define KBC_CMD_ENABLE_PS2_1 0xAE   /* enable first PS/2 port */
54: #define KBC_CMD_DISABLE_PS2_2 0xA7  /* disable second PS/2 port (if any) */
55: #define KBC_CMD_ENABLE_PS2_2 0xA8   /* enable second PS/2 port (if any) */
56: #define KBC_CMD_GET_IFACE  0xCA    /* get interface type (AT or MCA) */
57: #define KBC_CMD_HOTRESET   0xFE    /* Hot Reset */
58:
59: /* flags of the status register */
60: #define KB_STR_OUTBUSY   0x01      /* output buffer full, don't read yet */
61: #define KB_STR_INBUSY    0x02      /* input buffer full, don't write yet */
62: #define KB_STR_TXTMOUT   0x20      /* transmit time-out error */
63: #define KB_STR_RXTMOUT   0x40      /* receive time-out error */
64: #define KB_STR_PARERR    0x80      /* parity error */
65: #define KB_STR_COMMERR   (KB_STR_TXTMOUT | KB_STR_RXTMOUT)
66:
67: #define KB_RESET_OK     0xAA      /* self-test passed */

```

drivers/char/keyboard.c

Page 2/12

```

68: #define KB_ACK          0xFA    /* acknowledge */
69: #define KB_SETLED       0xED    /* set/reset status indicators (LEDs) */
70: #define KB_RATE         0xF3    /* set typematic rate/delay */
71: #define DELAY_250       0x00    /* typematic delay at 250ms (default) */
72: #define DELAY_500       0x40    /* typematic delay at 500ms */
73: #define DELAY_750       0x80    /* typematic delay at 750ms */
74: #define DELAY_1000      0xC0    /* typematic delay at 1000ms */
75: #define RATE_30         0x00    /* typematic rate at 30.0 reports/sec (default)
*/
76:
77: #define EXTKEY           0xE0    /* extended key (AltGr, Ctrl-Print, etc.) */
78:
79: __key_t *keymap_line;
80:
81: static unsigned char e0_keys[128] = {
82:     0, 0, 0, 0, 0, 0, 0, 0, /* 0x00-0x07 */
83:     0, 0, 0, 0, 0, 0, 0, 0, /* 0x08-0x0F */
84:     0, 0, 0, 0, 0, 0, 0, 0, /* 0x10-0x17 */
85:     0, 0, 0, 0, E0ENTER, RCTRL, 0, 0, /* 0x18-0x1F */
86:     0, 0, 0, 0, 0, 0, 0, 0, /* 0x20-0x27 */
87:     0, 0, 0, 0, 0, 0, 0, 0, /* 0x28-0x2F */
88:     0, 0, 0, 0, 0, E0SLASH, 0, 0, /* 0x30-0x37 */
89:     ALTGR, 0, 0, 0, 0, 0, 0, 0, /* 0x38-0x3F */
90:     0, 0, 0, 0, 0, 0, 0, E0HOME, /* 0x40-0x47 */
91:     E0UP, E0GUP, 0, E0LEFT, 0, E0RIGHT, 0, E0END, /* 0x48-0x4F */
92:     E0DOWN, E0PGDN, E0INS, E0DEL, 0, 0, 0, 0, /* 0x50-0x57 */
93:     0, 0, 0, 0, 0, 0, 0, 0, /* 0x58-0x5f */
94:     0, 0, 0, 0, 0, 0, 0, 0, /* 0x60-0x67 */
95:     0, 0, 0, 0, 0, 0, 0, 0, /* 0x68-0x6F */
96:     0, 0, 0, 0, 0, 0, 0, 0, /* 0x70-0x77 */
97:     0, 0, 0, 0, 0, 0, 0, 0, /* 0x78-0x7F */
98: };
99:
100: static unsigned char leds = 0;
101: static unsigned char shift = 0;
102: static unsigned char altgr = 0;
103: static unsigned char ctrl = 0;
104: static unsigned char alt = 0;
105: static unsigned char extkey = 0;
106: static unsigned char deadkey = 0;
107: static unsigned char sysrq = 0;
108: static int sysrq_op = 0;
109:
110: static unsigned char do_buf_scroll = 0;
111: static char do_switch_console = -1;
112: struct video_parms video;
113:
114: unsigned char kb_identify[2] = {0, 0};
115: char ps2_active_ports = 0;
116: char ps2_supp_ports = 0;
117: char ps2_iface = 0;
118: short int current_cons;
119: char ctrl_alt_del = 1;
120: char any_key_to_reboot = 0;
121:
122: static struct bh keyboard_bh = { 0, &irq_keyboard_bh, NULL };
123: static struct interrupt irq_config_keyboard = { 0, "keyboard", &irq_keyboard, NU
LL };
124:
125: struct diacritic *diacr;
126: static char *diacr_chars = "``^ \";
127: struct diacritic grave_table[NR_DIACR] = {
128:     { 'A', '\300' },
129:     { 'E', '\310' },
130:     { 'I', '\314' },
131:     { 'O', '\322' },
132:     { 'U', '\331' },

```

drivers/char/keyboard.c

Page 3/12

```

133:     { 'a', '\340' },
134:     { 'e', '\350' },
135:     { 'i', '\354' },
136:     { 'o', '\362' },
137:     { 'u', '\371' },
138: };
139: struct diacritic acute_table[NR_DIACR] = {
140:     { 'A', '\301' },
141:     { 'E', '\311' },
142:     { 'I', '\315' },
143:     { 'O', '\323' },
144:     { 'U', '\332' },
145:     { 'a', '\341' },
146:     { 'e', '\351' },
147:     { 'i', '\355' },
148:     { 'o', '\363' },
149:     { 'u', '\372' },
150: };
151: struct diacritic circm_table[NR_DIACR] = {
152:     { 'A', '\302' },
153:     { 'E', '\312' },
154:     { 'I', '\316' },
155:     { 'O', '\324' },
156:     { 'U', '\333' },
157:     { 'a', '\342' },
158:     { 'e', '\352' },
159:     { 'i', '\356' },
160:     { 'o', '\364' },
161:     { 'u', '\373' },
162: };
163: struct diacritic diere_table[NR_DIACR] = {
164:     { 'A', '\304' },
165:     { 'E', '\313' },
166:     { 'I', '\317' },
167:     { 'O', '\326' },
168:     { 'U', '\334' },
169:     { 'a', '\344' },
170:     { 'e', '\353' },
171:     { 'i', '\357' },
172:     { 'o', '\366' },
173:     { 'u', '\374' },
174: };
175:
176: static char *pad_chars = "0123456789+--*/\015,.";
177:
178: static char *pad_seq[] = {
179:     "\033[2~", /* INS */
180:     "\033[4~", /* END */
181:     "\033[B", /* DOWN */
182:     "\033[6~", /* PGDN */
183:     "\033[D", /* LEFT */
184:     "\033[G", /* MID */
185:     "\033[C", /* RIGHT */
186:     "\033[1~", /* HOME */
187:     "\033[A", /* UP */
188:     "\033[5~", /* PGUP */
189:     "+", /* PLUS */
190:     "-", /* MINUS */
191:     "*", /* ASTERISK */
192:     "/", /* SLASH */
193:     "'\n'", /* ENTER */
194:     ",", /* COMMA */
195:     "\033[3~", /* DEL */
196: };
197:
198: static char *fn_seq[] = {
199:     "\033[A", /* F1 */

```


drivers/char/keyboard.c

Page 4/12

```

200:         "\033[B",          /* F2 */
201:         "\033[C",          /* F3 */
202:         "\033[D",          /* F4 */
203:         "\033[E",          /* F5 */
204:         "\033[17~",        /* F6 */
205:         "\033[18~",        /* F7 */
206:         "\033[19~",        /* F8 */
207:         "\033[20~",        /* F9 */
208:         "\033[21~",        /* F10 */
209:         "\033[23~",        /* F11, SF1 */
210:         "\033[24~",        /* F12, SF2 */
211:         "\033[25~",        /* SF3 */
212:         "\033[26~",        /* SF4 */
213:         "\033[28~",        /* SF5 */
214:         "\033[29~",        /* SF6 */
215:         "\033[31~",        /* SF7 */
216:         "\033[32~",        /* SF8 */
217:         "\033[33~",        /* SF9 */
218:         "\033[34~",        /* SF10 */
219: };
220:
221: static void keyboard_delay(void)
222: {
223:     int n;
224:
225:     for(n = 0; n < 1000; n++) {
226:         NOP();
227:     }
228: }
229:
230: /* wait controller input buffer to be clear (ready to write) */
231: static int keyboard_wait_input(void)
232: {
233:     int n;
234:
235:     for(n = 0; n < 500000; n++) {
236:         if(!(inport_b(KBC_STATUS) & KB_STR_INBUSY)) {
237:             return 0;
238:         }
239:     }
240:     return 1;
241: }
242:
243: static int keyboard_write(const unsigned char port, const unsigned char byte)
244: {
245:     if(!keyboard_wait_input()) {
246:         outport_b(port, byte);
247:         if(!keyboard_wait_input()) {
248:             return 0;
249:         }
250:     }
251:
252:     return 1;
253: }
254:
255: /* wait controller output buffer to be full (ready to read) */
256: static int keyboard_wait_output(void)
257: {
258:     int n, value;
259:
260:     for(n = 0; n < 500000; n++) {
261:         if((value = inport_b(KBC_STATUS)) & KB_STR_OUTBUSY) {
262:             if(value & (KB_STR_COMMERR | KB_STR_PARERR)) {
263:                 continue;
264:             }
265:             return 0;
266:         }

```

drivers/char/keyboard.c

Page 5/12

```

267:         }
268:         return 1;
269:     }
270:
271:     static int keyboard_wait_ack(void)
272:     {
273:         int n;
274:
275:         keyboard_wait_output();
276:         for(n = 0; n < 1000; n++) {
277:             if(inport_b(KB_DATA) == KB_ACK) {
278:                 return 0;
279:             }
280:             keyboard_delay();
281:         }
282:         return 1;
283:     }
284:
285:     static void keyboard_identify(void)
286:     {
287:         /* disable */
288:         keyboard_write(KB_DATA, KB_CMD_DISABLE);
289:         if(keyboard_wait_ack()) {
290:             printk("WARNING: %s(): ACK not received on disable command!\n",
__FUNCTION__);
291:         }
292:
293:         /* identify */
294:         keyboard_write(KB_DATA, KB_CMD_IDENTIFY);
295:         if(keyboard_wait_ack()) {
296:             printk("WARNING: %s(): ACK not received on identify command!\n",
__FUNCTION__);
297:         }
298:         if(!keyboard_wait_output()) {
299:             kb_identify[0] = inport_b(KB_DATA);
300:         }
301:         if(!keyboard_wait_output()) {
302:             kb_identify[1] = inport_b(KB_DATA);
303:         }
304:
305:         /* enable */
306:         keyboard_write(KB_DATA, KB_CMD_ENABLE);
307:         if(keyboard_wait_ack()) {
308:             printk("WARNING: %s(): ACK not received on enable command!\n", __
__FUNCTION__);
309:         }
310:         keyboard_wait_output();
311:         inport_b(KB_DATA);
312:
313:         /* get the interface type */
314:         keyboard_write(KBC_COMMAND, KBC_CMD_GET_IFACE);
315:         keyboard_wait_output();
316:         ps2_iface = inport_b(KB_DATA);
317:     }
318:
319:     static void keyboard_reset(void)
320:     {
321:         int errno;
322:         unsigned char config;
323:
324:         /* disable device(s) */
325:         keyboard_write(KBC_COMMAND, KBC_CMD_DISABLE_PS2_1);
326:         keyboard_write(KBC_COMMAND, KBC_CMD_DISABLE_PS2_2);
327:
328:         /* flush buffers */
329:         while(!keyboard_wait_output()) {
330:             inport_b(KB_DATA);

```

drivers/char/keyboard.c

Page 6/12

```

331:         }
332:
333:         /* get controller configuration */
334:         keyboard_write(KBC_COMMAND, KBC_CMD_RECV_CONFIG);
335:         keyboard_wait_output();
336:         config = inport_b(KB_DATA);
337:         ps2_active_ports = config & 0x01 ? 1 : 0;
338:         ps2_active_ports += config & 0x02 ? 1 : 0;
339:         ps2_supp_ports = 1 + (config & 0x20 ? 1 : 0);
340:
341:         /* set controller configuration (disabling IRQs) */
342:         /*
343:         keyboard_write(KBC_COMMAND, KBC_CMD_SEND_CONFIG);
344:         keyboard_write(KB_DATA, config & ~(0x01 | 0x02 | 0x40));
345:         */
346:
347:         /* PS/2 controller self-test */
348:         keyboard_write(KBC_COMMAND, KBC_CMD_SELF_TEST);
349:         keyboard_wait_output();
350:         if((errno = inport_b(KB_DATA)) != 0x55) {
351:             printk("WARNING: %s(): keyboard returned 0x%x in self-test.\n",
__FUNCTION__, errno);
352:         }
353:
354:         /*
355:         * This sets again the controller configuration since the previous
356:         * step may also reset the PS/2 controller to its power-on defaults.
357:         */
358:         keyboard_write(KBC_COMMAND, KBC_CMD_SEND_CONFIG);
359:         keyboard_write(KB_DATA, config);
360:
361:         /* first PS/2 interface test */
362:         keyboard_write(KBC_COMMAND, KBC_CMD_PS2_1_TEST);
363:         keyboard_wait_output();
364:         if((errno = inport_b(KB_DATA)) != 0) {
365:             printk("WARNING: %s(): keyboard returned 0x%x in first PS/2 inte
rface test.\n", __FUNCTION__, errno);
366:         }
367:
368:         if(ps2_supp_ports > 1) {
369:             /* second PS/2 interface test */
370:             keyboard_write(KBC_COMMAND, KBC_CMD_PS2_2_TEST);
371:             keyboard_wait_output();
372:             if((errno = inport_b(KB_DATA)) != 0) {
373:                 printk("WARNING: %s(): keyboard returned 0x%x in second
PS/2 interface test.\n", __FUNCTION__, errno);
374:             }
375:         }
376:
377:         /* enable device(s) */
378:         keyboard_write(KBC_COMMAND, KBC_CMD_ENABLE_PS2_1);
379:         keyboard_write(KBC_COMMAND, KBC_CMD_ENABLE_PS2_2);
380:
381:         /* reset device(s) */
382:         keyboard_write(KB_DATA, KB_CMD_RESET);
383:         if(keyboard_wait_ack()) {
384:             printk("WARNING: %s(): ACK not received on reset command!\n", __
FUNCTION__);
385:         }
386:         if(!keyboard_wait_output()) {
387:             if((errno = inport_b(KB_DATA)) != KB_RESET_OK) {
388:                 printk("WARNING: %s(): keyboard returned 0x%x in reset.\n
n", __FUNCTION__, errno);
389:             }
390:         }
391:
392:         return;

```

drivers/char/keyboard.c

Page 7/12

```

393: }
394:
395: static void putc(struct tty *tty, unsigned char ch)
396: {
397:     if(tty_queue_putchar(tty, &tty->read_q, ch) < 0) {
398:         if(tty->termios.c_iflag & IMAXBEL) {
399:             vconsole_beep();
400:         }
401:     }
402: }
403:
404: static void puts(struct tty *tty, char *seq)
405: {
406:     char ch;
407:
408:     while((ch = *(seq++))) {
409:         putc(tty, ch);
410:     }
411: }
412:
413: void reboot(void)
414: {
415:     CLI();
416:     keyboard_write(PS2_SYSCTRL_A, 0x01); /* Fast Hot Reset */
417:     keyboard_write(KBC_COMMAND, KBC_CMD_HOTRESET); /* Hot Reset */
418:     HLT();
419: }
420:
421: void set_leds(unsigned char leds)
422: {
423:     keyboard_write(KB_DATA, KB_SETLED);
424:     keyboard_wait_ack();
425:
426:     keyboard_write(KB_DATA, leds);
427:     keyboard_wait_ack();
428: }
429:
430: void irq_keyboard(int num, struct sigcontext *sc)
431: {
432:     __key_t key, type;
433:     unsigned char scode, mod;
434:     struct tty *tty;
435:     struct vconsole *vc;
436:     unsigned char c;
437:     int n;
438:
439:     tty = get_tty(MKDEV(VCONSOLES_MAJOR, current_cons));
440:     vc = (struct vconsole *)tty->driver_data;
441:
442:     scode = inport_b(KB_DATA);
443:
444:     /* keyboard said all is OK, perfect */
445:     if(scode == KB_ACK) {
446:         return;
447:     }
448:
449:     video.screen_on(vc);
450:     keyboard_bh.flags |= BH_ACTIVE;
451:
452:     /* if in pure raw mode just queue the scan code and return */
453:     if(tty->kbd.mode == K_RAW) {
454:         putc(tty, scode);
455:         return;
456:     }
457:
458:     if(scode == EXTKEY) {
459:         extkey = 1;

```

drivers/char/keyboard.c

Page 8/12

```

460:         return;
461:     }
462:
463:     if(extkey) {
464:         key = e0_keys[scode & 0x7F];
465:     } else {
466:         key = scode & 0x7F;
467:     }
468:
469:     if(tty->kbd.mode == K_MEDIUMRAW) {
470:         putc(tty, key | (scode & 0x80));
471:         extkey = 0;
472:         return;
473:     }
474:
475:     key = keymap[NR_MODIFIERS * (scode & 0x7F)];
476:
477:     /* bit 7 enabled means a key has been released */
478:     if(scode & NR_SCODES) {
479:         switch(key) {
480:             case CTRL:
481:             case LCTRL:
482:             case RCTRL:
483:                 ctrl = 0;
484:                 break;
485:             case ALT:
486:                 if(!extkey) {
487:                     alt = 0;
488:                     sysrq = 0;
489:                 } else {
490:                     altgr = 0;
491:                 }
492:                 break;
493:             case SHIFT:
494:             case LSHIFT:
495:             case RSHIFT:
496:                 if(!extkey) {
497:                     shift = 0;
498:                 }
499:                 break;
500:             case CAPS:
501:             case NUMS:
502:             case SCRL:
503:                 leds = 0;
504:                 break;
505:         }
506:         extkey = 0;
507:         return;
508:     }
509:
510:     switch(key) {
511:         case CAPS:
512:             if(!leds) {
513:                 vc->led_status ^= CAPSBIT;
514:                 vc->capslock = !vc->capslock;
515:                 set_leds(vc->led_status);
516:             }
517:             leds = 1;
518:             return;
519:         case NUMS:
520:             if(!leds) {
521:                 vc->led_status ^= NUMSBIT;
522:                 vc->numlock = !vc->numlock;
523:                 set_leds(vc->led_status);
524:             }
525:             leds = 1;
526:             return;

```

drivers/char/keyboard.c

Page 9/12

```

527:         case SCRL:
528:             if(!leds) {
529:                 if(vc->scrlock) {
530:                     tty->start(tty);
531:                 } else {
532:                     tty->stop(tty);
533:                 }
534:             }
535:             leds = 1;
536:             return;
537:         case CTRL:
538:         case LCTRL:
539:         case RCTRL:
540:             ctrl = 1;
541:             return;
542:         case ALT:
543:             if(!extkey) {
544:                 alt = 1;
545:             } else {
546:                 altgr = 1;
547:             }
548:             return;
549:         case SHIFT:
550:         case LSHIFT:
551:         case RSHIFT:
552:             shift = 1;
553:             extkey = 0;
554:             return;
555:     }
556:
557:     if(ctrl && alt && key == DEL) {
558:         if(ctrl_alt_del) {
559:             reboot();
560:         } else {
561:             send_sig(&proc_table[INIT], SIGINT);
562:         }
563:         return;
564:     }
565:
566:     keymap_line = &keymap[(scode & 0x7F) * NR_MODIFIERS];
567:     mod = 0;
568:
569:     if(vc->capslock && (keymap_line[MOD_BASE] & LETTER_KEYS)) {
570:         mod = !vc->capslock ? shift : vc->capslock - shift;
571:     } else {
572:         if(shift && !extkey) {
573:             mod = 1;
574:         }
575:     }
576:     if(altgr) {
577:         mod = 2;
578:     }
579:     if(ctrl) {
580:         mod = 4;
581:     }
582:     if(alt) {
583:         mod = 8;
584:     }
585:
586:     key = keymap_line[mod];
587:
588:     if(key >= AF1 && key <= AF12) {
589:         do_switch_console = key - CONS_KEYS;
590:         return;
591:     }
592:
593:     if(shift && (key == PGUP)) {

```

drivers/char/keyboard.c

Page 10/12

```

594:         do_buf_scroll = SCROLL_UP;
595:         return;
596:     }
597:
598:     if(shift && (key == PGDN)) {
599:         do_buf_scroll = SCROLL_DOWN;
600:         return;
601:     }
602:
603:     if(extkey && (scode == SLASH_NPAD)) {
604:         key = SLASH;
605:     }
606:
607:     if(any_key_to_reboot) {
608:         reboot();
609:     }
610:
611:     if(tty->count) {
612:         type = key & 0xFF00;
613:         c = key & 0xFF;
614:
615:         if(sysrq) {
616:             /* treat 0-9 and a-z keys as normal */
617:             type &= ~META_KEYS;
618:         }
619:
620:         switch(type) {
621:             case FN_KEYS:
622:                 puts(tty, fn_seq[c]);
623:                 break;
624:
625:             case SPEC_KEYS:
626:                 switch(key) {
627:                     case CR:
628:                         putc(tty, C('M'));
629:                         break;
630:                     case SYSRQ:
631:                         sysrq = 1;
632:                         break;
633:                 }
634:                 break;
635:
636:             case PAD_KEYS:
637:                 if(!vc->numlock) {
638:                     puts(tty, pad_seq[c]);
639:                 } else {
640:                     putc(tty, pad_chars[c]);
641:                 }
642:                 break;
643:
644:             case DEAD_KEYS:
645:                 if(!deadkey) {
646:                     switch(c) {
647:                         case GRAVE ^ DEAD_KEYS:
648:                             deadkey = 1;
649:                             diacr = grave_table;
650:                             break;
651:                         case ACUTE ^ DEAD_KEYS:
652:                             deadkey = 2;
653:                             diacr = acute_table;
654:                             break;
655:                         case CIRCUM ^ DEAD_KEYS:
656:                             deadkey = 3;
657:                             diacr = circum_table;
658:                             break;
659:                         case DIERE ^ DEAD_KEYS:
660:                             deadkey = 5;

```

drivers/char/keyboard.c

Page 11/12

```

661:                                     diacr = diere_table;
662:                                     break;
663:                                     }
664:                                     return;
665:                                     }
666:                                     c = diacr_chars[c];
667:                                     deadkey = 0;
668:                                     putc(tty, c);
669:
670:                                     break;
671:
672:                                     case META_KEYS:
673:                                     putc(tty, '\033');
674:                                     putc(tty, c);
675:                                     break;
676:
677:                                     case LETTER_KEYS:
678:                                     if(deadkey) {
679:                                     for(n = 0; n < NR_DIACR; n++) {
680:                                     if(diacr[n].letter == c) {
681:                                     c = diacr[n].code;
682:                                     }
683:                                     }
684:                                     }
685:                                     putc(tty, c);
686:                                     break;
687:
688:                                     default:
689:                                     if(sysrq) {
690:                                     switch(c) {
691:                                     case 'l':
692:                                     sysrq_op = SYSRQ_STACK;
693:                                     break;
694:                                     case 't':
695:                                     sysrq_op = SYSRQ_TASKS;
696:                                     break;
697:                                     default:
698:                                     sysrq_op = SYSRQ_UNDEF;
699:                                     break;
700:                                     }
701:                                     return;
702:                                     }
703:                                     if(deadkey && c == ' ') {
704:                                     c = diacr_chars[deadkey - 1];
705:                                     }
706:                                     putc(tty, c);
707:                                     break;
708:                                     }
709:                                     }
710:                                     deadkey = 0;
711:                                     return;
712: }
713:
714: void irq_keyboard_bh(void)
715: {
716:     int n;
717:     struct tty *tty;
718:     struct vconsole *vc;
719:
720:     tty = get_tty(MKDEV(VCONSOLES_MAJOR, current_cons));
721:     vc = (struct vconsole *)tty->driver_data;
722:
723:     if(do_switch_console >= 0) {
724:         vconsole_select(do_switch_console);
725:         do_switch_console = -1;
726:     }
727:

```


drivers/char/keyboard.c

Page 12/12

```

728:         if(do_buf_scroll) {
729:             video.buf_scroll(vc, do_buf_scroll);
730:             do_buf_scroll = 0;
731:         }
732:
733:     tty = &tty_table[0];
734:     for(n = 0; n < NR_VCONSOLES; n++, tty++) {
735:         if(!tty->read_q.count) {
736:             continue;
737:         }
738:         if(tty->kbd.mode == K_RAW || tty->kbd.mode == K_MEDIUMRAW) {
739:             wakeup(&tty_read);
740:             continue;
741:         }
742:         if(lock_area(AREA_TTY_READ)) {
743:             keyboard_bh.flags |= BH_ACTIVE;
744:             continue;
745:         }
746:         tty->input(tty);
747:         unlock_area(AREA_TTY_READ);
748:     }
749:
750:     if(sysrq_op) {
751:         do_sysrq(sysrq_op);
752:         sysrq_op = 0;
753:     }
754: }
755:
756: void keyboard_init(void)
757: {
758:     struct tty *tty;
759:     struct vconsole *vc;
760:
761:     tty = get_tty(MKDEV(VCONSOLES_MAJOR, current_cons));
762:     vc = (struct vconsole *)tty->driver_data;
763:     video.screen_on(vc);
764:     video.cursor_blink((unsigned int)vc);
765:
766:     add_bh(&keyboard_bh);
767:     keyboard_reset();
768:
769:     /* flush buffers */
770:     while(!keyboard_wait_output()) {
771:         inport_b(KB_DATA);
772:     }
773:
774:     keyboard_identify();
775:
776:     keyboard_write(KB_DATA, KB_RATE);
777:     keyboard_wait_ack();
778:     keyboard_write(KB_DATA, DELAY_250 | RATE_30);
779:     keyboard_wait_ack();
780:
781:     printk("keyboard 0x%04X-0x%04X   %d   type=%s %s PS/2 devices=%d/%d\n
", 0x60, 0x64, KEYBOARD_IRQ, kb_identify[0] == 0xAB ? "MF2" : "unknown", ps2_iface & 0x
1 ? "MCA" : "AT", ps2_active_ports, ps2_supp_ports);
782:
783:     if(!register_irq(KEYBOARD_IRQ, &irq_config_keyboard)) {
784:         enable_irq(KEYBOARD_IRQ);
785:     }
786: }

```

drivers/char/lp.c

Page 1/4

```

1: /*
2:  * fiwix/drivers/char/lp.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/devices.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/errno.h>
12: #include <fiwix/lp.h>
13: #include <fiwix/stdio.h>
14: #include <fiwix/string.h>
15:
16: struct lp lp_table[LP_MINORS];
17:
18: static struct fs_operations lp_driver_fsop = {
19:     0,
20:     0,
21:
22:     lp_open,
23:     lp_close,
24:     NULL, /* read */
25:     lp_write,
26:     NULL, /* ioctl */
27:     NULL, /* lseek */
28:     NULL, /* readdir */
29:     NULL, /* mmap */
30:     NULL, /* select */
31:
32:     NULL, /* readlink */
33:     NULL, /* followlink */
34:     NULL, /* bmap */
35:     NULL, /* lockup */
36:     NULL, /* rmdir */
37:     NULL, /* link */
38:     NULL, /* unlink */
39:     NULL, /* symlink */
40:     NULL, /* mkdir */
41:     NULL, /* mknod */
42:     NULL, /* truncate */
43:     NULL, /* create */
44:     NULL, /* rename */
45:
46:     NULL, /* read_block */
47:     NULL, /* write_block */
48:
49:     NULL, /* read_inode */
50:     NULL, /* write_inode */
51:     NULL, /* ialloc */
52:     NULL, /* ifree */
53:     NULL, /* statfs */
54:     NULL, /* read_superblock */
55:     NULL, /* remount_fs */
56:     NULL, /* write_superblock */
57:     NULL, /* release_superblock */
58: };
59:
60: static struct device lp_device = {
61:     "lp",
62:     LP_MAJOR,
63:     { 0, 0, 0, 0, 0, 0, 0, 0 },
64:     0,
65:     NULL,
66:     &lp_driver_fsop,
67:     NULL

```

drivers/char/lp.c

Page 2/4

```

68: };
69:
70: struct lp lp_table[LP_MINORS] = {
71:     { LP0_ADDR, LP0_ADDR + 1, LP0_ADDR + 2, 0 }
72: };
73:
74: static void lp_delay(void)
75: {
76:     int n;
77:
78:     for(n = 0; n < 10000; n++) {
79:         NOP();
80:     }
81: }
82:
83: static int lp_ready(int minor)
84: {
85:     int n;
86:
87:     for(n = 0; n < LP_RDY_RETR; n++) {
88:         if(inport_b(lp_table[minor].stat) & LP_STAT_BUS) {
89:             break;
90:         }
91:         lp_delay();
92:     }
93:     if(n == LP_RDY_RETR) {
94:         return 0;
95:     }
96:     return 1;
97: }
98:
99: static int lp_probe(int minor)
100: {
101:     /* first check */
102:     outport_b(lp_table[minor].data, 0x55);
103:     lp_delay();
104:     if(inport_b(lp_table[minor].data) != 0x55) {
105:         return 1;          /* did not retain data */
106:     }
107:
108:     /* second check */
109:     outport_b(lp_table[minor].data, 0xAA);
110:     lp_delay();
111:     if(inport_b(lp_table[minor].data) != 0xAA) {
112:         return 1;          /* did not retain data */
113:     }
114:     return 0;
115: }
116:
117: static int lp_write_data(int minor, unsigned char c)
118: {
119:     unsigned char ctrl;
120:
121:     if(!lp_ready(minor)) {
122:         return -EBUSY;
123:     }
124:     outport_b(lp_table[minor].data, c);
125:     ctrl = inport_b(lp_table[minor].ctrl);
126:     outport_b(lp_table[minor].ctrl, ctrl | LP_CTRL_STR);
127:     lp_delay();
128:     outport_b(lp_table[minor].ctrl, ctrl);
129:     if(!lp_ready(minor)) {
130:         return -EBUSY;
131:     }
132:     return 1;
133: }
134:

```

drivers/char/lp.c

Page 3/4

```

135: int lp_open(struct inode *i, struct fd *fd_table)
136: {
137:     int minor;
138:
139:     minor = MINOR(i->rdev);
140:     if(!TEST_MINOR(lp_device.minors, minor)) {
141:         return -ENXIO;
142:     }
143:     if(!(lp_table[minor].flags & LP_CTRL_SEL)) {
144:         return -ENXIO;
145:     }
146:     if(lp_table[minor].flags & LP_STAT_BUS) {
147:         return -EBUSY;
148:     }
149:     lp_table[minor].flags |= LP_STAT_BUS;
150:     return 0;
151: }
152:
153: int lp_close(struct inode *i, struct fd *fd_table)
154: {
155:     int minor;
156:
157:     minor = MINOR(i->rdev);
158:     if(!TEST_MINOR(lp_device.minors, minor)) {
159:         return -ENXIO;
160:     }
161:     lp_table[minor].flags &= ~LP_STAT_BUS;
162:     return 0;
163: }
164:
165: int lp_write(struct inode *i, struct fd *fd_table, const char *buffer, __size_t
count)
166: {
167:     unsigned int n;
168:     int bytes_written, total_written;
169:     int minor;
170:
171:     minor = MINOR(i->rdev);
172:     if(!TEST_MINOR(lp_device.minors, minor)) {
173:         return -ENXIO;
174:     }
175:
176:     total_written = 0;
177:     for(n = 0; n < count; n++) {
178:         bytes_written = lp_write_data(minor, buffer[n]);
179:         if(bytes_written != 1) {
180:             break;
181:         }
182:         total_written += bytes_written;
183:     }
184:
185:     return total_written;
186: }
187:
188: void lp_init(void)
189: {
190:     int n;
191:     unsigned char ctrl;
192:
193:     for(n = 0; n < LP_MINORS; n++) {
194:         if(!lp_probe(n)) {
195:             ctrl = inport_b(lp_table[n].ctrl);
196:             ctrl &= ~LP_CTRL_AUT; /* disable auto LF */
197:             ctrl |= LP_CTRL_INI; /* initialize */
198:             ctrl |= LP_CTRL_SEL; /* select in */
199:             ctrl &= ~LP_CTRL_IRQ; /* disable IRQ */
200:             ctrl &= ~LP_CTRL_BID; /* disable bidirectional mode */

```

drivers/char/lp.c

Page 4/4

```
201:         outport_b(lp_table[n].ctrl, ctrl);
202:         lp_table[n].flags |= LP_CTRL_SEL;
203:         printk("lp%d      0x%04X-0x%04X      -      \n", n,
lp_table[n].data, lp_table[n].data + 2);
204:         SET_MINOR(lp_device.minors, n);
205:     }
206: }
207:
208:     for(n = 0; n < LP_MINORS; n++) {
209:         if(lp_table[n].flags & LP_CTRL_SEL) {
210:             if(register_device(CHR_DEV, &lp_device)) {
211:                 printk("WARNING: %s(): unable to register lp dev
ice.\n", __FUNCTION__);
212:             }
213:             break;
214:         }
215:     }
216: }
```

drivers/char/Makefile

Page 1/1

```
1: # fiwix/drivers/char/Makefile
2: #
3: # Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
4: # Distributed under the terms of the Fiwix License.
5: #
6:
7: .S.o:
8:     $(CC) -traditional -I$(INCLUDE) -c -o $@ $<
9: .c.o:
10:     $(CC) $(CFLAGS) -c -o $@ $<
11:
12: OBJS = console.o tty.o tty_queue.o vt.o defkeymap.o keyboard.o memdev.o \
13:     serial.o lp.o fb.o sysrq.o
14:
15: char:     $(OBJS)
16:     $(LD) $(LDFLAGS) -r $(OBJS) -o char.o
17:
18: clean:
19:     rm -f *.o
20:
```

drivers/char/memdev.c

Page 1/9

```

1: /*
2:  * fiwix/drivers/char/memdev.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/kernel.h>
10: #include <fiwix/memdev.h>
11: #include <fiwix/devices.h>
12: #include <fiwix/fs.h>
13: #include <fiwix/errno.h>
14: #include <fiwix/mm.h>
15: #include <fiwix/mman.h>
16: #include <fiwix/bios.h>
17: #include <fiwix/stdio.h>
18: #include <fiwix/string.h>
19:
20: static struct fs_operations mem_driver_fsop = {
21:     0,
22:     0,
23:
24:     mem_open,
25:     mem_close,
26:     mem_read,
27:     mem_write,
28:     NULL,                /* ioctl */
29:     mem_lseek,
30:     NULL,                /* readdir */
31:     mem_mmap,
32:     NULL,                /* select */
33:
34:     NULL,                /* readlink */
35:     NULL,                /* followlink */
36:     NULL,                /* bmap */
37:     NULL,                /* lockup */
38:     NULL,                /* rmdir */
39:     NULL,                /* link */
40:     NULL,                /* unlink */
41:     NULL,                /* symlink */
42:     NULL,                /* mkdir */
43:     NULL,                /* mknod */
44:     NULL,                /* truncate */
45:     NULL,                /* create */
46:     NULL,                /* rename */
47:
48:     NULL,                /* read_block */
49:     NULL,                /* write_block */
50:
51:     NULL,                /* read_inode */
52:     NULL,                /* write_inode */
53:     NULL,                /* ialloc */
54:     NULL,                /* ifree */
55:     NULL,                /* statfs */
56:     NULL,                /* read_superblock */
57:     NULL,                /* remount_fs */
58:     NULL,                /* write_superblock */
59:     NULL,                /* release_superblock */
60: };
61:
62: static struct fs_operations kmem_driver_fsop = {
63:     0,
64:     0,
65:
66:     kmem_open,
67:     kmem_close,

```

drivers/char/memdev.c

Page 2/9

```

68:         kmem_read,
69:         kmem_write,
70:         NULL,                /* ioctl */
71:         kmem_lseek,
72:         NULL,                /* readdir */
73:         mem_mmap,
74:         NULL,                /* select */
75:
76:         NULL,                /* readlink */
77:         NULL,                /* followlink */
78:         NULL,                /* bmap */
79:         NULL,                /* lockup */
80:         NULL,                /* rmdir */
81:         NULL,                /* link */
82:         NULL,                /* unlink */
83:         NULL,                /* symlink */
84:         NULL,                /* mkdir */
85:         NULL,                /* mknod */
86:         NULL,                /* truncate */
87:         NULL,                /* create */
88:         NULL,                /* rename */
89:
90:         NULL,                /* read_block */
91:         NULL,                /* write_block */
92:
93:         NULL,                /* read_inode */
94:         NULL,                /* write_inode */
95:         NULL,                /* ialloc */
96:         NULL,                /* ifree */
97:         NULL,                /* statfs */
98:         NULL,                /* read_superblock */
99:         NULL,                /* remount_fs */
100:        NULL,                /* write_superblock */
101:        NULL,                /* release_superblock */
102:    };
103:
104:    static struct fs_operations null_driver_fsop = {
105:        0,
106:        0,
107:
108:        null_open,
109:        null_close,
110:        null_read,
111:        null_write,
112:        NULL,                /* ioctl */
113:        null_lseek,
114:        NULL,                /* readdir */
115:        NULL,                /* mmap */
116:        NULL,                /* select */
117:
118:        NULL,                /* readlink */
119:        NULL,                /* followlink */
120:        NULL,                /* bmap */
121:        NULL,                /* lockup */
122:        NULL,                /* rmdir */
123:        NULL,                /* link */
124:        NULL,                /* unlink */
125:        NULL,                /* symlink */
126:        NULL,                /* mkdir */
127:        NULL,                /* mknod */
128:        NULL,                /* truncate */
129:        NULL,                /* create */
130:        NULL,                /* rename */
131:
132:        NULL,                /* read_block */
133:        NULL,                /* write_block */
134:

```


drivers/char/memdev.c

Page 3/9

```

135:     NULL,                /* read_inode */
136:     NULL,                /* write_inode */
137:     NULL,                /* ialloc */
138:     NULL,                /* ifree */
139:     NULL,                /* statfs */
140:     NULL,                /* read_superblock */
141:     NULL,                /* remount_fs */
142:     NULL,                /* write_superblock */
143:     NULL,                /* release_superblock */
144: };
145:
146: static struct fs_operations zero_driver_fsop = {
147:     0,
148:     0,
149:
150:     zero_open,
151:     zero_close,
152:     zero_read,
153:     zero_write,
154:     NULL,                /* ioctl */
155:     zero_lseek,
156:     NULL,                /* readdir */
157:     NULL,                /* mmap */
158:     NULL,                /* select */
159:
160:     NULL,                /* readlink */
161:     NULL,                /* followlink */
162:     NULL,                /* bmap */
163:     NULL,                /* lockup */
164:     NULL,                /* rmdir */
165:     NULL,                /* link */
166:     NULL,                /* unlink */
167:     NULL,                /* symlink */
168:     NULL,                /* mkdir */
169:     NULL,                /* mknod */
170:     NULL,                /* truncate */
171:     NULL,                /* create */
172:     NULL,                /* rename */
173:
174:     NULL,                /* read_block */
175:     NULL,                /* write_block */
176:
177:     NULL,                /* read_inode */
178:     NULL,                /* write_inode */
179:     NULL,                /* ialloc */
180:     NULL,                /* ifree */
181:     NULL,                /* statfs */
182:     NULL,                /* read_superblock */
183:     NULL,                /* remount_fs */
184:     NULL,                /* write_superblock */
185:     NULL,                /* release_superblock */
186: };
187:
188: static struct fs_operations memdev_driver_fsop = {
189:     0,
190:     0,
191:
192:     memdev_open,
193:     NULL,                /* close */
194:     NULL,                /* read */
195:     NULL,                /* write */
196:     NULL,                /* ioctl */
197:     NULL,                /* lseek */
198:     NULL,                /* readdir */
199:     NULL,                /* mmap */
200:     NULL,                /* select */
201:

```

drivers/char/memdev.c

Page 4/9

```

202:         NULL,                /* readlink */
203:         NULL,                /* followlink */
204:         NULL,                /* bmap */
205:         NULL,                /* lockup */
206:         NULL,                /* rmdir */
207:         NULL,                /* link */
208:         NULL,                /* unlink */
209:         NULL,                /* symlink */
210:         NULL,                /* mkdir */
211:         NULL,                /* mknod */
212:         NULL,                /* truncate */
213:         NULL,                /* create */
214:         NULL,                /* rename */
215:
216:         NULL,                /* read_block */
217:         NULL,                /* write_block */
218:
219:         NULL,                /* read_inode */
220:         NULL,                /* write_inode */
221:         NULL,                /* ialloc */
222:         NULL,                /* ifree */
223:         NULL,                /* statfs */
224:         NULL,                /* read_superblock */
225:         NULL,                /* remount_fs */
226:         NULL,                /* write_superblock */
227:         NULL,                /* release_superblock */
228: };
229:
230: static struct fs_operations urandom_driver_fsop = {
231:     0,
232:     0,
233:
234:     urandom_open,
235:     urandom_close,
236:     urandom_read,
237:     urandom_write,
238:     NULL,                /* ioctl */
239:     urandom_lseek,
240:     NULL,                /* readdir */
241:     NULL,                /* mmap */
242:     NULL,                /* select */
243:
244:     NULL,                /* readlink */
245:     NULL,                /* followlink */
246:     NULL,                /* bmap */
247:     NULL,                /* lockup */
248:     NULL,                /* rmdir */
249:     NULL,                /* link */
250:     NULL,                /* unlink */
251:     NULL,                /* symlink */
252:     NULL,                /* mkdir */
253:     NULL,                /* mknod */
254:     NULL,                /* truncate */
255:     NULL,                /* create */
256:     NULL,                /* rename */
257:
258:     NULL,                /* read_block */
259:     NULL,                /* write_block */
260:
261:     NULL,                /* read_inode */
262:     NULL,                /* write_inode */
263:     NULL,                /* ialloc */
264:     NULL,                /* ifree */
265:     NULL,                /* statfs */
266:     NULL,                /* read_superblock */
267:     NULL,                /* remount_fs */
268:     NULL,                /* write_superblock */

```

drivers/char/memdev.c

Page 5/9

```

269:         NULL                                     /* release_superblock */
270:     };
271:
272:     static struct device memdev_device = {
273:         "mem",
274:         MEMDEV_MAJOR,
275:         { 0, 0, 0, 0, 0, 0, 0, 0, 0 },
276:         0,
277:         NULL,
278:         &memdev_driver_fsop,
279:         NULL
280:     };
281:
282:     int mem_open(struct inode *i, struct fd *fd_table)
283:     {
284:         return 0;
285:     }
286:
287:     int mem_close(struct inode *i, struct fd *fd_table)
288:     {
289:         return 0;
290:     }
291:
292:     int mem_read(struct inode *i, struct fd *fd_table, char *buffer, __size_t count)
293:     {
294:         unsigned int physical_memory;
295:
296:         physical_memory = (kstat.physical_pages << PAGE_SHIFT);
297:         if(fd_table->offset >= physical_memory) {
298:             return 0;
299:         }
300:         count = MIN(count, physical_memory - fd_table->offset);
301:         memcpy_b(buffer, (void *)P2V(fd_table->offset), count);
302:         fd_table->offset += count;
303:         return count;
304:     }
305:
306:     int mem_write(struct inode *i, struct fd *fd_table, const char *buffer, __size_t
count)
307:     {
308:         unsigned int physical_memory;
309:
310:         physical_memory = (kstat.physical_pages << PAGE_SHIFT);
311:         if(fd_table->offset >= physical_memory) {
312:             return 0;
313:         }
314:         count = MIN(count, physical_memory - fd_table->offset);
315:         memcpy_b((void *)P2V(fd_table->offset), buffer, count);
316:         fd_table->offset += count;
317:         return count;
318:     }
319:
320:     int mem_lseek(struct inode *i, __off_t offset)
321:     {
322:         return offset;
323:     }
324:
325:     int kmem_open(struct inode *i, struct fd *fd_table)
326:     {
327:         return 0;
328:     }
329:
330:     int kmem_close(struct inode *i, struct fd *fd_table)
331:     {
332:         return 0;
333:     }
334:

```

drivers/char/memdev.c

Page 6/9

```

335: int kmem_read(struct inode *i, struct fd *fd_table, char *buffer, __size_t count
)
336: {
337:     unsigned int physical_memory;
338:
339:     physical_memory = P2V((kstat.physical_pages << PAGE_SHIFT));
340:     if(P2V(fd_table->offset + count) < physical_memory) {
341:         memcpy_b(buffer, (void *)P2V(fd_table->offset), count);
342:         fd_table->offset += count;
343:     } else {
344:         count = 0;
345:     }
346:     return count;
347: }
348:
349: int kmem_write(struct inode *i, struct fd *fd_table, const char *buffer, __size_
t count)
350: {
351:     unsigned int physical_memory;
352:
353:     physical_memory = P2V((kstat.physical_pages << PAGE_SHIFT));
354:     if(P2V(fd_table->offset + count) < physical_memory) {
355:         memcpy_b((void *)P2V(fd_table->offset), buffer, count);
356:         fd_table->offset += count;
357:     } else {
358:         count = 0;
359:     }
360:     return count;
361: }
362:
363: int kmem_lseek(struct inode *i, __off_t offset)
364: {
365:     return offset;
366: }
367:
368: int null_open(struct inode *i, struct fd *fd_table)
369: {
370:     return 0;
371: }
372:
373: int null_close(struct inode *i, struct fd *fd_table)
374: {
375:     return 0;
376: }
377:
378: int null_read(struct inode *i, struct fd *fd_table, char *buffer, __size_t count
)
379: {
380:     return 0;
381: }
382:
383: int null_write(struct inode *i, struct fd *fd_table, const char *buffer, __size_
t count)
384: {
385:     return count;
386: }
387:
388: int null_lseek(struct inode *i, __off_t offset)
389: {
390:     return offset;
391: }
392:
393: int zero_open(struct inode *i, struct fd *fd_table)
394: {
395:     return 0;
396: }
397:

```

drivers/char/memdev.c

Page 7/9

```

398: int zero_close(struct inode *i, struct fd *fd_table)
399: {
400:     return 0;
401: }
402:
403: int zero_read(struct inode *i, struct fd *fd_table, char *buffer, __size_t count
)
404: {
405:     memset_b(buffer, NULL, count);
406:     return count;
407: }
408:
409: int zero_write(struct inode *i, struct fd *fd_table, const char *buffer, __size_
t count)
410: {
411:     return count;
412: }
413:
414: int zero_lseek(struct inode *i, __off_t offset)
415: {
416:     return offset;
417: }
418:
419: int urandom_open(struct inode *i, struct fd *fd_table)
420: {
421:     return 0;
422: }
423:
424: int urandom_close(struct inode *i, struct fd *fd_table)
425: {
426:     return 0;
427: }
428:
429: int urandom_read(struct inode *i, struct fd *fd_table, char *buffer, __size_t co
unt)
430: {
431:     int n;
432:
433:     for(n = 0; n < count; n++) {
434:         kstat.random_seed = kstat.random_seed * 1103515245 + 12345;
435:         *buffer = (char)(unsigned int)(kstat.random_seed / 65536) % 256;
436:         buffer++;
437:     }
438:     return count;
439: }
440:
441: int urandom_write(struct inode *i, struct fd *fd_table, const char *buffer, __si
ze_t count)
442: {
443:     return count;
444: }
445:
446: int urandom_lseek(struct inode *i, __off_t offset)
447: {
448:     return offset;
449: }
450:
451: int memdev_open(struct inode *i, struct fd *fd_table)
452: {
453:     unsigned char minor;
454:
455:     minor = MINOR(i->rdev);
456:     switch(minor) {
457:         case MEMDEV_MEM:
458:             i->fsop = &mem_driver_fsop;
459:             break;
460:         case MEMDEV_KMEM:

```

drivers/char/memdev.c

Page 8/9

```

461:             i->fsop = &kmem_driver_fsop;
462:             break;
463:         case MEMDEV_NULL:
464:             i->fsop = &null_driver_fsop;
465:             break;
466:         case MEMDEV_ZERO:
467:             i->fsop = &zero_driver_fsop;
468:             break;
469:         case MEMDEV_RANDOM:
470:             i->fsop = &urandom_driver_fsop;
471:             break;
472:         case MEMDEV_URANDOM:
473:             i->fsop = &urandom_driver_fsop;
474:             break;
475:         default:
476:             return -ENXIO;
477:     }
478:     if(i->fsop->open) {
479:         return i->fsop->open(i, fd_table);
480:     }
481:     return 0;
482: }
483:
484: /*
485:  * This function maps a range of physical addresses marked as not available for
486:  * use in the BIOS memory map, like the video RAM.
487:  */
488: int mem_mmap(struct inode *i, struct vma *vma)
489: {
490:     unsigned int addr, length;
491:
492:     length = (vma->end - vma->start) & PAGE_MASK;
493:
494:     /* this breaks down the range in 4KB chunks */
495:     for(addr = 0; addr < length; addr += PAGE_SIZE) {
496:         /* map the page only if is NOT available in the BIOS map */
497:         if(!addr_in_bios_map(vma->offset + addr)) {
498:             if(!map_page(current, (vma->start + addr) & PAGE_MASK, (
vma->offset + addr) & PAGE_MASK, PROT_READ | PROT_WRITE)) {
499:                 return -ENOMEM;
500:             }
501:         } else {
502:             printk("ERROR: %s(): mapping AVAILABLE pages in BIOS mem
ory map isn't supported.\n", __FUNCTION__);
503:             printk("\tinvalid mapping: 0x%08x -> 0x%08x\n", (vma->st
art + addr) & PAGE_MASK, (vma->offset + addr) & PAGE_MASK);
504:             return -EAGAIN;
505:         }
506:     }
507:     invalidate_tlb();
508:     return 0;
509: }
510:
511: void memdev_init(void)
512: {
513:     SET_MINOR(memdev_device.minors, MEMDEV_MEM);
514:     SET_MINOR(memdev_device.minors, MEMDEV_KMEM);
515:     SET_MINOR(memdev_device.minors, MEMDEV_NULL);
516:     SET_MINOR(memdev_device.minors, MEMDEV_ZERO);
517:     SET_MINOR(memdev_device.minors, MEMDEV_RANDOM);
518:     SET_MINOR(memdev_device.minors, MEMDEV_URANDOM);
519:
520:     if(register_device(CHR_DEV, &memdev_device)) {
521:         printk("ERROR: %s(): unable to register memory devices.\n", __FU
NCTION__);
522:         return;
523:     }

```

```
524: }
```

drivers/char/serial.c

Page 1/9

```

1: /*
2:  * fiwix/drivers/char/serial.c
3:  *
4:  * Copyright 2020-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/kernel.h>
10: #include <fiwix/devices.h>
11: #include <fiwix/fs.h>
12: #include <fiwix/errno.h>
13: #include <fiwix/pic.h>
14: #include <fiwix/sleep.h>
15: #include <fiwix/serial.h>
16: #include <fiwix/tty.h>
17: #include <fiwix/ctype.h>
18: #include <fiwix/stdio.h>
19: #include <fiwix/string.h>
20:
21: static struct fs_operations serial_driver_fsop = {
22:     0,
23:     0,
24:
25:     tty_open,
26:     tty_close,
27:     tty_read,
28:     tty_write,
29:     tty_ioctl,
30:     tty_lseek,
31:     NULL,                /* readdir */
32:     NULL,                /* mmap */
33:     tty_select,
34:
35:     NULL,                /* readlink */
36:     NULL,                /* followlink */
37:     NULL,                /* bmap */
38:     NULL,                /* lockup */
39:     NULL,                /* rmdir */
40:     NULL,                /* link */
41:     NULL,                /* unlink */
42:     NULL,                /* symlink */
43:     NULL,                /* mkdir */
44:     NULL,                /* mknod */
45:     NULL,                /* truncate */
46:     NULL,                /* create */
47:     NULL,                /* rename */
48:
49:     NULL,                /* read_block */
50:     NULL,                /* write_block */
51:
52:     NULL,                /* read_inode */
53:     NULL,                /* write_inode */
54:     NULL,                /* ialloc */
55:     NULL,                /* ifree */
56:     NULL,                /* statfs */
57:     NULL,                /* read_superblock */
58:     NULL,                /* remount_fs */
59:     NULL,                /* write_superblock */
60:     NULL,                /* release_superblock */
61: };
62:
63: static struct device serial_device = {
64:     "ttyS",
65:     SERIAL_MAJOR,
66:     { 0, 0, 0, 0, 0, 0, 0, 0, 0 },
67:     0,

```


drivers/char/serial.c

Page 2/9

```

68:         NULL,
69:         &serial_driver_fsop,
70:         NULL
71: };
72:
73: /* 9600,N,8,1 by default */
74: struct serial serial_table[NR_SERIAL] = {
75:     { 0x3F8, SERIAL4_IRQ, 9600, "ttyS0", UART_LCR_NP | UART_LCR_WL8 | UART_L
CR_1STB, 0, NULL, NULL },
76:     { 0x2F8, SERIAL3_IRQ, 9600, "ttyS1", UART_LCR_NP | UART_LCR_WL8 | UART_L
CR_1STB, 0, NULL, NULL },
77:     { 0x3E8, SERIAL4_IRQ, 9600, "ttyS2", UART_LCR_NP | UART_LCR_WL8 | UART_L
CR_1STB, 0, NULL, NULL },
78:     { 0x2E8, SERIAL3_IRQ, 9600, "ttyS3", UART_LCR_NP | UART_LCR_WL8 | UART_L
CR_1STB, 0, NULL, NULL }
79: };
80:
81: char *serial_chip[] = {
82:     NULL,
83:     "8250",
84:     "16450",
85:     "16550",
86:     "16550A",
87: };
88:
89: static int baud_table[] = {
90:     0,
91:     50,
92:     75,
93:     110,
94:     134,
95:     150,
96:     200,
97:     300,
98:     600,
99:     1200,
100:    1800,
101:    2400,
102:    4800,
103:    9600,
104:   19200,
105:   38400,
106:   57200,
107:  115200,
108:    0
109: };
110:
111: static struct serial *serial_ports = NULL;
112: static struct bh serial_bh = { 0, &irq_serial_bh, NULL };
113: static struct interrupt irq_config_serial0 = { 0, "serial", &irq_serial, NULL };
114: static struct interrupt irq_config_serial1 = { 0, "serial", &irq_serial, NULL };
115:
116: static int is_serial(__dev_t dev)
117: {
118:     if (MAJOR(dev) == SERIAL_MAJOR && ((MINOR(dev) >= (1 << SERIAL_MSF) && MI
NOR(dev) < (1 << SERIAL_MSF) + SERIAL_MINORS))) {
119:         return 1;
120:     }
121:
122:     return 0;
123: }
124:
125: static int serial_identify(struct serial *s)
126: {
127:     int value;
128:
129:     /* set all features in FCR register to test the status of FIFO */

```

drivers/char/serial.c

Page 3/9

```

130:         outport_b(s->addr + UART_FCR, (UART_FCR_FIFO |
131:                                         UART_FCR_DMA |
132:                                         UART_FCR_FIFO64 |
133:                                         UART_FCR_FIFO14));
134:
135:         value = inport_b(s->addr + UART_IIR);
136:         if(value & UART_IIR_FIFOKO) {
137:             if(value & UART_IIR_FIFO) {
138:                 if(value & UART_IIR_FIFO64) {
139:                     /* 16750 chip is not supported */
140:                 } else {
141:                     s->flags |= UART_IS_16550A | UART_HAS_FIFO;
142:                     return 4;
143:                 }
144:             } else {
145:                 s->flags |= UART_IS_16550;
146:                 return 3;
147:             }
148:         } else {
149:             /*
150:              * At this point we know this device don't has FIFO,
151:              * the Scratch Register will help us to know the final chip.
152:              */
153:             value = inport_b(s->addr + UART_SR); /* save its value */
154:             outport_b(s->addr + UART_SR, 0xAA); /* put a random value */
155:             if(inport_b(s->addr + UART_SR) != 0xAA) {
156:                 s->flags |= UART_IS_8250;
157:                 return 1;
158:             } else {
159:                 outport_b(s->addr + UART_SR, value); /* restore it */
160:                 s->flags |= UART_IS_16450;
161:                 return 2;
162:             }
163:         }
164:         return 0;
165:     }
166:
167:     static void serial_default(struct serial *s)
168:     {
169:         int divisor;
170:
171:         outport_b(s->addr + UART_IER, 0); /* disable all interrupts */
172:
173:         divisor = 115200 / s->baud;
174:         outport_b(s->addr + UART_LCR, UART_LCR_DLAB); /* enable DLAB */
175:         outport_b(s->addr + UART_DLL, divisor & 0xFF); /* LSB of divisor */
176:         outport_b(s->addr + UART_DLH, divisor >> 8); /* MSB of divisor */
177:         outport_b(s->addr + UART_LCR, s->lctrl); /* line control */
178:     }
179:
180:     /* disable transmitter interrupts */
181:     static void serial_stop(struct tty *tty)
182:     {
183:         struct serial *s;
184:         unsigned long int flags;
185:
186:         SAVE_FLAGS(flags); CLI();
187:         s = (struct serial *)tty->driver_data;
188:         outport_b(s->addr + UART_IER, UART_IER_RDAI);
189:         RESTORE_FLAGS(flags);
190:     }
191:
192:     /* enable transmitter interrupts */
193:     static void serial_start(struct tty *tty)
194:     {
195:         struct serial *s;
196:         unsigned long int flags;

```

drivers/char/serial.c

Page 4/9

```

197:
198:     SAVE_FLAGS(flags); CLI();
199:     s = (struct serial *)tty->driver_data;
200:     outport_b(s->addr + UART_IER, UART_IER_RDAI | UART_IER_THREI);
201:     RESTORE_FLAGS(flags);
202: }
203:
204: static void serial_deltab(struct tty *tty)
205: {
206:     unsigned short int col, n, count;
207:     struct cblock *cb;
208:     unsigned char ch;
209:
210:     cb = tty->cooked_q.head;
211:     col = count = 0;
212:
213:     while(cb) {
214:         for(n = 0; n < cb->end_off; n++) {
215:             if(n >= cb->start_off) {
216:                 ch = cb->data[n];
217:                 if(ch == '\\t') {
218:                     while(!tty->tab_stop[++col]);
219:                 } else {
220:                     col++;
221:                     if(ISCNTRL(ch) && !ISSPACE(ch) && tty->t
ermios.c_lflag & ECHOCTL) {
222:                         col++;
223:                     }
224:                 }
225:                 col %= 80;
226:             }
227:         }
228:         cb = cb->next;
229:     }
230:     count = tty->column - col;
231:
232:     while(count-->0) {
233:         tty_queue_putchar(tty, &tty->write_q, '\\b');
234:         tty->column--;
235:     }
236: }
237:
238: static void serial_reset(struct tty *tty)
239: {
240:     termios_reset(tty);
241:     tty->winsize.ws_row = 25;
242:     tty->winsize.ws_col = 80;
243:     tty->winsize.ws_xpixel = 0;
244:     tty->winsize.ws_ypixel = 0;
245:     tty->lnext = 0;
246: }
247:
248: static void serial_errors(struct serial *s, int status)
249: {
250:     struct tty *tty;
251:
252:     tty = s->tty;
253:
254:     if(!(tty->termios.c_iflag & IGNBRK) && tty->termios.c_iflag & BRKINT) {
255:         if(status & UART_LSR_BI) {
256:             printf("WARNING: break interrupt in %s.\n", s->name);
257:         }
258:     }
259:
260:     /* this includes also overrun errors */
261:     if(!(tty->termios.c_iflag & IGNPAR) && tty->termios.c_iflag & PARMRK) {
262:         if(status & UART_LSR_OE) {

```

drivers/char/serial.c

Page 5/9

```

263:             printk("WARNING: overrun error in %s.\n", s->name);
264:
265:         } else if(status & UART_LSR_PE) {
266:             printk("WARNING: parity error in %s.\n", s->name);
267:
268:         } else if(status & UART_LSR_FE) {
269:             printk("WARNING: framing error in %s.\n", s->name);
270:
271:         } else if(status & UART_LSR_EFIFO) {
272:             printk("WARNING: FIFO error in %s.\n", s->name);
273:         }
274:     }
275: }
276:
277: static void serial_send(struct tty *tty)
278: {
279:     unsigned char ch;
280:     struct serial *s;
281:     int count;
282:
283:     s = (struct serial *)tty->driver_data;
284:
285:     if(!tty->write_q.count) {
286:         outport_b(s->addr + UART_IER, UART_IER_RDAI);
287:         return;
288:     }
289:
290:     count = 0;
291:     while(tty->write_q.count > 0 && count < UART_FIFO_SIZE) {
292:         ch = tty_queue_getchar(&tty->write_q);
293:         outport_b(s->addr + UART_TD, ch);
294:         count++;
295:     }
296:
297:     if(!tty->write_q.count) {
298:         outport_b(s->addr + UART_IER, UART_IER_RDAI);
299:     }
300:     wakeup(&tty_write);
301: }
302:
303: static int serial_receive(struct serial *s)
304: {
305:     int status, errno;
306:     unsigned char ch;
307:     struct tty *tty;
308:
309:     errno = 0;
310:     tty = s->tty;
311:
312:     do {
313:         if(!tty_queue_room(&tty->read_q)) {
314:             errno = -EAGAIN;
315:             break;
316:         }
317:         ch = inport_b(s->addr + UART_RD);
318:         tty_queue_putchar(tty, &tty->read_q, ch);
319:         status = inport_b(s->addr + UART_LSR);
320:     } while(status & UART_LSR_RDA);
321:
322:     serial_bh.flags |= BH_ACTIVE;
323:     return errno;
324: }
325:
326: void irq_serial(int num, struct sigcontext *sc)
327: {
328:     struct serial *s;
329:     int status;

```

drivers/char/serial.c

Page 6/9

```

330:
331:     s = serial_ports;
332:
333:     if(s) {
334:         do {
335:             if(s->irq == num) {
336:                 while(! (inport_b(s->addr + UART_IIR) & UART_IIR_
NOINT)) {
337:                     status = inport_b(s->addr + UART_LSR);
338:                     if(status & UART_LSR_RDA) {
339:                         if(serial_receive(s)) {
340:                             break;
341:                         }
342:                     }
343:                     if(status & UART_LSR_THRE) {
344:                         serial_send(s->tty);
345:                     }
346:                     serial_errors(s, status);
347:                 }
348:             }
349:             s = s->next;
350:         } while(s);
351:     }
352: }
353:
354: int serial_open(struct tty *tty)
355: {
356:     struct serial *s;
357:     int minor;
358:
359:     minor = MINOR(tty->dev);
360:     if(!TEST_MINOR(serial_device.minors, minor)) {
361:         return -ENXIO;
362:     }
363:
364:     s = (struct serial *)tty->driver_data;
365:
366:     /* enable FIFO */
367:     if(s->flags & UART_HAS_FIFO) {
368:         outport_b(s->addr + UART_FCR, UART_FCR_FIFO | UART_FCR_FIFO14);
369:     }
370:     outport_b(s->addr + UART_MCR, UART_MCR_OUT2 | UART_MCR_RTS | UART_MCR_DT
R);
371:
372:     /* enable interrupts */
373:     outport_b(s->addr + UART_IER, UART_IER_RDAI);
374:
375:     /* clear all input registers */
376:     inport_b(s->addr + UART_RD);
377:     inport_b(s->addr + UART_IIR);
378:     inport_b(s->addr + UART_LSR);
379:     inport_b(s->addr + UART_MSR);
380:
381:     return 0;
382: }
383:
384: int serial_close(struct tty *tty)
385: {
386:     struct serial *s;
387:     int minor;
388:
389:     minor = MINOR(tty->dev);
390:     if(!TEST_MINOR(serial_device.minors, minor)) {
391:         return -ENXIO;
392:     }
393:
394:     s = (struct serial *)tty->driver_data;

```

drivers/char/serial.c

Page 7/9

```

395:
396:     if(tty->count > 1) {
397:         return 0;
398:     }
399:
400:     /* disable all interrupts */
401:     outport_b(s->addr + UART_IER, 0);
402:
403:     /* disable FIFO */
404:     outport_b(s->addr + UART_FCR, UART_FCR_CRCVCR | UART_FCR_CXMTR);
405:
406:     /* clear all input register */
407:     inport_b(s->addr + UART_RD);
408:
409:     return 0;
410: }
411:
412: void serial_set_termios(struct tty *tty)
413: {
414:     short int divisor;
415:     int baud, size, stop;
416:     int lctrl;
417:     struct serial *s;
418:
419:     s = (struct serial *)tty->driver_data;
420:     lctrl = 0;
421:
422:     if(!(baud = tty->termios.c_cflag & CBAUD)) {
423:         return;
424:     }
425:     divisor = 115200 / baud_table[baud];
426:
427:     outport_b(s->addr + UART_LCR, UART_LCR_DLAB); /* enable DLAB */
428:     outport_b(s->addr + UART_DLL, divisor & 0xFF); /* LSB of divisor */
429:     outport_b(s->addr + UART_DLH, divisor >> 8); /* MSB of divisor */
430:
431:     size = tty->termios.c_cflag & CSIZE;
432:     switch(size) {
433:         case CS5:
434:             lctrl = UART_LCR_WL5;
435:             break;
436:         case CS6:
437:             lctrl = UART_LCR_WL6;
438:             break;
439:         case CS7:
440:             lctrl = UART_LCR_WL7;
441:             break;
442:         case CS8:
443:             lctrl = UART_LCR_WL8;
444:             break;
445:         default:
446:             lctrl = UART_LCR_WL5;
447:             break;
448:     }
449:
450:     stop = tty->termios.c_cflag & CSTOPB;
451:     if(stop) {
452:         lctrl |= UART_LCR_2STB;
453:     } else {
454:         lctrl |= UART_LCR_1STB;
455:     }
456:
457:     if(tty->termios.c_cflag & PARENB) {
458:         lctrl |= UART_LCR_EP;
459:     } else if(tty->termios.c_cflag & PARODD) {
460:         lctrl |= UART_LCR_OP;
461:     } else {

```

drivers/char/serial.c

Page 8/9

```

462:         lctrl |= UART_LCR_NP;
463:     }
464:
465:     /* FIXME: flow control RTSCTS no supported */
466:
467:     outport_b(s->addr + UART_LCR, lctrl); /* line control */
468: }
469:
470: void serial_write(struct tty *tty)
471: {
472:     struct serial *s;
473:     unsigned long int flags;
474:
475:     SAVE_FLAGS(flags); CLI();
476:     s = (struct serial *)tty->driver_data;
477:     outport_b(s->addr + UART_IER, UART_IER_RDAI | UART_IER_THREI);
478:     RESTORE_FLAGS(flags);
479: }
480:
481: void irq_serial_bh(void)
482: {
483:     struct tty *tty;
484:     struct serial *s;
485:
486:     s = serial_ports;
487:
488:     if(s) {
489:         do {
490:             tty = s->tty;
491:             if(tty->read_q.count) {
492:                 if(!lock_area(AREA_SERIAL_READ)) {
493:                     tty->input(tty);
494:                     unlock_area(AREA_SERIAL_READ);
495:                 } else {
496:                     serial_bh.flags |= BH_ACTIVE;
497:                 }
498:             }
499:             s = s->next;
500:         } while(s);
501:     }
502: }
503:
504: void serial_init(void)
505: {
506:     int n, n2, type, found;
507:     struct serial **sp, *s;
508:     struct tty *tty;
509:
510:     for(n = 0, found = 0; n < SERIAL_MINORS; n++) {
511:         s = &serial_table[n];
512:         if((type = serial_identify(s)) {
513:             printk("ttyS%d    0x%04X-0x%04X    %d    type=%s%s\n",
n, s->addr, s->addr + 7, s->irq, serial_chip[type], s->flags & UART_HAS_FIFO ? " FIFO=y
es" : "");
514:
515:             SET_MINOR(serial_device.minors, (1 << SERIAL_MSF) + n);
516:             serial_default(s);
517:             sp = &serial_ports;
518:             if(*sp) {
519:                 do {
520:                     sp = &(*sp)->next;
521:                 } while(*sp);
522:             }
523:             if(!register_tty(MKDEV(SERIAL_MAJOR, (1 << SERIAL_MSF) +
n))) {
524:                 tty = get_tty(MKDEV(SERIAL_MAJOR, (1 << SERIAL_M
SF) + n));

```

drivers/char/serial.c

Page 9/9

```

525:         tty->driver_data = (void *)s;
526:         tty->stop = serial_stop;
527:         tty->start = serial_start;
528:         tty->deltab = serial_deltab;
529:         tty->reset = serial_reset;
530:         tty->input = do_cook;
531:         tty->output = serial_write;
532:         tty->open = serial_open;
533:         tty->close = serial_close;
534:         tty->set_termios = serial_set_termios;
535:         serial_reset(tty);
536:         for(n2 = 0; n2 < MAX_TAB_COLS; n2++) {
537:             if(!(n2 % TAB_SIZE)) {
538:                 tty->tab_stop[n2] = 1;
539:             } else {
540:                 tty->tab_stop[n2] = 0;
541:             }
542:         }
543:         tty->count = 0;
544:         s->tty = tty;
545:         *sp = s;
546:         found++;
547:     } else {
548:         printk("WARNING: %s(): unable to register ttyS%d
.\n", __FUNCTION__, n);
549:     }
550: }
551: }
552: if(found) {
553:     add_bh(&serial_bh);
554:     if(register_device(CHR_DEV, &serial_device)) {
555:         printk("WARNING: %s(): unable to register serial device.
\n", __FUNCTION__);
556:     }
557:     if(!register_irq(SERIAL4_IRQ, &irq_config_serial0)) {
558:         enable_irq(SERIAL4_IRQ);
559:     }
560:     if(found > 1) {
561:         if(!register_irq(SERIAL3_IRQ, &irq_config_serial1)) {
562:             enable_irq(SERIAL3_IRQ);
563:         }
564:     }
565:
566:     if(is_serial(_syscondev)) {
567:         register_console(console_flush_log_buf);
568:     }
569: }
570: }

```


drivers/char/sysrq.c

Page 1/1

```

1: /*
2:  * fiwix/drivers/char/sysrq.c
3:  *
4:  * Copyright 2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/sysrq.h>
10: #include <fiwix/traps.h>
11: #include <fiwix/stdio.h>
12: #include <fiwix/string.h>
13: #include <fiwix/sleep.h>
14: #include <fiwix/sched.h>
15: #include <fiwix/mm.h>
16:
17: static const char *pstate[] = {
18:     "?",
19:     "R",
20:     "S",
21:     "Z",
22:     "T",
23:     "D"
24: };
25:
26: static void process_list(void)
27: {
28:     struct proc *p;
29:
30:     printk("USER  PID  PPID  S  CMD\n");
31:     FOR_EACH_PROCESS(p) {
32:         if(p->state != PROC_ZOMBIE) {
33:             printk("%d  %5d  %5d  %s  %s\n",
34:                 p->uid,
35:                 p->pid,
36:                 p->ppid,
37:                 pstate[p->state],
38:                 p->argv0
39:             );
40:         }
41:         p = p->next;
42:     }
43:
44:     printk("PIDs in running queue: ");
45:     FOR_EACH_PROCESS_RUNNING(p) {
46:         printk("%d ", p->pid);
47:         p = p->next_run;
48:     }
49:     printk("\n");
50: }
51:
52: void do_sysrq(int op)
53: {
54:     switch(op) {
55:         case SYSRQ_STACK:
56:             printk("sysrq: Stack backtrace.\n");
57:             stack_backtrace();
58:             break;
59:         case SYSRQ_TASKS:
60:             printk("sysrq: Task list.\n");
61:             process_list();
62:             break;
63:         case SYSRQ_UNDEF:
64:             printk("sysrq: Undefined operation.\n");
65:             break;
66:     }
67: }

```

drivers/char/tty.c

Page 1/16

```

1: /*
2:  * fiwix/drivers/char/tty.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/ioctl.h>
10: #include <fiwix/tty.h>
11: #include <fiwix/ctype.h>
12: #include <fiwix/console.h>
13: #include <fiwix/pic.h>
14: #include <fiwix/devices.h>
15: #include <fiwix/fs.h>
16: #include <fiwix/errno.h>
17: #include <fiwix/sched.h>
18: #include <fiwix/timer.h>
19: #include <fiwix/sleep.h>
20: #include <fiwix/process.h>
21: #include <fiwix/fcntl.h>
22: #include <fiwix/kd.h>
23: #include <fiwix/stdio.h>
24: #include <fiwix/string.h>
25:
26: struct tty tty_table[NR_TTYS];
27: extern short int current_cons;
28:
29: static void wait_vtime_off(unsigned int arg)
30: {
31:     unsigned int *fn = (unsigned int *)arg;
32:
33:     wakeup(fn);
34: }
35:
36: static void termios2termio(struct termios *termios, struct termio *termio)
37: {
38:     int n;
39:
40:     termio->c_iflag = termios->c_iflag;
41:     termio->c_oflag = termios->c_oflag;
42:     termio->c_cflag = termios->c_cflag;
43:     termio->c_lflag = termios->c_lflag;
44:     termio->c_line = termios->c_line;
45:     for(n = 0; n < NCC; n++) {
46:         termio->c_cc[n] = termios->c_cc[n];
47:     }
48: }
49:
50: static void termio2termios(struct termio *termio, struct termios *termios)
51: {
52:     int n;
53:
54:     termios->c_iflag = termio->c_iflag;
55:     termios->c_oflag = termio->c_oflag;
56:     termios->c_cflag = termio->c_cflag;
57:     termios->c_lflag = termio->c_lflag;
58:     termios->c_line = termio->c_line;
59:     for(n = 0; n < NCC; n++) {
60:         termios->c_cc[n] = termio->c_cc[n];
61:     }
62: }
63:
64: static int opost(struct tty *tty, unsigned char ch)
65: {
66:     int status;
67:

```

drivers/char/tty.c

Page 2/16

```

68:         status = 0;
69:
70:         if(tty->termios.c_oflag & OPOST) {
71:             switch(ch) {
72:                 case '\n':
73:                     if(tty->termios.c_oflag & ONLCR) {
74:                         if(tty_queue_room(&tty->write_q) >= 2) {
75:                             tty_queue_putchar(tty, &tty->wri
te_q, '\r');
76:                                     tty->column = 0;
77:                                 } else {
78:                                     return -1;
79:                                 }
80:                             }
81:                             break;
82:                 case '\t':
83:                     while(tty->column < (tty->winsize.ws_col - 1)) {
84:                         if(tty->tab_stop[++tty->column]) {
85:                             break;
86:                         }
87:                     }
88:                     break;
89:                 case '\b':
90:                     if(tty->column > 0) {
91:                         tty->column--;
92:                     }
93:                 default:
94:                     if(tty->termios.c_oflag & OLCUC) {
95:                         ch = TOUPPER(ch);
96:                     }
97:                     if(!ISCNTRL(ch)) {
98:                         tty->column++;
99:                     }
100:                    break;
101:            }
102:        }
103:        if(tty_queue_putchar(tty, &tty->write_q, ch) < 0) {
104:            status = -1;
105:        }
106:        return status;
107:    }
108:
109:    static void out_char(struct tty *tty, unsigned char ch)
110:    {
111:        if(ISCNTRL(ch) && !ISSPACE(ch) && (tty->termios.c_lflag & ECHOCTL)) {
112:            if(tty->lnext || (!tty->lnext && ch != tty->termios.c_cc[VEOF]))
113:            {
114:                tty_queue_putchar(tty, &tty->write_q, '^');
115:                tty_queue_putchar(tty, &tty->write_q, ch + 64);
116:                tty->column += 2;
117:            }
118:        } else {
119:            opost(tty, ch);
120:        }
121:    }
122:
123:    static void erase_char(struct tty *tty, unsigned char erasechar)
124:    {
125:        unsigned char ch;
126:
127:        if(erasechar == tty->termios.c_cc[VERASE]) {
128:            if((ch = tty_queue_unputchar(&tty->cooked_q)) &&
129:            if(tty->termios.c_lflag & ECHO) {
130:                tty_queue_putchar(tty, &tty->write_q, '\b');
131:                tty_queue_putchar(tty, &tty->write_q, ' ');
132:                tty_queue_putchar(tty, &tty->write_q, '\b');
133:            }
134:            if(ch == '\t') {

```

drivers/char/tty.c

Page 3/16

```

133:         tty->deltab(tty);
134:     }
135:     if(ISCNTRL(ch) && !ISSPACE(ch) && tty->termios.c
_lflag & ECHOCTL) {
136:         tty_queue_putchar(tty, &tty->write_q, '\
b');
137:         tty_queue_putchar(tty, &tty->write_q, '
');
138:         tty_queue_putchar(tty, &tty->write_q, '\
b');
139:     }
140: }
141: }
142: }
143: if(erasechar == tty->termios.c_cc[VWERASE]) {
144:     unsigned char word_seen = 0;
145:
146:     while(tty->cooked_q.count > 0) {
147:         ch = LAST_CHAR(&tty->cooked_q);
148:         if((ch == ' ' || ch == '\t') && word_seen) {
149:             break;
150:         }
151:         if(ch != ' ' && ch != '\t') {
152:             word_seen = 1;
153:         }
154:         erase_char(tty, tty->termios.c_cc[VERASE]);
155:     }
156: }
157: if(erasechar == tty->termios.c_cc[VKILL]) {
158:     while(tty->cooked_q.count > 0) {
159:         erase_char(tty, tty->termios.c_cc[VERASE]);
160:     }
161:     if(tty->termios.c_lflag & ECHOK && !(tty->termios.c_lflag & ECHO
E)) {
162:         tty_queue_putchar(tty, &tty->write_q, '\n');
163:     }
164: }
165: }
166:
167: static void set_termios(struct tty *tty, struct termios *new_termios)
168: {
169:     memcpy_b(&tty->termios, new_termios, sizeof(struct termios));
170:     if(tty->set_termios) {
171:         tty->set_termios(tty);
172:     }
173: }
174:
175: static void set_termio(struct tty *tty, struct termio *new_termio)
176: {
177:     struct termios new_termios;
178:
179:     termio2termios(new_termio, &new_termios);
180: }
181:
182: int register_tty(__dev_t dev)
183: {
184:     int n;
185:
186:     for(n = 0; n < NR_TTYS; n++) {
187:         if(tty_table[n].dev == dev) {
188:             printk("ERROR: %s(): tty device %d,%d already registered
!\n", __FUNCTION__, MAJOR(dev), MINOR(dev));
189:             return 1;
190:         }
191:         if(!tty_table[n].dev) {
192:             tty_table[n].dev = dev;
193:             tty_table[n].count = 0;

```

drivers/char/tty.c

Page 4/16

```

194:             return 0;
195:         }
196:     }
197:     printk("ERROR: %s(): tty table is full!\n", __FUNCTION__);
198:     return 1;
199: }
200:
201: struct tty * get_tty(__dev_t dev)
202: {
203:     int n;
204:
205:     if(!dev) {
206:         return NULL;
207:     }
208:
209:     /* /dev/console = system console */
210:     if(dev == MKDEV(SYSCON_MAJOR, 1)) {
211:         dev = (__dev_t)_syscondev;
212:     }
213:
214:     /* /dev/tty0 = current virtual console */
215:     if(dev == MKDEV(VCONSOLES_MAJOR, 0)) {
216:         dev = MKDEV(VCONSOLES_MAJOR, current_cons);
217:     }
218:
219:     /* /dev/tty = controlling TTY device */
220:     if(dev == MKDEV(SYSCON_MAJOR, 0)) {
221:         if(!current->ctty) {
222:             return NULL;
223:         }
224:         dev = current->ctty->dev;
225:     }
226:
227:     for(n = 0; n < NR_TTYS; n++) {
228:         if(tty_table[n].dev == dev) {
229:             return &tty_table[n];
230:         }
231:     }
232:     return NULL;
233: }
234:
235: void disassociate_ctty(struct tty *tty)
236: {
237:     struct proc *p;
238:
239:     if(!tty) {
240:         return;
241:     }
242:
243:     /* this tty is no longer the controlling tty of any session */
244:     tty->pgid = tty->sid = 0;
245:
246:     /* clear the controlling tty for all processes in the same SID */
247:     FOR_EACH_PROCESS(p) {
248:         if(p->sid == current->sid) {
249:             p->ctty = NULL;
250:         }
251:         p = p->next;
252:     }
253:     kill_pgrp(current->pgid, SIGHUP);
254:     kill_pgrp(current->pgid, SIGCONT);
255: }
256:
257: void termios_reset(struct tty *tty)
258: {
259:     tty->kbd.mode = K_XLATE;
260:     tty->termios.c_iflag = ICRNL | IXON | IXOFF;

```

drivers/char/tty.c

Page 5/16

```

261:         tty->termios.c_oflag = OPOST | ONLCR;
262:         tty->termios.c_cflag = B9600 | CS8 | HUPCL | CREAD | CLOCAL;
263:         tty->termios.c_lflag = ISIG | ICANON | ECHO | ECHOE | ECHOK | ECHOCTL |
ECHOKE | IEXTEN;
264:         tty->termios.c_line = 0;
265:         tty->termios.c_cc[VINTR] = 3; /* ^C */
266:         tty->termios.c_cc[VQUIT] = 28; /* ^\ */
267:         tty->termios.c_cc[VERASE] = BS; /* ^? (127) not '\b' (^H) */
268:         tty->termios.c_cc[VKILL] = 21; /* ^U */
269:         tty->termios.c_cc[VEOF] = 4; /* ^D */
270:         tty->termios.c_cc[VTIME] = 0;
271:         tty->termios.c_cc[VMIN] = 1;
272:         tty->termios.c_cc[VSWTC] = 0;
273:         tty->termios.c_cc[VSTART] = 17; /* ^Q */
274:         tty->termios.c_cc[VSTOP] = 19; /* ^S */
275:         tty->termios.c_cc[VSUSP] = 26; /* ^Z */
276:         tty->termios.c_cc[VEOL] = '\n'; /* ^J */
277:         tty->termios.c_cc[VREPRINT] = 18; /* ^R */
278:         tty->termios.c_cc[VDISCARD] = 15; /* ^O */
279:         tty->termios.c_cc[VWERASE] = 23; /* ^W */
280:         tty->termios.c_cc[VLNEXT] = 22; /* ^V */
281:         tty->termios.c_cc[VEOL2] = 0;
282:     }
283:
284:     void do_cook(struct tty *tty)
285:     {
286:         int n;
287:         unsigned char ch;
288:         struct cblock *cb;
289:
290:         while(tty->read_q.count > 0) {
291:             ch = tty_queue_getchar(&tty->read_q);
292:
293:             if((tty->termios.c_lflag & ISIG) && !tty->lnext) {
294:                 if(ch == tty->termios.c_cc[VINTR]) {
295:                     if(!(tty->termios.c_lflag & NOFLSH)) {
296:                         tty_queue_flush(&tty->read_q);
297:                         tty_queue_flush(&tty->cooked_q);
298:                     }
299:                     if(tty->pgid > 0) {
300:                         kill_pgrp(tty->pgid, SIGINT);
301:                     }
302:                     break;
303:                 }
304:                 if(ch == tty->termios.c_cc[VQUIT]) {
305:                     if(tty->pgid > 0) {
306:                         kill_pgrp(tty->pgid, SIGQUIT);
307:                     }
308:                     break;
309:                 }
310:                 if(ch == tty->termios.c_cc[VSUSP]) {
311:                     if(tty->pgid > 0) {
312:                         kill_pgrp(tty->pgid, SIGTSTP);
313:                     }
314:                     break;
315:                 }
316:             }
317:
318:             if(tty->termios.c_iflag & ISTRIP) {
319:                 ch = TOASCII(ch);
320:             }
321:             if(tty->termios.c_iflag & IUCLC) {
322:                 if(ISUPPER(ch)) {
323:                     ch = TOLOWER(ch);
324:                 }
325:             }
326:

```

drivers/char/tty.c

Page 6/16

```

327:         if(!tty->lnext) {
328:             if(ch == '\r') {
329:                 if(tty->termios.c_iflag & IGNCR) {
330:                     continue;
331:                 }
332:                 if(tty->termios.c_iflag & ICRNL) {
333:                     ch = '\n';
334:                 }
335:             } else {
336:                 if(ch == '\n') {
337:                     if(tty->termios.c_iflag & INLCR) {
338:                         ch = '\r';
339:                     }
340:                 }
341:             }
342:         }
343:
344:         if(tty->termios.c_lflag & ICANON && !tty->lnext) {
345:             if(ch == tty->termios.c_cc[VERASE] || ch == tty->termios
.c_cc[VWERASE] || ch == tty->termios.c_cc[VKILL]) {
346:                 erase_char(tty, ch);
347:                 continue;
348:             }
349:
350:             if(ch == tty->termios.c_cc[VREPRINT]) {
351:                 out_char(tty, ch);
352:                 tty_queue_putchar(tty, &tty->write_q, '\n');
353:                 cb = tty->cooked_q.head;
354:                 while(cb) {
355:                     for(n = 0; n < cb->end_off; n++) {
356:                         if(n >= cb->start_off) {
357:                             out_char(tty, cb->data[n
]);
358:                         }
359:                     }
360:                     cb = cb->next;
361:                 }
362:                 continue;
363:             }
364:
365:             if(ch == tty->termios.c_cc[VLNEXT] && tty->termios.c_lfl
ag & IEXTEN) {
366:                 tty->lnext = 1;
367:                 if(tty->termios.c_lflag & ECHOCTL) {
368:                     tty_queue_putchar(tty, &tty->write_q, '^
');
369:                     tty_queue_putchar(tty, &tty->write_q, '\
b');
370:                 }
371:                 break;
372:             }
373:
374:             if(tty->termios.c_iflag & IXON) {
375:                 if(ch == tty->termios.c_cc[VSTART]) {
376:                     tty->start(tty);
377:                     continue;
378:                 }
379:                 if(ch == tty->termios.c_cc[VSTOP]) {
380:                     tty->stop(tty);
381:                     continue;
382:                 }
383:                 if(tty->termios.c_iflag & IXANY) {
384:                     tty->start(tty);
385:                 }
386:             }
387:         }
388:

```

drivers/char/tty.c

Page 7/16

```

389:          /* FIXME: using ISSPACE here makes LNEXT working incorrectly */
390:          if(tty->termios.c_lflag & ICANON) {
391:              if(ISCNTRL(ch) && !ISSPACE(ch) && (tty->termios.c_lflag
& ECHOCTL)) {
392:                  out_char(tty, ch);
393:                  tty_queue_putchar(tty, &tty->cooked_q, ch);
394:                  tty->lnext = 0;
395:                  continue;
396:              }
397:              if(ch == '\n') {
398:                  tty->canon_data = 1;
399:              }
400:          }
401:
402:          if(tty->termios.c_lflag & ECHO) {
403:              out_char(tty, ch);
404:          } else {
405:              if((tty->termios.c_lflag & ECHONL) && (ch == '\n')) {
406:                  out_char(tty, ch);
407:              }
408:          }
409:          tty_queue_putchar(tty, &tty->cooked_q, ch);
410:          tty->lnext = 0;
411:      }
412:      tty->output(tty);
413:      if(!(tty->termios.c_lflag & ICANON) || ((tty->termios.c_lflag & ICANON)
&& tty->canon_data)) {
414:          wakeup(&do_select);
415:      }
416:      wakeup(&tty_read);
417: }
418:
419: int tty_open(struct inode *i, struct fd *fd_table)
420: {
421:     int noctty_flag;
422:     struct tty *tty;
423:     int errno;
424:
425:     noctty_flag = fd_table->flags & O_NOCTTY;
426:
427:     if(MAJOR(i->rdev) == SYSCON_MAJOR && MINOR(i->rdev) == 0) {
428:         if(!current->ctty) {
429:             return -ENXIO;
430:         }
431:     }
432:
433:     if(MAJOR(i->rdev) == VCONSOLES_MAJOR && MINOR(i->rdev) == 0) {
434:         noctty_flag = 1;
435:     }
436:
437:     if(!(tty = get_tty(i->rdev))) {
438:         printk("%s(): oops! (%x)\n", __FUNCTION__, i->rdev);
439:         printk("_syscondev = %x\n", _syscondev);
440:         return -ENXIO;
441:     }
442:
443:     if(tty->open) {
444:         if((errno = tty->open(tty)) < 0) {
445:             return errno;
446:         }
447:     }
448:     tty->count++;
449:     tty->column = 0;
450:
451:     if(SESS_LEADER(current) && !current->ctty && !noctty_flag && !tty->sid)
{
452:         current->ctty = tty;

```


drivers/char/tty.c

Page 8/16

```

453:             tty->sid = current->sid;
454:             tty->pgid = current->pgid;
455:         }
456:         return 0;
457: }
458:
459: int tty_close(struct inode *i, struct fd *fd_table)
460: {
461:     struct proc *p;
462:     struct tty *tty;
463:     int errno;
464:
465:     if(!(tty = get_tty(i->rdev))) {
466:         printk("%s(): oops! (%x)\n", __FUNCTION__, i->rdev);
467:         return -ENXIO;
468:     }
469:
470:     if(tty->close) {
471:         if((errno = tty->close(tty)) < 0) {
472:             return errno;
473:         }
474:     }
475:     tty->count--;
476:     if(!tty->count) {
477:         termios_reset(tty);
478:         tty->pgid = tty->sid = 0;
479:
480:         /* this tty is no longer the controlling tty of any process */
481:         FOR_EACH_PROCESS(p) {
482:             if(p->ctty == tty) {
483:                 p->ctty = NULL;
484:             }
485:             p = p->next;
486:         }
487:     }
488:     return 0;
489: }
490:
491: int tty_read(struct inode *i, struct fd *fd_table, char *buffer, __size_t count)
492: {
493:     unsigned int min;
494:     unsigned char ch;
495:     struct tty *tty;
496:     struct callout_req creq;
497:     int n;
498:
499:     if(!(tty = get_tty(i->rdev))) {
500:         printk("%s(): oops! (%x)\n", __FUNCTION__, i->rdev);
501:         return -ENXIO;
502:     }
503:
504:     /* only the foreground process group is allowed to read from the tty */
505:     if(current->ctty == tty && current->pgid != tty->pgid) {
506:         if(current->sigaction[SIGTTIN - 1].sa_handler == SIG_IGN || curr
ent->sigblocked & (1 << (SIGTTIN - 1)) || is_orphaned_pgrp(current->pgid)) {
507:             return -EIO;
508:         }
509:         kill_pgrp(current->pgid, SIGTTIN);
510:         return -ERESTART;
511:     }
512:
513:     n = min = 0;
514:     while(count > 0) {
515:         if(tty->kbd.mode == K_RAW || tty->kbd.mode == K_MEDIUMRAW) {
516:             n = 0;
517:             while(n < count) {
518:                 if((ch = tty_queue_getchar(&tty->read_q)) {

```

drivers/char/tty.c

Page 9/16

```

519:                buffer[n++] = ch;
520:            } else {
521:                break;
522:            }
523:        }
524:        if(n) {
525:            break;
526:        }
527:    }
528:
529:    if(tty->termios.c_lflag & ICANON) {
530:        if((ch = LAST_CHAR(&tty->cooked_q)) {
531:            if(ch == '\n' || ch == tty->termios.c_cc[VEOL] |
| ch == tty->termios.c_cc[VEOF] || (tty->termios.c_lflag & IEXTEN && ch == tty->termios
.c_cc[VEOL2] && tty->termios.c_cc[VEOL2] != 0)) {
532:
533:                tty->canon_data = 0;
534:                /* EOF is not passed to the reading proc
ess */
535:                if(ch == tty->termios.c_cc[VEOF]) {
536:                    tty_queue_unputchar(&tty->cooked
_q);
537:                }
538:
539:                while(n < count) {
540:                    if((ch = tty_queue_getchar(&tty-
>cooked_q)) {
541:                        buffer[n++] = ch;
542:                    } else {
543:                        break;
544:                    }
545:                }
546:                break;
547:            }
548:        }
549:    } else {
550:        if(tty->termios.c_cc[VTIME] > 0) {
551:            unsigned int ini_ticks = kstat.ticks;
552:            unsigned int timeout;
553:
554:            if(!tty->termios.c_cc[VMIN]) {
555:                /* VTIME is measured in tenths of second
*/
556:                timeout = tty->termios.c_cc[VTIME] * (HZ
/ 10);
557:
558:                while(kstat.ticks - ini_ticks < timeout
&& !tty->cooked_q.count) {
559:                    creq.fn = wait_vtime_off;
560:                    creq.arg = (unsigned int)&tty->c
ooked_q;
561:                    add_callout(&creq, timeout);
562:                    if(fd_table->flags & O_NONBLOCK)
{
563:                        return -EAGAIN;
564:                    }
565:                    if(sleep(&tty_read, PROC_INTERRU
PTIBLE)) {
566:                        return -EINTR;
567:                    }
568:                }
569:                while(n < count) {
570:                    if((ch = tty_queue_getchar(&tty-
>cooked_q))) {
571:                        buffer[n++] = ch;
572:                    } else {
573:                        break;

```

drivers/char/tty.c

Page 10/16

```

574:                                     }
575:                                     }
576:                                     break;
577:         } else {
578:             if(tty->cooked_q.count > 0) {
579:                 if(n < MIN(tty->termios.c_cc[VMI
N], count)) {
580:                     ch = tty_queue_getchar(&
tty->cooked_q);
581:                     buffer[n++] = ch;
582:                 }
583:                 if(n >= MIN(tty->termios.c_cc[VM
IN], count)) {
584:                     del_callout(&creq);
585:                     break;
586:                 }
587:                 timeout = tty->termios.c_cc[VTIM
E] * (HZ / 10);
588:                 creq.fn = wait_vtime_off;
589:                 creq.arg = (unsigned int)&tty->c
ooked_q;
590:                 add_callout(&creq, timeout);
591:                 if(fd_table->flags & O_NONBLOCK)
{
592:                     n = -EAGAIN;
593:                     break;
594:                 }
595:                 if(sleep(&tty_read, PROC_INTERRUPTI
BLE)) {
596:                     n = -EINTR;
597:                     break;
598:                 }
599:                 if(!tty->cooked_q.count) {
600:                     break;
601:                 }
602:                 continue;
603:             }
604:         }
605:     } else {
606:         if(tty->cooked_q.count > 0) {
607:             if(min < tty->termios.c_cc[VMIN] || !tty
->termios.c_cc[VMIN]) {
608:                 if(n < count) {
609:                     ch = tty_queue_getchar(&
tty->cooked_q);
610:                     buffer[n++] = ch;
611:                     if(--tty->canon_data < 0
) {
612:                         tty->canon_data
= 0;
613:                     }
614:                 }
615:                 min++;
616:             }
617:         }
618:         if(min >= tty->termios.c_cc[VMIN]) {
619:             break;
620:         }
621:     }
622: }
623: if(fd_table->flags & O_NONBLOCK) {
624:     n = -EAGAIN;
625:     break;
626: }
627: if(sleep(&tty_read, PROC_INTERRUPTIBLE)) {
628:     n = -EINTR;
629:     break;

```

drivers/char/tty.c

Page 11/16

```

630:         }
631:     }
632:
633:     if(n) {
634:         i->i_atime = CURRENT_TIME;
635:     }
636:     return n;
637: }
638:
639: int tty_write(struct inode *i, struct fd *fd_table, const char *buffer, __size_t
count)
640: {
641:     unsigned char ch;
642:     struct tty *tty;
643:     int n;
644:
645:     if(!(tty = get_tty(i->rdev))) {
646:         printk("%s(): oops! (%x)\n", __FUNCTION__, i->rdev);
647:         return -ENXIO;
648:     }
649:
650:     /* only the foreground process group is allowed to write to the tty */
651:     if(current->ctty == tty && current->pgid != tty->pgid) {
652:         if(tty->termios.c_lflag & TOSTOP) {
653:             if(current->sigaction[SIGTTIN - 1].sa_handler != SIG_IGN
&& !(current->sigblocked & (1 << (SIGTTIN - 1)))) {
654:                 if(is_orphaned_pgrp(current->pgid)) {
655:                     return -EIO;
656:                 }
657:                 kill_pgrp(current->pgid, SIGTTOU);
658:                 return -ERESTART;
659:             }
660:         }
661:     }
662:
663:     n = 0;
664:     for(;;) {
665:         if(current->sigpending & ~current->sigblocked) {
666:             return -ERESTART;
667:         }
668:         while(count && n < count) {
669:             ch = *(buffer + n);
670:             /* FIXME: check if *(buffer + n) address is valid */
671:             if(opost(tty, ch) < 0) {
672:                 break;
673:             }
674:             n++;
675:         }
676:         tty->output(tty);
677:         if(n == count) {
678:             break;
679:         }
680:         if(fd_table->flags & O_NONBLOCK) {
681:             n = -EAGAIN;
682:             break;
683:         }
684:         if(tty->write_q.count > 0) {
685:             if(sleep(&tty_write, PROC_INTERRUPTIBLE)) {
686:                 n = -EINTR;
687:                 break;
688:             }
689:         }
690:         if(need_resched) {
691:             do_sched();
692:         }
693:     }
694:

```

drivers/char/tty.c

Page 12/16

```

695:         if(n) {
696:             i->i_mtime = CURRENT_TIME;
697:         }
698:         return n;
699:     }
700:
701:     /* FIXME: http://www.lafn.org/~dave/linux/termios.txt (doc/termios.txt) */
702:     int tty_ioctl(struct inode *i, int cmd, unsigned long int arg)
703:     {
704:         struct proc *p;
705:         struct tty *tty;
706:         int errno;
707:
708:         if(!(tty = get_tty(i->rdev))) {
709:             printk("%s(): oops! (%x)\n", __FUNCTION__, i->rdev);
710:             return -ENXIO;
711:         }
712:
713:         switch(cmd) {
714:             /*
715:              * Fetch and store the current terminal parameters to a termios
716:              * structure pointed to by the argument.
717:              */
718:             case TCGETS:
719:                 if((errno = check_user_area(VERIFY_WRITE, (void *)arg, s
sizeof(struct termios)))) {
720:                     return errno;
721:                 }
722:                 memcpy_b((struct termios *)arg, &tty->termios, sizeof(st
ruct termios));
723:                 break;
724:
725:             /*
726:              * Set the current terminal parameters according to the
727:              * values in the termios structure pointed to by the argument.
728:              */
729:             case TCSETS:
730:                 if((errno = check_user_area(VERIFY_READ, (void *)arg, si
sizeof(struct termios)))) {
731:                     return errno;
732:                 }
733:                 set_termios(tty, (struct termios *)arg);
734:                 break;
735:
736:             /*
737:              * Same as TCSETS except it doesn't take effect until all
738:              * the characters queued for output have been transmitted.
739:              */
740:             case TCSETSW:
741:                 if((errno = check_user_area(VERIFY_READ, (void *)arg, si
sizeof(struct termios)))) {
742:                     return errno;
743:                 }
744:                 set_termios(tty, (struct termios *)arg);
745:                 break;
746:
747:             /*
748:              * Same as TCSETSW except that all characters queued for
749:              * input are discarded.
750:              */
751:             case TCSETSF:
752:                 if((errno = check_user_area(VERIFY_READ, (void *)arg, si
sizeof(struct termios)))) {
753:                     return errno;
754:                 }
755:                 set_termios(tty, (struct termios *)arg);
756:                 tty_queue_flush(&tty->read_q);

```

drivers/char/tty.c

Page 13/16

```

757:                 break;
758:
759:                 /*
760:                 * Fetch and store the current terminal parameters to a termio
761:                 * structure pointed to by the argument.
762:                 */
763:                 case TCGETA:
764:                 if((errno = check_user_area(VERIFY_WRITE, (void *)arg, s
sizeof(struct termio)))) {
765:                 return errno;
766:                 }
767:                 termios2termio(&tty->termios, (struct termio *)arg);
768:                 break;
769:
770:                 /*
771:                 * Set the current terminal parameters according to the
772:                 * values in the termio structure pointed to by the argument.
773:                 */
774:                 case TCSETA:
775:                 if((errno = check_user_area(VERIFY_READ, (void *)arg, si
sizeof(struct termio)))) {
776:                 return errno;
777:                 }
778:                 set_termio(tty, (struct termio *)arg);
779:                 break;
780:
781:                 /*
782:                 * Same as TCSET except it doesn't take effect until all
783:                 * the characters queued for output have been transmitted.
784:                 */
785:                 case TCSETAW:
786:                 if((errno = check_user_area(VERIFY_READ, (void *)arg, si
sizeof(struct termio)))) {
787:                 return errno;
788:                 }
789:                 set_termio(tty, (struct termio *)arg);
790:                 break;
791:
792:                 /*
793:                 * Same as TCSETAW except that all characters queued for
794:                 * input are discarded.
795:                 */
796:                 case TCSETAF:
797:                 if((errno = check_user_area(VERIFY_READ, (void *)arg, si
sizeof(struct termio)))) {
798:                 return errno;
799:                 }
800:                 set_termio(tty, (struct termio *)arg);
801:                 tty_queue_flush(&tty->read_q);
802:                 break;
803:
804:                 /* Perform start/stop control */
805:                 case TCXONC:
806:                 switch(arg) {
807:                 case TCOFF:
808:                 tty->stop(tty);
809:                 break;
810:                 case TCOON:
811:                 tty->start(tty);
812:                 break;
813:                 default:
814:                 return -EINVAL;
815:                 }
816:                 break;
817:                 case TCFLSH:
818:                 switch(arg) {
819:                 case TCIFLUSH:

```

drivers/char/tty.c

Page 14/16

```

820:                 tty_queue_flush(&tty->read_q);
821:                 tty_queue_flush(&tty->cooked_q);
822:                 break;
823:             case TCOFLUSH:
824:                 tty_queue_flush(&tty->write_q);
825:                 break;
826:             case TCIOFLUSH:
827:                 tty_queue_flush(&tty->read_q);
828:                 tty_queue_flush(&tty->cooked_q);
829:                 tty_queue_flush(&tty->write_q);
830:                 break;
831:             default:
832:                 return -EINVAL;
833:         }
834:         break;
835:     case TIOCSCTTY:
836:         if(SESS_LEADER(current) && (current->sid == tty->sid)) {
837:             return 0;
838:         }
839:         if(!SESS_LEADER(current) || current->ctty) {
840:             return -EPERM;
841:         }
842:         if(tty->sid) {
843:             if((arg == 1) && IS_SUPERUSER) {
844:                 FOR_EACH_PROCESS(p) {
845:                     if(p->ctty == tty) {
846:                         p->ctty = NULL;
847:                     }
848:                     p = p->next;
849:                 }
850:             } else {
851:                 return -EPERM;
852:             }
853:         }
854:         current->ctty = tty;
855:         tty->sid = current->sid;
856:         tty->pgid = current->pgid;
857:         break;
858:
859:         /*
860:          * Get the process group ID of the '__pid_t' pointed to by
861:          * the arg to the foreground processes group ID.
862:          */
863:     case TIOCGPRG:
864:         if((errno = check_user_area(VERIFY_WRITE, (void *)arg, s
sizeof(__pid_t)))) {
865:             return errno;
866:         }
867:         memcpy_b((void *)arg, &tty->pgid, sizeof(__pid_t));
868:         break;
869:
870:         /*
871:          * Associate the process pointed to by '__pid_t' in the arg to
872:          * the value of the terminal.
873:          */
874:     case TIOCSPGRP:
875:         if(arg < 1) {
876:             return -EINVAL;
877:         }
878:         if((errno = check_user_area(VERIFY_READ, (void *)arg, si
sizeof(__pid_t)))) {
879:             return errno;
880:         }
881:         memcpy_b(&tty->pgid, (void *)arg, sizeof(__pid_t));
882:         break;
883:
884:         /*

```

drivers/char/tty.c

Page 15/16

```

885:         * The session ID of the terminal is fetched and stored in
886:         * the '__pid_t' pointed to by the arg.
887:     case TIOCSID:    FIXME
888:         */
889:
890:     /*
891:     * The terminal drivers notion of terminal size is stored in
892:     * the 'winsize' structure pointed to by the arg.
893:     */
894:     case TIOCGWINSZ:
895:         if((errno = check_user_area(VERIFY_WRITE, (void *)arg, s
sizeof(struct winsize)))) {
896:             return errno;
897:         }
898:         memcpy_b((void *)arg, &tty->winsize, sizeof(struct winsi
ze));
899:         break;
900:
901:     /*
902:     * The terminal drivers notion of the terminal size is set
903:     * to value in the 'winsize' structure pointed to by the arg.
904:     */
905:     case TIOCSWINSZ:
906:     {
907:         struct winsize *ws = (struct winsize *)arg;
908:         short int changed;
909:
910:         if((errno = check_user_area(VERIFY_READ, (void *)arg, si
zeof(struct winsize)))) {
911:             return errno;
912:         }
913:         changed = 0;
914:         if(tty->winsize.ws_row != ws->ws_row) {
915:             changed = 1;
916:         }
917:         if(tty->winsize.ws_col != ws->ws_col) {
918:             changed = 1;
919:         }
920:         if(tty->winsize.ws_xpixel != ws->ws_xpixel) {
921:             changed = 1;
922:         }
923:         if(tty->winsize.ws_ypixel != ws->ws_ypixel) {
924:             changed = 1;
925:         }
926:         tty->winsize.ws_row = ws->ws_row;
927:         tty->winsize.ws_col = ws->ws_col;
928:         tty->winsize.ws_xpixel = ws->ws_xpixel;
929:         tty->winsize.ws_ypixel = ws->ws_ypixel;
930:         if(changed) {
931:             kill_pgrp(tty->pgid, SIGWINCH);
932:         }
933:     }
934:     break;
935:     case TIOCNOTTY:
936:         if(current->ctty != tty) {
937:             return -ENOTTY;
938:         }
939:         if(SESS_LEADER(current)) {
940:             disassociate_ctty(tty);
941:         }
942:         break;
943:     case TIOCLINUX:
944:     {
945:         int val = *(unsigned char *)arg;
946:         if((errno = check_user_area(VERIFY_READ, (void *)arg, si
zeof(unsigned char)))) {
947:             return errno;

```


drivers/char/tty.c

Page 16/16

```

948:         }
949:         switch(val) {
950:             case 12:         /* get current console */
951:                 return current_cons;
952:                 break;
953:             default:
954:                 return -EINVAL;
955:                 break;
956:         }
957:         break;
958:     }
959:
960:     default:
961:         return vt_ioctl(tty, cmd, arg);
962:     }
963:     return 0;
964: }
965:
966: int tty_lseek(struct inode *i, __off_t offset)
967: {
968:     return -ESPIPE;
969: }
970:
971: int tty_select(struct inode *i, int flag)
972: {
973:     struct tty *tty;
974:
975:     if(!(tty = get_tty(i->rdev))) {
976:         printk("%s(): oops! (%x)\n", __FUNCTION__, i->rdev);
977:         return 0;
978:     }
979:
980:     switch(flag) {
981:         case SEL_R:
982:             if(tty->cooked_q.count > 0) {
983:                 if(!(tty->termios.c_lflag & ICANON) || ((tty->te
rmios.c_lflag & ICANON) && tty->canon_data)) {
984:                     return 1;
985:                 }
986:             }
987:             break;
988:         case SEL_W:
989:             if(!tty->write_q.count) {
990:                 return 1;
991:             }
992:             break;
993:     }
994:     return 0;
995: }
996:
997: void tty_init(void)
998: {
999:     tty_queue_init();
1000:     memset_b(tty_table, NULL, sizeof(tty_table));
1001: }

```

drivers/char/tty_queue.c

Page 1/4

```

1: /*
2:  * fiwix/drivers/char/tty_queue.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/kernel.h>
10: #include <fiwix/tty.h>
11: #include <fiwix/errno.h>
12: #include <fiwix/string.h>
13:
14: /*
15:  * tty_queue.c implements a queue using a static-sized doubly linked list of a
16:  * central pool of buffers which covers all ttys.
17:  *
18:  * head                                     tail
19:  * +-----+ +-----+ ... +-----+
20:  * |prev|data|next| |prev|data|next| ... |prev|data|next|
21:  * | / | | --> <-- | | --> ... <-- | | / |
22:  * +-----+ +-----+ ... +-----+
23:  * (cblock)      (cblock)      (cblock)
24:  */
25:
26: struct cblock cblock_pool[CB_POOL_SIZE];
27: struct cblock *cblock_pool_head;
28:
29: static struct cblock *get_free_cblock(void)
30: {
31:     struct cblock *new = NULL;
32:
33:     if(cblock_pool_head) {
34:         new = cblock_pool_head;
35:         cblock_pool_head = cblock_pool_head->next;
36:         new->prev = new->next = NULL;
37:     }
38:     return new;
39: }
40:
41: static void put_free_cblock(struct cblock *old)
42: {
43:     old->prev = NULL;
44:     old->next = cblock_pool_head;
45:     cblock_pool_head = old;
46: }
47:
48: static struct cblock *insert_cblock_in_head(struct clist *q)
49: {
50:     struct cblock *cb;
51:
52:     if(q->cb_num >= NR_CB_QUEUE) {
53:         return NULL;
54:     }
55:     if(!(cb = get_free_cblock())) {
56:         return NULL;
57:     }
58:
59:     /* initialize cblock */
60:     cb->start_off = cb->end_off = 0;
61:     memset_b(cb->data, 0, CBSIZE);
62:     cb->prev = cb->next = NULL;
63:     q->cb_num++;
64:
65:     if(!q->head) {
66:         q->head = q->tail = cb;
67:     } else {

```

drivers/char/tty_queue.c

Page 2/4

```

68:         cb->prev = NULL;
69:         cb->next = q->head;
70:         q->head->prev = cb;
71:         q->head = cb;
72:     }
73:     return cb;
74: }
75:
76: static struct cblock *insert_cblock_in_tail(struct clist *q)
77: {
78:     struct cblock *cb;
79:
80:     if(q->cb_num >= NR_CB_QUEUE) {
81:         return NULL;
82:     }
83:     if(!(cb = get_free_cblock())) {
84:         return NULL;
85:     }
86:
87:     /* initialize cblock */
88:     cb->start_off = cb->end_off = 0;
89:     memset_b(cb->data, 0, CBSIZE);
90:     cb->prev = cb->next = NULL;
91:     q->cb_num++;
92:
93:     if(!q->tail) {
94:         q->head = q->tail = cb;
95:     } else {
96:         cb->prev = q->tail;
97:         cb->next = NULL;
98:         q->tail->next = cb;
99:         q->tail = cb;
100:    }
101:    return cb;
102: }
103:
104: static void delete_cblock_from_head(struct clist *q)
105: {
106:     struct cblock *tmp;
107:
108:     if(!q->head) {
109:         return;
110:     }
111:
112:     tmp = q->head;
113:     if(q->head == q->tail) {
114:         q->head = q->tail = NULL;
115:     } else {
116:         q->head = q->head->next;
117:         q->head->prev = NULL;
118:     }
119:
120:     q->count -= tmp->end_off - tmp->start_off;
121:     q->cb_num--;
122:     put_free_cblock(tmp);
123: }
124:
125: static void delete_cblock_from_tail(struct clist *q)
126: {
127:     struct cblock *tmp;
128:
129:     if(!q->tail) {
130:         return;
131:     }
132:
133:     tmp = q->tail;
134:     if(q->head == q->tail) {

```

drivers/char/tty_queue.c

Page 3/4

```

135:         q->head = q->tail = NULL;
136:     } else {
137:         q->tail = q->tail->prev;
138:         q->tail->next = NULL;
139:     }
140:
141:     q->count -= tmp->end_off - tmp->start_off;
142:     q->cb_num--;
143:     put_free_cblock(tmp);
144: }
145:
146: int tty_queue_putchar(struct tty *tty, struct clist *q, unsigned char ch)
147: {
148:     unsigned long int flags;
149:     struct cblock *cb;
150:     int errno;
151:
152:     SAVE_FLAGS(flags); CLI();
153:
154:     cb = q->tail;
155:     if(!cb) {
156:         cb = insert_cblock_in_tail(q);
157:         if(!cb) {
158:             RESTORE_FLAGS(flags);
159:             return -EAGAIN;
160:         }
161:     }
162:
163:     if(cb->end_off < CBSIZE) {
164:         cb->data[cb->end_off] = ch;
165:         cb->end_off++;
166:         q->count++;
167:         errno = 0;
168:     } else if(insert_cblock_in_tail(q)) {
169:         tty_queue_putchar(tty, q, ch);
170:         errno = 0;
171:     } else {
172:         errno = -EAGAIN;
173:     }
174:
175:     RESTORE_FLAGS(flags);
176:     return errno;
177: }
178:
179: int tty_queue_unputchar(struct clist *q)
180: {
181:     unsigned long int flags;
182:     struct cblock *cb;
183:     unsigned char ch;
184:
185:     SAVE_FLAGS(flags); CLI();
186:
187:     ch = 0;
188:     cb = q->tail;
189:     if(cb) {
190:         if(cb->end_off > cb->start_off) {
191:             ch = cb->data[cb->end_off - 1];
192:             cb->end_off--;
193:             q->count--;
194:         }
195:         if(cb->end_off - cb->start_off == 0) {
196:             delete_cblock_from_tail(q);
197:         }
198:     }
199:
200:     RESTORE_FLAGS(flags);
201:     return ch;

```

drivers/char/tty_queue.c

Page 4/4

```
202: }
203:
204: unsigned char tty_queue_getchar(struct clist *q)
205: {
206:     unsigned long int flags;
207:     struct cblock *cb;
208:     unsigned char ch;
209:
210:     SAVE_FLAGS(flags); CLI();
211:
212:     ch = 0;
213:     cb = q->head;
214:     if(cb) {
215:         if(cb->start_off < cb->end_off) {
216:             ch = cb->data[cb->start_off];
217:             cb->start_off++;
218:             q->count--;
219:         }
220:         if(cb->end_off - cb->start_off == 0) {
221:             delete_cblock_from_head(q);
222:         }
223:     }
224:
225:     RESTORE_FLAGS(flags);
226:     return ch;
227: }
228:
229: void tty_queue_flush(struct clist *q)
230: {
231:     unsigned long int flags;
232:
233:     SAVE_FLAGS(flags); CLI();
234:
235:     while(q->head != NULL) {
236:         delete_cblock_from_head(q);
237:     }
238:
239:     RESTORE_FLAGS(flags);
240: }
241:
242: int tty_queue_room(struct clist *q)
243: {
244:     return (NR_CB_QUEUE * CBSIZE) - q->count;
245: }
246:
247: void tty_queue_init(void)
248: {
249:     int n;
250:     struct cblock *cb;
251:
252:     memset_b(cblock_pool, NULL, sizeof(cblock_pool));
253:
254:     /* cblock free list initialization */
255:     cblock_pool_head = NULL;
256:     n = CB_POOL_SIZE;
257:     while(n-->0) {
258:         cb = &cblock_pool[n];
259:         put_free_cblock(cb);
260:     }
261: }
```

drivers/char/vt.c

Page 1/4

```

1: /*
2:  * fiwix/drivers/char/vt.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/console.h>
10: #include <fiwix/keyboard.h>
11: #include <fiwix/tty.h>
12: #include <fiwix/vt.h>
13: #include <fiwix/kd.h>
14: #include <fiwix/errno.h>
15: #include <fiwix/stdio.h>
16: #include <fiwix/string.h>
17:
18: int kbdmode = 0;
19:
20: int vt_ioctl(struct tty *tty, int cmd, unsigned long int arg)
21: {
22:     struct vconsole *vc;
23:     int n, errno;
24:
25:     /* only virtual consoles support the following ioctl commands */
26:     if(MAJOR(tty->dev) != VCONSOLES_MAJOR) {
27:         return -ENXIO;
28:     }
29:
30:     vc = (struct vconsole *)tty->driver_data;
31:
32:     switch(cmd) {
33:         case KDGETLED:
34:             if((errno = check_user_area(VERIFY_WRITE, (void *)arg, s
sizeof(unsigned char)))) {
35:                 return errno;
36:             }
37:             memset_b((void *)arg, vc->led_status, sizeof(char));
38:             break;
39:
40:         case KDSETLED:
41:             if(arg > 7) {
42:                 return -EINVAL;
43:             }
44:             vc->led_status = arg;
45:             set_leds(vc->led_status);
46:             break;
47:
48:             /* FIXME: implement KDGKBLED and KDSKBLED
49:             * it will need to convert 'scrlock, numlock, capslock' into led
50:             _flags.
51:             */
52:
53:         case KDGKBTYPE:
54:             if((errno = check_user_area(VERIFY_WRITE, (void *)arg, s
sizeof(unsigned char)))) {
55:                 return errno;
56:             }
57:             memset_b((void *)arg, KB_101, sizeof(char));
58:             break;
59:
60:         case KDSETMODE:
61:             if(arg != KD_TEXT && arg != KD_GRAPHICS) {
62:                 return -EINVAL;
63:             }
64:             if(vc->vc_mode != arg) {
65:                 vc->vc_mode = arg;

```

drivers/char/vt.c

Page 2/4

```

65:                                if(arg == KD_GRAPHICS) {
66:                                    video.blank_screen(vc);
67:                                } else {
68:                                    unblank_screen(vc);
69:                                }
70:                                }
71:                                break;
72:
73:                                case KDGETMODE:
74:                                    if((errno = check_user_area(VERIFY_WRITE, (void *)arg, s
izeof(unsigned char)))) {
75:                                        return errno;
76:                                    }
77:                                    memset_b((void *)arg, vc->vc_mode, sizeof(char));
78:                                    break;
79:
80:                                case KDGKBMODE:
81:                                    if((errno = check_user_area(VERIFY_WRITE, (void *)arg, s
izeof(unsigned char)))) {
82:                                        return errno;
83:                                    }
84:                                    memset_b((void *)arg, tty->kbd.mode, sizeof(unsigned cha
r));
85:                                    break;
86:
87:                                case KDSKBMODE:
88:                                    if(arg != K_RAW && arg != K_XLATE && arg != K_MEDIUMRAW)
{
89:                                        arg = K_XLATE;
90:                                    }
91:                                    tty->kbd.mode = arg;
92:                                    tty_queue_flush(&tty->read_q);
93:                                    break;
94:
95:                                case KDSKBENT:
96:                                    {
97:                                        struct kbentry *k = (struct kbentry *)arg;
98:                                        if((errno = check_user_area(VERIFY_WRITE, (void *)k, siz
eof(struct kbentry)))) {
99:                                            return errno;
100:                                        }
101:                                        if(k->kb_table < NR_MODIFIERS) {
102:                                            if(k->kb_index < NR_SCODES) {
103:                                                keymap[(k->kb_index * NR_MODIFIERS) + k-
>kb_table] = k->kb_value;
104:                                            } else {
105:                                                return -EINVAL;
106:                                            }
107:                                        } else {
108:                                            printk("%s(): kb_table value '%d' not supported.
\n", __FUNCTION__, k->kb_table);
109:                                            return -EINVAL;
110:                                        }
111:                                    }
112:                                    break;
113:
114:                                case VT_OPENQRY:
115:                                    {
116:                                        int *val = (int *)arg;
117:                                        if((errno = check_user_area(VERIFY_WRITE, (void *)arg, s
izeof(unsigned int)))) {
118:                                            return errno;
119:                                        }
120:                                        for(n = 1; n < NR_VCONSOLES + 1; n++) {
121:                                            tty = get_tty(MKDEV(VCONSOLES_MAJOR, n));
122:                                            if(!tty->count) {
123:                                                break;

```

drivers/char/vt.c

Page 3/4

```

124:                }
125:            }
126:            *val = (n < NR_VCONSOLES + 1 ? n : -1);
127:        }
128:        break;
129:
130:        case VT_GETMODE:
131:        {
132:            struct vt_mode *vt_mode = (struct vt_mode *)arg;
133:            if((errno = check_user_area(VERIFY_WRITE, (void *)vt_mode
e, sizeof(struct vt_mode)))) {
134:                return errno;
135:            }
136:            memcpy_b(vt_mode, &vc->vt_mode, sizeof(struct vt_mode));
137:        }
138:        break;
139:
140:        case VT_SETMODE:
141:        {
142:            struct vt_mode *vt_mode = (struct vt_mode *)arg;
143:            if((errno = check_user_area(VERIFY_READ, (void *)vt_mode
, sizeof(struct vt_mode)))) {
144:                return errno;
145:            }
146:            if(vt_mode->mode != VT_AUTO && vt_mode->mode != VT_PROCE
SS) {
147:                return -EINVAL;
148:            }
149:            memcpy_b(&vc->vt_mode, vt_mode, sizeof(struct vt_mode));
150:            vc->vt_mode.frsig = 0; /* ignored */
151:            tty->pid = current->pid;
152:            vc->switchto_tty = 0;
153:        }
154:        break;
155:
156:        case VT_GETSTATE:
157:        {
158:            struct vt_stat *vt_stat = (struct vt_stat *)arg;
159:            if((errno = check_user_area(VERIFY_WRITE, (void *)vt_sta
t, sizeof(struct vt_stat)))) {
160:                return errno;
161:            }
162:            vt_stat->v_active = current_cons;
163:            vt_stat->v_state = 1; /* /dev/tty0 is always opened */
164:            for(n = 1; n < NR_VCONSOLES + 1; n++) {
165:                tty = get_tty(MKDEV(VCONSOLES_MAJOR, n));
166:                if(tty->count) {
167:                    vt_stat->v_state |= (1 << n);
168:                }
169:            }
170:        }
171:        break;
172:
173:        case VT_RELDISP:
174:            if(vc->vt_mode.mode != VT_PROCESS) {
175:                return -EINVAL;
176:            }
177:            if(vc->switchto_tty < 0) {
178:                if(arg != VT_ACKACQ) {
179:                    return -EINVAL;
180:                }
181:            } else {
182:                if(arg) {
183:                    int switchto_tty;
184:                    switchto_tty = vc->switchto_tty;
185:                    vc->switchto_tty = -1;
186:                    vconsole_select_final(switchto_tty);

```


drivers/char/vt.c

Page 4/4

```
187:                } else {
188:                    vc->switchto_tty = -1;
189:                }
190:            }
191:            break;
192:
193:        case VT_ACTIVATE:
194:            if(current_cons == MINOR(tty->dev) || IS_SUPERUSER) {
195:                if(!arg || arg > NR_VCONSOLES) {
196:                    return -ENXIO;
197:                }
198:                vconsole_select(--arg);
199:            } else {
200:                return -EPERM;
201:            }
202:            break;
203:
204:        case VT_WAITACTIVE:
205:            if(current_cons == MINOR(tty->dev)) {
206:                break;
207:            }
208:            if(!arg || arg > NR_VCONSOLES) {
209:                return -ENXIO;
210:            }
211:            printk("ACTIVATING another tty!! (cmd = 0x%x)\n", cmd);
212:            break;
213:
214:        default:
215:            return -EINVAL;
216:    }
217:    return 0;
218: }
```

drivers/video/fbcon.c

Page 1/10

```

1: /*
2:  * fiwix/drivers/char/fbcon.c
3:  *
4:  * Copyright 2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/fb.h>
10: #include <fiwix/fbcon.h>
11: #include <fiwix/font.h>
12: #include <fiwix/console.h>
13: #include <fiwix/tty.h>
14: #include <fiwix/timer.h>
15: #include <fiwix/stdio.h>
16: #include <fiwix/string.h>
17:
18: #define SPACE_CHAR      32
19:
20: unsigned char *font_data;
21: unsigned char *cursor_shape;
22: struct video_parms video;
23: static unsigned char screen_is_off = 0;
24:
25: /* RGB colors */
26: static int color_table[] = {
27:     0x000000,    /* black */
28:     0x0000AA,    /* blue */
29:     0x00AA00,    /* green */
30:     0x00AAAA,    /* cyan */
31:     0xAA0000,    /* red */
32:     0xAA00AA,    /* magenta */
33:     0xAA5000,    /* brown */
34:     0xAAAAAA,    /* gray */
35:
36:     0x555555,    /* dark gray */
37:     0x5555FF,    /* light blue */
38:     0x55FF55,    /* light green */
39:     0x55FFFF,    /* light cyan */
40:     0xFF5555,    /* light red */
41:     0xFF55FF,    /* light magenta */
42:     0xFFFF55,    /* yellow */
43:     0FFFFFFF,    /* white */
44: };
45:
46: static int get_fg_color(unsigned char color)
47: {
48:     int fg, bright;
49:
50:     fg = color & 7;
51:     bright = (color & 0xF) & 8;
52:     return color_table[bright + fg];
53: }
54:
55: static int get_bg_color(unsigned char color)
56: {
57:     int bg;
58:
59:     bg = (color >> 4) & 7;
60:     return color_table[bg];
61: }
62:
63: static void set_color(void *addr, int color)
64: {
65:     unsigned int *addr32;
66:     unsigned short int *addr16;
67:     unsigned char *addr8;

```

drivers/video/fbcon.c

Page 2/10

```

68:         short int r, g, b;
69:
70:         switch(video.fb_bpp) {
71:             case 32:
72:                 addr32 = (unsigned int *)addr;
73:                 *addr32 = color;
74:                 break;
75:             case 24:
76:                 addr8 = (unsigned char *)addr;
77:                 *(addr8++) = color & 0xFF;
78:                 *(addr8++) = (color >> 8) & 0xFF;
79:                 *(addr8++) = (color >> 16) & 0xFF;
80:                 break;
81:             case 16:
82:                 /* 0:5:6:5 */
83:                 r = ((color >> 16) & 0xFF) << 8;
84:                 g = ((color >> 8) & 0xFF) << 8;
85:                 b = (color & 0xFF) << 8;
86:                 addr16 = (unsigned short int *)addr;
87:                 *addr16 = (r & 0xf800) | ((g & 0xfc00) >> 5) | ((b & 0xf
800) >> 11);
88:             case 15:
89:                 /* 1:5:5:5 */
90:                 r = ((color >> 16) & 0xFF) << 8;
91:                 g = ((color >> 8) & 0xFF) << 8;
92:                 b = (color & 0xFF) << 8;
93:                 addr16 = (unsigned short int *)addr;
94:                 *addr16 = ((r & 0xf800) >> 1) | ((g & 0xf800) >> 6) | ((
b & 0xf800) >> 11);
95:                 break;
96:         }
97:     }
98:
99: static void draw_glyph(unsigned char *addr, int x, int y, unsigned char *ch, int
color)
100: {
101:     int n, b, offset;
102:
103:     if(screen_is_off) {
104:         return;
105:     }
106:
107:     offset = (y * video.fb_linesize) + (x * video.fb_bpp);
108:     addr += offset;
109:
110:     for(n = 0; n < video.fb_char_height; n++) {
111:         if(*ch + n == 0) {
112:             if(ch == cursor_shape) {
113:                 addr += video.fb_pitch;
114:                 continue;
115:             }
116:             b = video.fb_char_width - 1;
117:             do {
118:                 set_color(addr, get_bg_color(color));
119:                 addr += video.fb_pixelwidth;
120:                 b--;
121:             } while(b >= 0);
122:         } else {
123:             b = video.fb_char_width - 1;
124:             do {
125:                 if(*ch + n & (1 << b)) {
126:                     set_color(addr, get_fg_color(color));
127:                 } else {
128:                     set_color(addr, get_bg_color(color));
129:                 }
130:                 addr += video.fb_pixelwidth;
131:                 b--;

```

drivers/video/fbcon.c

Page 3/10

```

132:                } while (b >= 0);
133:                }
134:                addr += (video.fb_width - video.fb_char_width) * video.fb_pixelw
idth;
135:                }
136: }
137:
138: static void remove_cursor(struct vconsole *vc)
139: {
140:     int soffset;
141:     unsigned char *vidmem, *ch;
142:     short int *screen, sch;
143:
144:     vidmem = vc->vidmem;
145:     screen = vc->screen;
146:     soffset = (vc->cursor_y * vc->columns) + vc->cursor_x;
147:
148:     sch = screen[soffset];
149:     if(sch & 0xFF) {
150:         ch = &font_data[(sch & 0xFF) * video.fb_char_height];
151:     } else {
152:         ch = &font_data[SPACE_CHAR * video.fb_char_height];
153:     }
154:     draw_glyph(vidmem, vc->cursor_x, vc->cursor_y, ch, sch >> 8);
155: }
156:
157: static void draw_cursor(struct vconsole *vc)
158: {
159:     unsigned char *vidmem;
160:
161:     vidmem = vc->vidmem;
162:     draw_glyph(vidmem, vc->x, vc->y, cursor_shape, DEF_MODE >> 8);
163: }
164:
165: void fbcon_put_char(struct vconsole *vc, unsigned char ch)
166: {
167:     short int *screen;
168:     unsigned char *vidmem;
169:
170:     screen = vc->screen;
171:
172:     if(!(vc->flags & CONSOLE_HAS_FOCUS)) {
173:         screen[(vc->y * vc->columns) + vc->x] = vc->color_attr | ch;
174:         return;
175:     }
176:
177:     vidmem = vc->vidmem;
178:     draw_glyph(vidmem, vc->x, vc->y, &font_data[ch * video.fb_char_height],
vc->color_attr >> 8);
179:     screen[(vc->y * vc->columns) + vc->x] = vc->color_attr | ch;
180:     vcbuf[(video.buf_y * vc->columns) + vc->x] = vc->color_attr | ch;
181: }
182:
183: void fbcon_insert_char(struct vconsole *vc)
184: {
185:     int n, soffset;
186:     short int tmp, slast_ch;
187:     unsigned char *vidmem, *last_ch;
188:     short int *screen;
189:
190:     vidmem = vc->vidmem;
191:     screen = vc->screen;
192:     soffset = (vc->y * vc->columns) + vc->x;
193:     n = vc->x;
194:     last_ch = &font_data[SPACE_CHAR * video.fb_char_height];
195:     slast_ch = BLANK_MEM;
196:

```

drivers/video/fbcon.c

Page 4/10

```

197:         while(n < vc->columns) {
198:             tmp = screen[soffset];
199:             if(vc->flags & CONSOLE_HAS_FOCUS) {
200:                 draw_glyph(vidmem, n, vc->y, last_ch, vc->color_attr >>
8);
201:                 last_ch = &font_data[(tmp & 0xFF) * video.fb_char_height
];
202:             }
203:             memset_w(screen + soffset, slast_ch, 1);
204:             slast_ch = tmp;
205:             soffset++;
206:             n++;
207:         }
208:     }
209:
210: void fbcon_delete_char(struct vconsole *vc)
211: {
212:     int n, soffset;
213:     short int sch;
214:     unsigned char *vidmem, *ch;
215:     short int *screen;
216:
217:     vidmem = vc->vidmem;
218:     screen = vc->screen;
219:     soffset = (vc->y * vc->columns) + vc->x;
220:     n = vc->x;
221:
222:     while(n < vc->columns) {
223:         sch = screen[soffset + 1];
224:         if(vc->flags & CONSOLE_HAS_FOCUS) {
225:             if(sch & 0xFF) {
226:                 ch = &font_data[(sch & 0xFF) * video.fb_char_hei
ght];
227:             } else {
228:                 ch = &font_data[SPACE_CHAR * video.fb_char_heigh
t];
229:             }
230:             draw_glyph(vidmem, n, vc->y, ch, vc->color_attr >> 8);
231:         }
232:         memset_w(screen + soffset, sch, 1);
233:         soffset++;
234:         n++;
235:     }
236:     memset_w(screen + soffset, BLANK_MEM, 1);
237: }
238:
239: void fbcon_update_curpos(struct vconsole *vc)
240: {
241:     if(!(vc->flags & CONSOLE_HAS_FOCUS)) {
242:         return;
243:     }
244:
245:     /* remove old cursor */
246:     if(vc->x != vc->cursor_x || vc->y != vc->cursor_y) {
247:         remove_cursor(vc);
248:     }
249:
250:     if(video.flags & VPF_CURSOR_ON) {
251:         draw_cursor(vc);
252:     }
253:     vc->cursor_x = vc->x;
254:     vc->cursor_y = vc->y;
255: }
256:
257: void fbcon_show_cursor(struct vconsole *vc, int mode)
258: {
259:     switch(mode) {

```

drivers/video/fbcon.c

Page 5/10

```

260:         case COND:
261:             if(!(video.flags & VPF_CURSOR_ON)) {
262:                 break;
263:             }
264:             /* fall through */
265:         case ON:
266:             video.flags |= VPF_CURSOR_ON;
267:             fbcon_update_curpos(vc);
268:             break;
269:         case OFF:
270:             video.flags &= ~VPF_CURSOR_ON;
271:             fbcon_update_curpos(vc);
272:             break;
273:     }
274: }
275:
276: void fbcon_get_curpos(struct vconsole *vc)
277: {
278:     /* not used */
279: }
280:
281: void fbcon_write_screen(struct vconsole *vc, int from, int count, short int color)
282: {
283:     int n, n2, lines, columns, x, y;
284:     unsigned char *vidmem, *ch;
285:     short int *screen;
286:
287:     screen = vc->screen;
288:     if(!(vc->flags & CONSOLE_HAS_FOCUS)) {
289:         memset_w(screen + from, color, count);
290:         return;
291:     }
292:
293:     vidmem = vc->vidmem;
294:     ch = &font_data[SPACE_CHAR * video.fb_char_height];
295:     x = from % vc->columns;
296:     y = from / vc->columns;
297:     lines = count / vc->columns;
298:     columns = x + count;
299:     if(!lines) {
300:         lines = 1;
301:     }
302:     if(!columns) {
303:         columns = vc->columns;
304:     }
305:     for(n = 0; n < lines; n++) {
306:         for(n2 = x; n2 < columns; n2++) {
307:             draw_glyph(vidmem, n2, y + n, ch, color >> 8);
308:         }
309:         x = 0;
310:         columns = vc->columns;
311:     }
312:     memset_w(screen + from, color, count);
313: }
314:
315: void fbcon_blank_screen(struct vconsole *vc)
316: {
317:     unsigned char *vidmem;
318:
319:     if(vc->flags & CONSOLE_BLANKED) {
320:         return;
321:     }
322:
323:     vidmem = vc->vidmem;
324:     if(!(int)vidmem) {
325:         return;

```

drivers/video/fbcon.c

Page 6/10

```

326:         }
327:
328:         memset_b(vidmem, 0, video.fb_size);
329:         vc->flags |= CONSOLE_BLANKED;
330:     }
331:
332: void fbcon_scroll_screen(struct vconsole *vc, int top, int mode)
333: {
334:     int soffset, poffset, count;
335:     int x, y;
336:     short int *screen, sch, pch;
337:     unsigned char *vidmem, *ch;
338:
339:     vidmem = vc->vidmem;
340:     screen = vc->screen;
341:
342:     if(!top) {
343:         top = vc->top;
344:     }
345:     switch(mode) {
346:         case SCROLL_UP:
347:             if(vc->flags & CONSOLE_HAS_FOCUS) {
348:                 for(y = top + 1; y < vc->lines; y++) {
349:                     for(x = 0; x < vc->columns; x++) {
350:                         soffset = (y * vc->columns) + x;
351:                         poffset = ((y - 1) * vc->columns
) + x;
352:                         sch = screen[soffset];
353:                         pch = screen[poffset];
354:                         if(sch == pch) {
355:                             continue;
356:                         }
357:                         if(sch & 0xFF) {
358:                             ch = &font_data[(sch & 0
xFF) * video.fb_char_height];
359:                         } else {
360:                             ch = &font_data[SPACE_CH
AR * video.fb_char_height];
361:                         }
362:                         draw_glyph(vidmem, x, y - 1, ch,
sch >> 8);
363:                     }
364:                 }
365:                 if(!screen_is_off) {
366:                     count = video.fb_pitch * video.fb_char_h
eight;
367:                     memset_l(vidmem + video.fb_vsize - count
, 0, count / sizeof(unsigned int));
368:                 }
369:             }
370:             count = vc->columns * (vc->lines - top - 1);
371:             soffset = top * vc->columns;
372:             top = (top + 1) * vc->columns;
373:             if(vc->cursor_y) {
374:                 vc->cursor_y--;
375:             }
376:             memcpy_w(screen + soffset, screen + top, count);
377:             memset_w(screen + soffset + count, BLANK_MEM, top);
378:             break;
379:         case SCROLL_DOWN:
380:             for(y = vc->lines - 2; y >= top; y--) {
381:                 for(x = 0; x < vc->columns; x++) {
382:                     if(vc->flags & CONSOLE_HAS_FOCUS) {
383:                         soffset = (y * vc->columns) + x;
384:                         poffset = ((y + 1) * vc->columns
) + x;
385:                         sch = screen[soffset];

```

drivers/video/fbcon.c

Page 7/10

```

386:                                     pch = screen[poffset];
387:                                     if(sch == pch) {
388:                                         continue;
389:                                     }
390:                                     if(sch & 0xFF) {
391:                                         ch = &font_data[(sch & 0
xFF) * video.fb_char_height];
392:                                     } else {
393:                                         ch = &font_data[SPACE_CH
AR * video.fb_char_height];
394:                                     }
395:                                     draw_glyph(vidmem, x, y + 1, ch,
sch >> 8);
396:                                     }
397:                                 }
398:                                 memcpy_w(screen + (vc->columns * (y + 1)), scree
n + (vc->columns * y), vc->columns);
399:                                 }
400:                                 if((vc->flags & CONSOLE_HAS_FOCUS) && !screen_is_off) {
401:                                     count = video.fb_pitch * video.fb_char_height;
402:                                     memset_l(vidmem + (top * count), 0, count / size
of(unsigned int));
403:                                 }
404:                                 memset_w(screen + (top * vc->columns), BLANK_MEM, vc->co
lumn);
405:                                 break;
406:                             }
407:                             return;
408:     }
409:
410: void fbcon_restore_screen(struct vconsole *vc)
411: {
412:     int x, y;
413:     short int *screen, sch;
414:     unsigned char *vidmem, *ch, c;
415:
416:     vidmem = vc->vidmem;
417:     screen = vc->screen;
418:
419:     if(!screen_is_off && !video.buf_top) {
420:         memset_b(vidmem, 0, video.fb_size);
421:     }
422:     for(y = 0; y < video.lines; y++) {
423:         for(x = 0; x < vc->columns; x++) {
424:             sch = screen[(y * vc->columns) + x];
425:             c = sch & 0xFF;
426:             if(!c || (c == SPACE_CHAR && !(sch >> 8))) {
427:                 continue;
428:             }
429:             ch = &font_data[c * video.fb_char_height];
430:             draw_glyph(vidmem, x, y, ch, sch >> 8);
431:         }
432:     }
433:     vc->flags &= ~CONSOLE_BLANKED;
434: }
435:
436: void fbcon_screen_on(struct vconsole *vc)
437: {
438:     unsigned long int flags;
439:     struct callout_req creq;
440:
441:     if(screen_is_off) {
442:         screen_is_off = 0;
443:         SAVE_FLAGS(flags); CLI();
444:         fbcon_restore_screen(vc);
445:         fbcon_update_curpos(vc);
446:         RESTORE_FLAGS(flags);

```


drivers/video/fbcon.c

Page 8/10

```

447:             vc->flags &= ~CONSOLE_BLANKED;
448:         }
449:
450:         if(BLANK_INTERVAL) {
451:             creq.fn = fbcon_screen_off;
452:             creq.arg = (unsigned int)vc;
453:             add_callout(&creq, BLANK_INTERVAL);
454:         }
455:     }
456:
457: void fbcon_screen_off(unsigned int arg)
458: {
459:     struct vconsole *vc;
460:     unsigned long int flags;
461:
462:     vc = (struct vconsole *)arg;
463:     screen_is_off = 1;
464:     SAVE_FLAGS(flags); CLI();
465:     fbcon_blank_screen(vc);
466:     RESTORE_FLAGS(flags);
467: }
468:
469: void fbcon_buf_scroll(struct vconsole *vc, int mode)
470: {
471:     short int sch;
472:     int y, x, offset;
473:     unsigned char *vidmem, *ch;
474:
475:     if(video.buf_y <= SCREEN_LINES) {
476:         return;
477:     }
478:
479:     vidmem = vc->vidmem;
480:
481:     if(mode == SCROLL_UP) {
482:         if(video.buf_top < 0) {
483:             return;
484:         }
485:         if(!video.buf_top) {
486:             video.buf_top = (video.buf_y - SCREEN_LINES + 1) * SCREE
N_COLS;
487:         }
488:         video.buf_top -= (SCREEN_LINES / 2) * SCREEN_COLS;
489:         if(video.buf_top < 0) {
490:             video.buf_top = 0;
491:         }
492:         for(offset = 0, y = 0; y < video.lines; y++) {
493:             for(x = 0; x < vc->columns; x++, offset++) {
494:                 sch = vbuf[video.buf_top + offset];
495:                 if(sch & 0xFF) {
496:                     ch = &font_data[(sch & 0xFF) * video.fb_
char_height];
497:                 } else {
498:                     ch = &font_data[SPACE_CHAR * video.fb_ch
ar_height];
499:                 }
500:                 draw_glyph(vidmem, x, y, ch, sch >> 8);
501:             }
502:         }
503:         if(!video.buf_top) {
504:             video.buf_top = -1;
505:         }
506:         fbcon_show_cursor(vc, OFF);
507:         return;
508:     }
509:     if(mode == SCROLL_DOWN) {
510:         if(!video.buf_top) {

```

drivers/video/fbcon.c

Page 9/10

```

511:             return;
512:         }
513:         if(video.buf_top == video.buf_y * SCREEN_COLS) {
514:             return;
515:         }
516:         if(video.buf_top < 0) {
517:             video.buf_top = 0;
518:         }
519:         video.buf_top += (SCREEN_LINES / 2) * SCREEN_COLS;
520:         if(video.buf_top >= (video.buf_y - SCREEN_LINES + 1) * SCREEN_CO
LS) {
521:             video.buf_top = (video.buf_y - SCREEN_LINES + 1) * SCREE
N_COLS;
522:         }
523:         for(offset = 0, y = 0; y < video.lines; y++) {
524:             for(x = 0; x < vc->columns; x++, offset++) {
525:                 sch = vcbuf[video.buf_top + offset];
526:                 if(sch & 0xFF) {
527:                     ch = &font_data[(sch & 0xFF) * video.fb_
char_height];
528:                 } else {
529:                     ch = &font_data[SPACE_CHAR * video.fb_ch
ar_height];
530:                 }
531:                 draw_glyph(vidmem, x, y, ch, sch >> 8);
532:             }
533:         }
534:         if(video.buf_top >= (video.buf_y - SCREEN_LINES + 1) * SCREEN_CO
LS) {
535:             fbcon_show_cursor(vc, ON);
536:             fbcon_update_curpos(vc);
537:         }
538:         return;
539:     }
540: }
541:
542: void fbcon_cursor_blink(unsigned int arg)
543: {
544:     struct vconsole *vc;
545:     struct callout_req creq;
546:     static int blink_on = 0;
547:
548:     vc = (struct vconsole *)arg;
549:     if(!(vc->flags & CONSOLE_HAS_FOCUS)) {
550:         return;
551:     }
552:
553:     if(video.flags & VPF_CURSOR_ON && !screen_is_off) {
554:         if(blink_on) {
555:             draw_cursor(vc);
556:         } else {
557:             remove_cursor(vc);
558:         }
559:     }
560:     blink_on = !blink_on;
561:     creq.fn = fbcon_cursor_blink;
562:     creq.arg = arg;
563:     add_callout(&creq, 25);           /* 250ms */
564: }
565:
566: void fbcon_init(void)
567: {
568:     struct fbcon_font_desc *font_desc;
569:
570:     video.put_char = fbcon_put_char;
571:     video.insert_char = fbcon_insert_char;
572:     video.delete_char = fbcon_delete_char;

```

drivers/video/fbcon.c

Page 10/10

```
573:         video.update_curpos = fbcon_update_curpos;
574:         video.show_cursor = fbcon_show_cursor;
575:         video.get_curpos = fbcon_get_curpos;
576:         video.write_screen = fbcon_write_screen;
577:         video.blank_screen = fbcon_blank_screen;
578:         video.scroll_screen = fbcon_scroll_screen;
579:         video.restore_screen = fbcon_restore_screen;
580:         video.screen_on = fbcon_screen_on;
581:         video.buf_scroll = fbcon_buf_scroll;
582:         video.cursor_blink = fbcon_cursor_blink;
583:
584:         if (!(font_desc = fbcon_find_font(video.fb_char_height))) {
585:             font_desc = fbcon_find_font(16);
586:         }
587:         font_data = font_desc->data;
588:         cursor_shape = font_desc->cursorshape;
589: }
```

drivers/video/font-lat9-8x10.c

Page 1/47

```

1: #include <fiwix/font.h>
2:
3: static unsigned char fontdata_8x10[] = {
4:     /* 0x00 */
5:     0x7e, /* -#####- */
6:     0xc3, /* ##----## */
7:     0x99, /* #--##--# */
8:     0xf9, /* #####--# */
9:     0xf3, /* #####--# */
10:    0xe7, /* ###--### */
11:    0xff, /* ######## */
12:    0xe7, /* ###--### */
13:    0x7e, /* -#####- */
14:    0x00, /* ----- */
15:
16:    /* 0x01 */
17:    0x00, /* ----- */
18:    0x00, /* ----- */
19:    0x76, /* -###-##- */
20:    0xdc, /* #-###-- */
21:    0x00, /* ----- */
22:    0x76, /* -###-##- */
23:    0xdc, /* #-###-- */
24:    0x00, /* ----- */
25:    0x00, /* ----- */
26:    0x00, /* ----- */
27:
28:    /* 0x02 */
29:    0x00, /* ----- */
30:    0x76, /* -###-##- */
31:    0xd8, /* #-###-- */
32:    0xd8, /* #-###-- */
33:    0xdc, /* #-###-- */
34:    0xd8, /* #-###-- */
35:    0xd8, /* #-###-- */
36:    0x76, /* -###-##- */
37:    0x00, /* ----- */
38:    0x00, /* ----- */
39:
40:    /* 0x03 */
41:    0x00, /* ----- */
42:    0x00, /* ----- */
43:    0x00, /* ----- */
44:    0x6e, /* -##-###- */
45:    0xd8, /* #-###-- */
46:    0xde, /* #-#####- */
47:    0xd8, /* #-###-- */
48:    0x6e, /* -##-###- */
49:    0x00, /* ----- */
50:    0x00, /* ----- */
51:
52:    /* 0x04 */
53:    0x00, /* ----- */
54:    0x10, /* ---#---- */
55:    0x38, /* --###---- */
56:    0x7c, /* -#####-- */
57:    0xfe, /* ########- */
58:    0x7c, /* -#####-- */
59:    0x38, /* --###---- */
60:    0x10, /* ---#---- */
61:    0x00, /* ----- */
62:    0x00, /* ----- */
63:
64:    /* 0x05 */
65:    0xa0, /* #-#----- */
66:    0xa0, /* #-#----- */
67:    0xe0, /* ###----- */

```

drivers/video/font-lat9-8x10.c

Page 2/47

```

68:      0xae, /* #-###- */
69:      0xa4, /* #-#--#-- */
70:      0x04, /* -----#-- */
71:      0x04, /* -----#-- */
72:      0x04, /* -----#-- */
73:      0x00, /* ----- */
74:      0x00, /* ----- */
75:
76:      /* 0x06 */
77:      0xe0, /* ###----- */
78:      0x80, /* #----- */
79:      0xc0, /* ##----- */
80:      0x8e, /* #---###- */
81:      0x88, /* #---#--- */
82:      0x0c, /* ----##-- */
83:      0x08, /* ----#--- */
84:      0x08, /* ----#--- */
85:      0x00, /* ----- */
86:      0x00, /* ----- */
87:
88:      /* 0x07 */
89:      0x60, /* -##----- */
90:      0x80, /* #----- */
91:      0x80, /* #----- */
92:      0x8c, /* #---##-- */
93:      0x6a, /* -##-##-#- */
94:      0x0c, /* ----##-- */
95:      0x0a, /* ----#-#- */
96:      0x0a, /* ----#-#- */
97:      0x00, /* ----- */
98:      0x00, /* ----- */
99:
100:     /* 0x08 */
101:     0x80, /* #----- */
102:     0x80, /* #----- */
103:     0x80, /* #----- */
104:     0x8e, /* #---###- */
105:     0xe8, /* ###-#--- */
106:     0x0c, /* ----##-- */
107:     0x08, /* ----#--- */
108:     0x08, /* ----#--- */
109:     0x00, /* ----- */
110:     0x00, /* ----- */
111:
112:     /* 0x09 */
113:     0x22, /* --#---#- */
114:     0x88, /* #---#--- */
115:     0x22, /* --#---#- */
116:     0x88, /* #---#--- */
117:     0x22, /* --#---#- */
118:     0x88, /* #---#--- */
119:     0x22, /* --#---#- */
120:     0x88, /* #---#--- */
121:     0x22, /* --#---#- */
122:     0x88, /* #---#--- */
123:
124:     /* 0x0a */
125:     0x55, /* -#-#-#-#- */
126:     0xaa, /* #-#-#-#-#- */
127:     0x55, /* -#-#-#-#- */
128:     0xaa, /* #-#-#-#-#- */
129:     0x55, /* -#-#-#-#- */
130:     0xaa, /* #-#-#-#-#- */
131:     0x55, /* -#-#-#-#- */
132:     0xaa, /* #-#-#-#-#- */
133:     0x55, /* -#-#-#-#- */
134:     0xaa, /* #-#-#-#-#- */

```

drivers/video/font-lat9-8x10.c

Page 3/47

```

135:
136:          /* 0x0b          */
137:    0xee,  /* ###-###- */
138:    0xbb,  /* #-###-## */
139:    0xee,  /* ###-###- */
140:    0xbb,  /* #-###-## */
141:    0xee,  /* ###-###- */
142:    0xbb,  /* #-###-## */
143:    0xee,  /* ###-###- */
144:    0xbb,  /* #-###-## */
145:    0xee,  /* ###-###- */
146:    0xbb,  /* #-###-## */
147:
148:          /* 0x0c          */
149:    0xff,  /* ##### */
150:    0xff,  /* ##### */
151:    0xff,  /* ##### */
152:    0xff,  /* ##### */
153:    0xff,  /* ##### */
154:    0xff,  /* ##### */
155:    0xff,  /* ##### */
156:    0xff,  /* ##### */
157:    0xff,  /* ##### */
158:    0xff,  /* ##### */
159:
160:          /* 0x0d          */
161:    0x00,  /* ----- */
162:    0x00,  /* ----- */
163:    0x00,  /* ----- */
164:    0x00,  /* ----- */
165:    0x00,  /* ----- */
166:    0xff,  /* ##### */
167:    0xff,  /* ##### */
168:    0xff,  /* ##### */
169:    0xff,  /* ##### */
170:    0xff,  /* ##### */
171:
172:          /* 0x0e          */
173:    0xff,  /* ##### */
174:    0xff,  /* ##### */
175:    0xff,  /* ##### */
176:    0xff,  /* ##### */
177:    0xff,  /* ##### */
178:    0x00,  /* ----- */
179:    0x00,  /* ----- */
180:    0x00,  /* ----- */
181:    0x00,  /* ----- */
182:    0x00,  /* ----- */
183:
184:          /* 0x0f          */
185:    0xf0,  /* ####- */
186:    0xf0,  /* ####- */
187:    0xf0,  /* ####- */
188:    0xf0,  /* ####- */
189:    0xf0,  /* ####- */
190:    0xf0,  /* ####- */
191:    0xf0,  /* ####- */
192:    0xf0,  /* ####- */
193:    0xf0,  /* ####- */
194:    0xf0,  /* ####- */
195:
196:          /* 0x10          */
197:    0x0f,  /* ----#### */
198:    0x0f,  /* ----#### */
199:    0x0f,  /* ----#### */
200:    0x0f,  /* ----#### */
201:    0x0f,  /* ----#### */

```

drivers/video/font-lat9-8x10.c

Page 4/47

```

202:      0x0f, /* ----#### */
203:      0x0f, /* ----#### */
204:      0x0f, /* ----#### */
205:      0x0f, /* ----#### */
206:      0x0f, /* ----#### */
207:
208:          /* 0x11 */
209:      0x90, /* #-#----- */
210:      0xd0, /* ##-#----- */
211:      0xf0, /* ####----- */
212:      0xb4, /* #-##-#--- */
213:      0x94, /* #-#-#--- */
214:      0x04, /* -----#-- */
215:      0x04, /* -----#-- */
216:      0x07, /* -----### */
217:      0x00, /* ----- */
218:      0x00, /* ----- */
219:
220:          /* 0x12 */
221:      0xa0, /* #-#----- */
222:      0xa0, /* #-#----- */
223:      0xa0, /* #-#----- */
224:      0xae, /* #-#-##-#- */
225:      0x44, /* -#---#--- */
226:      0x04, /* -----#-- */
227:      0x04, /* -----#-- */
228:      0x04, /* -----#-- */
229:      0x00, /* ----- */
230:      0x00, /* ----- */
231:
232:          /* 0x13 */
233:      0x00, /* ----- */
234:      0x18, /* ---##---- */
235:      0x30, /* --##----- */
236:      0x60, /* -##----- */
237:      0x30, /* --##----- */
238:      0x18, /* ---##---- */
239:      0x00, /* ----- */
240:      0xfc, /* #####--- */
241:      0x00, /* ----- */
242:      0x00, /* ----- */
243:
244:          /* 0x14 */
245:      0x00, /* ----- */
246:      0x60, /* -##----- */
247:      0x30, /* --##----- */
248:      0x18, /* ---##---- */
249:      0x30, /* --##----- */
250:      0x60, /* -##----- */
251:      0x00, /* ----- */
252:      0xfc, /* #####--- */
253:      0x00, /* ----- */
254:      0x00, /* ----- */
255:
256:          /* 0x15 */
257:      0x00, /* ----- */
258:      0x06, /* -----##- */
259:      0x0c, /* -----##- */
260:      0xfe, /* #####--- */
261:      0x18, /* ---##---- */
262:      0x30, /* --##----- */
263:      0xfe, /* #####--- */
264:      0x60, /* -##----- */
265:      0xc0, /* ##----- */
266:      0x00, /* ----- */
267:
268:          /* 0x16 */

```

drivers/video/font-lat9-8x10.c Page 5/47

```

269:      0x00, /* ----- */
270:      0x02, /* -----# */
271:      0x0e, /* ----### */
272:      0x3e, /* ---##### */
273:      0xfe, /* #####- */
274:      0x3e, /* ---##### */
275:      0x0e, /* ----### */
276:      0x02, /* -----# */
277:      0x00, /* ----- */
278:      0x00, /* ----- */
279:
280:      /* 0x17 */
281:      0x00, /* ----- */
282:      0x80, /* #----- */
283:      0xe0, /* ###----- */
284:      0xf8, /* #####--- */
285:      0xfe, /* #####- */
286:      0xf8, /* #####--- */
287:      0xe0, /* ###----- */
288:      0x80, /* #----- */
289:      0x00, /* ----- */
290:      0x00, /* ----- */
291:
292:      /* 0x18 */
293:      0x00, /* ----- */
294:      0x18, /* ---##--- */
295:      0x3c, /* --####-- */
296:      0x7e, /* -#####- */
297:      0x18, /* ---##--- */
298:      0x18, /* ---##--- */
299:      0x18, /* ---##--- */
300:      0x18, /* ---##--- */
301:      0x00, /* ----- */
302:      0x00, /* ----- */
303:
304:      /* 0x19 */
305:      0x00, /* ----- */
306:      0x18, /* ---##--- */
307:      0x18, /* ---##--- */
308:      0x18, /* ---##--- */
309:      0x18, /* ---##--- */
310:      0x7e, /* -#####- */
311:      0x3c, /* --####-- */
312:      0x18, /* ---##--- */
313:      0x00, /* ----- */
314:      0x00, /* ----- */
315:
316:      /* 0x1a */
317:      0x00, /* ----- */
318:      0x00, /* ----- */
319:      0x18, /* ---##--- */
320:      0x0c, /* ----##-- */
321:      0xfe, /* #####- */
322:      0x0c, /* ----##-- */
323:      0x18, /* ---##--- */
324:      0x00, /* ----- */
325:      0x00, /* ----- */
326:      0x00, /* ----- */
327:
328:      /* 0x1b */
329:      0x00, /* ----- */
330:      0x00, /* ----- */
331:      0x30, /* --##--- */
332:      0x60, /* -##----- */
333:      0xfe, /* #####- */
334:      0x60, /* -##----- */
335:      0x30, /* --##--- */

```


drivers/video/font-lat9-8x10.c

Page 6/47

```

336:      0x00, /* ----- */
337:      0x00, /* ----- */
338:      0x00, /* ----- */
339:
340:          /* 0x1c */
341:      0x00, /* ----- */
342:      0x18, /* ---##--- */
343:      0x3c, /* --####-- */
344:      0x7e, /* -#####- */
345:      0x18, /* ---##--- */
346:      0x18, /* ---##--- */
347:      0x7e, /* -#####- */
348:      0x3c, /* --####-- */
349:      0x18, /* ---##--- */
350:      0x00, /* ----- */
351:
352:          /* 0x1d */
353:      0x00, /* ----- */
354:      0x24, /* --#--#-- */
355:      0x66, /* -##---##- */
356:      0xff, /* ##### */
357:      0x66, /* -##---##- */
358:      0x24, /* --#--#-- */
359:      0x00, /* ----- */
360:      0x00, /* ----- */
361:      0x00, /* ----- */
362:      0x00, /* ----- */
363:
364:          /* 0x1e */
365:      0x06, /* -----##- */
366:      0x06, /* -----##- */
367:      0x36, /* --##-##- */
368:      0x66, /* -##---##- */
369:      0xfe, /* #####-#- */
370:      0x60, /* -##----- */
371:      0x30, /* --##----- */
372:      0x00, /* ----- */
373:      0x00, /* ----- */
374:      0x00, /* ----- */
375:
376:          /* 0x1f */
377:      0x00, /* ----- */
378:      0x00, /* ----- */
379:      0xc0, /* ##----- */
380:      0x7c, /* -#####- */
381:      0x6e, /* -##-###- */
382:      0x6c, /* -##-##-#- */
383:      0x6c, /* -##-##-#- */
384:      0x6c, /* -##-##-#- */
385:      0x00, /* ----- */
386:      0x00, /* ----- */
387:
388:          /* 0x20 (' ') */
389:      0x00, /* ----- */
390:      0x00, /* ----- */
391:      0x00, /* ----- */
392:      0x00, /* ----- */
393:      0x00, /* ----- */
394:      0x00, /* ----- */
395:      0x00, /* ----- */
396:      0x00, /* ----- */
397:      0x00, /* ----- */
398:      0x00, /* ----- */
399:
400:          /* 0x21 ('!') */
401:      0x00, /* ----- */
402:      0x30, /* ---##--- */

```

drivers/video/font-lat9-8x10.c

Page 7/47

```

403:      0x78,    /* #####--- */
404:      0x78,    /* #####--- */
405:      0x30,    /* ---#----- */
406:      0x30,    /* ---#----- */
407:      0x00,    /* ----- */
408:      0x30,    /* ---#----- */
409:      0x00,    /* ----- */
410:      0x00,    /* ----- */
411:
412:      /* 0x22 ('"') */
413:      0x00,    /* ----- */
414:      0x6c,    /* -##-##- */
415:      0x6c,    /* -##-##- */
416:      0x6c,    /* -##-##- */
417:      0x00,    /* ----- */
418:      0x00,    /* ----- */
419:      0x00,    /* ----- */
420:      0x00,    /* ----- */
421:      0x00,    /* ----- */
422:      0x00,    /* ----- */
423:
424:      /* 0x23 ('#') */
425:      0x00,    /* ----- */
426:      0x6c,    /* -##-##- */
427:      0x6c,    /* -##-##- */
428:      0xfe,    /* #####- */
429:      0x6c,    /* -##-##- */
430:      0xfe,    /* #####- */
431:      0x6c,    /* -##-##- */
432:      0x6c,    /* -##-##- */
433:      0x00,    /* ----- */
434:      0x00,    /* ----- */
435:
436:      /* 0x24 ('$') */
437:      0x00,    /* ----- */
438:      0x30,    /* ---#----- */
439:      0x7c,    /* -#####- */
440:      0xc0,    /* ##----- */
441:      0x78,    /* -#####- */
442:      0x0c,    /* ----##- */
443:      0xf8,    /* #####- */
444:      0x30,    /* ---#----- */
445:      0x00,    /* ----- */
446:      0x00,    /* ----- */
447:
448:      /* 0x25 ('%') */
449:      0x00,    /* ----- */
450:      0x00,    /* ----- */
451:      0xc6,    /* ##---##- */
452:      0xcc,    /* ##---##- */
453:      0x18,    /* ---#----- */
454:      0x30,    /* ---#----- */
455:      0x66,    /* -##-##- */
456:      0xc6,    /* ##---##- */
457:      0x00,    /* ----- */
458:      0x00,    /* ----- */
459:
460:      /* 0x26 ('&') */
461:      0x00,    /* ----- */
462:      0x38,    /* ---##----- */
463:      0x6c,    /* -##-##- */
464:      0x38,    /* ---##----- */
465:      0x76,    /* -###-##- */
466:      0xdc,    /* ##-##- */
467:      0xcc,    /* ##---##- */
468:      0x76,    /* -###-##- */
469:      0x00,    /* ----- */

```

drivers/video/font-lat9-8x10.c

Page 8/47

```

470:      0x00,    /* ----- */
471:
472:      /* 0x27 ('') */
473:      0x30,    /* --##---- */
474:      0x30,    /* --##---- */
475:      0x60,    /* -##----- */
476:      0x00,    /* ----- */
477:      0x00,    /* ----- */
478:      0x00,    /* ----- */
479:      0x00,    /* ----- */
480:      0x00,    /* ----- */
481:      0x00,    /* ----- */
482:      0x00,    /* ----- */
483:
484:      /* 0x28 ('(') */
485:      0x00,    /* ----- */
486:      0x18,    /* ---##---- */
487:      0x30,    /* --##---- */
488:      0x60,    /* -##----- */
489:      0x60,    /* -##----- */
490:      0x60,    /* -##----- */
491:      0x30,    /* --##---- */
492:      0x18,    /* ---##---- */
493:      0x00,    /* ----- */
494:      0x00,    /* ----- */
495:
496:      /* 0x29 ('') */
497:      0x00,    /* ----- */
498:      0x60,    /* -##----- */
499:      0x30,    /* --##---- */
500:      0x18,    /* ---##---- */
501:      0x18,    /* ---##---- */
502:      0x18,    /* ---##---- */
503:      0x30,    /* --##---- */
504:      0x60,    /* -##----- */
505:      0x00,    /* ----- */
506:      0x00,    /* ----- */
507:
508:      /* 0x2a ('*') */
509:      0x00,    /* ----- */
510:      0x00,    /* ----- */
511:      0x66,    /* -##--##- */
512:      0x3c,    /* ---####-- */
513:      0xff,    /* ######## */
514:      0x3c,    /* ---####-- */
515:      0x66,    /* -##--##- */
516:      0x00,    /* ----- */
517:      0x00,    /* ----- */
518:      0x00,    /* ----- */
519:
520:      /* 0x2b ('+') */
521:      0x00,    /* ----- */
522:      0x00,    /* ----- */
523:      0x30,    /* --##---- */
524:      0x30,    /* --##---- */
525:      0xfc,    /* #####--- */
526:      0x30,    /* --##---- */
527:      0x30,    /* --##---- */
528:      0x00,    /* ----- */
529:      0x00,    /* ----- */
530:      0x00,    /* ----- */
531:
532:      /* 0x2c (',') */
533:      0x00,    /* ----- */
534:      0x00,    /* ----- */
535:      0x00,    /* ----- */
536:      0x00,    /* ----- */

```

drivers/video/font-lat9-8x10.c

Page 9/47

```

537:      0x00, /* ----- */
538:      0x00, /* ----- */
539:      0x30, /* --##---- */
540:      0x30, /* --##---- */
541:      0x60, /* -##----- */
542:      0x00, /* ----- */
543:
544:          /* 0x2d ('-') */
545:      0x00, /* ----- */
546:      0x00, /* ----- */
547:      0x00, /* ----- */
548:      0x00, /* ----- */
549:      0xfc, /* #####-- */
550:      0x00, /* ----- */
551:      0x00, /* ----- */
552:      0x00, /* ----- */
553:      0x00, /* ----- */
554:      0x00, /* ----- */
555:
556:          /* 0x2e ('.') */
557:      0x00, /* ----- */
558:      0x00, /* ----- */
559:      0x00, /* ----- */
560:      0x00, /* ----- */
561:      0x00, /* ----- */
562:      0x00, /* ----- */
563:      0x30, /* --##---- */
564:      0x30, /* --##---- */
565:      0x00, /* ----- */
566:      0x00, /* ----- */
567:
568:          /* 0x2f ('/') */
569:      0x00, /* ----- */
570:      0x00, /* ----- */
571:      0x06, /* -----##- */
572:      0x0c, /* ----##-- */
573:      0x18, /* ----##-- */
574:      0x30, /* --##---- */
575:      0x60, /* -##----- */
576:      0xc0, /* ##----- */
577:      0x00, /* ----- */
578:      0x00, /* ----- */
579:
580:          /* 0x30 ('0') */
581:      0x00, /* ----- */
582:      0x7c, /* -#####- */
583:      0xc6, /* ##---##- */
584:      0xc6, /* ##---##- */
585:      0xd6, /* ##-##-##- */
586:      0xc6, /* ##---##- */
587:      0xc6, /* ##---##- */
588:      0x7c, /* -#####- */
589:      0x00, /* ----- */
590:      0x00, /* ----- */
591:
592:          /* 0x31 ('1') */
593:      0x00, /* ----- */
594:      0x30, /* --##---- */
595:      0x70, /* -##----- */
596:      0x30, /* --##---- */
597:      0x30, /* --##---- */
598:      0x30, /* --##---- */
599:      0x30, /* --##---- */
600:      0xfc, /* #####-- */
601:      0x00, /* ----- */
602:      0x00, /* ----- */
603:

```

drivers/video/font-lat9-8x10.c

Page 10/47

```

604:                /* 0x32 ('2') */
605:    0x00,          /* ----- */
606:    0x78,          /* -####--- */
607:    0xcc,          /* ##---##-- */
608:    0x0c,          /* ----##-- */
609:    0x38,          /* --###--- */
610:    0x60,          /* -##----- */
611:    0xcc,          /* ##---##-- */
612:    0xfc,          /* #####--- */
613:    0x00,          /* ----- */
614:    0x00,          /* ----- */
615:
616:                /* 0x33 ('3') */
617:    0x00,          /* ----- */
618:    0x78,          /* -####--- */
619:    0xcc,          /* ##---##-- */
620:    0x0c,          /* ----##-- */
621:    0x38,          /* --###--- */
622:    0x0c,          /* ----##-- */
623:    0xcc,          /* ##---##-- */
624:    0x78,          /* -####--- */
625:    0x00,          /* ----- */
626:    0x00,          /* ----- */
627:
628:                /* 0x34 ('4') */
629:    0x00,          /* ----- */
630:    0x1c,          /* ---###-- */
631:    0x3c,          /* --####-- */
632:    0x6c,          /* -##-##-- */
633:    0xcc,          /* ##---##-- */
634:    0xfe,          /* #####--- */
635:    0x0c,          /* ----##-- */
636:    0x1e,          /* ---###-- */
637:    0x00,          /* ----- */
638:    0x00,          /* ----- */
639:
640:                /* 0x35 ('5') */
641:    0x00,          /* ----- */
642:    0xfc,          /* #####--- */
643:    0xc0,          /* ##----- */
644:    0xf8,          /* #####--- */
645:    0x0c,          /* ----##-- */
646:    0x0c,          /* ----##-- */
647:    0xcc,          /* ##---##-- */
648:    0x78,          /* -####--- */
649:    0x00,          /* ----- */
650:    0x00,          /* ----- */
651:
652:                /* 0x36 ('6') */
653:    0x00,          /* ----- */
654:    0x38,          /* --###--- */
655:    0x60,          /* -##----- */
656:    0xc0,          /* ##----- */
657:    0xf8,          /* #####--- */
658:    0xcc,          /* ##---##-- */
659:    0xcc,          /* ##---##-- */
660:    0x78,          /* -####--- */
661:    0x00,          /* ----- */
662:    0x00,          /* ----- */
663:
664:                /* 0x37 ('7') */
665:    0x00,          /* ----- */
666:    0xfc,          /* #####--- */
667:    0xcc,          /* ##---##-- */
668:    0x0c,          /* ----##-- */
669:    0x18,          /* ---##--- */
670:    0x30,          /* --##----- */

```

drivers/video/font-lat9-8x10.c

Page 11/47

```

671:      0x30, /* --##----- */
672:      0x30, /* --##----- */
673:      0x00, /* ----- */
674:      0x00, /* ----- */
675:
676:          /* 0x38 ('8') */
677:      0x00, /* ----- */
678:      0x78, /* -####----- */
679:      0xc0, /* ##--##-- */
680:      0xc0, /* ##--##-- */
681:      0x78, /* -####----- */
682:      0xc0, /* ##--##-- */
683:      0xc0, /* ##--##-- */
684:      0x78, /* -####----- */
685:      0x00, /* ----- */
686:      0x00, /* ----- */
687:
688:          /* 0x39 ('9') */
689:      0x00, /* ----- */
690:      0x78, /* -####----- */
691:      0xc0, /* ##--##-- */
692:      0xc0, /* ##--##-- */
693:      0x7c, /* -#####----- */
694:      0x0c, /* ----##-- */
695:      0x18, /* ----##-- */
696:      0x70, /* -###----- */
697:      0x00, /* ----- */
698:      0x00, /* ----- */
699:
700:          /* 0x3a (':') */
701:      0x00, /* ----- */
702:      0x00, /* ----- */
703:      0x30, /* --##----- */
704:      0x30, /* --##----- */
705:      0x00, /* ----- */
706:      0x00, /* ----- */
707:      0x30, /* --##----- */
708:      0x30, /* --##----- */
709:      0x00, /* ----- */
710:      0x00, /* ----- */
711:
712:          /* 0x3b (';') */
713:      0x00, /* ----- */
714:      0x00, /* ----- */
715:      0x30, /* --##----- */
716:      0x30, /* --##----- */
717:      0x00, /* ----- */
718:      0x00, /* ----- */
719:      0x30, /* --##----- */
720:      0x30, /* --##----- */
721:      0x60, /* -##----- */
722:      0x00, /* ----- */
723:
724:          /* 0x3c ('<') */
725:      0x00, /* ----- */
726:      0x18, /* ----##-- */
727:      0x30, /* --##----- */
728:      0x60, /* -##----- */
729:      0xc0, /* ##----- */
730:      0x60, /* -##----- */
731:      0x30, /* --##----- */
732:      0x18, /* ----##-- */
733:      0x00, /* ----- */
734:      0x00, /* ----- */
735:
736:          /* 0x3d ('=') */
737:      0x00, /* ----- */

```

drivers/video/font-lat9-8x10.c

Page 12/47

```

738:      0x00, /* ----- */
739:      0x00, /* ----- */
740:      0xfc, /* #####-- */
741:      0x00, /* ----- */
742:      0x00, /* ----- */
743:      0xfc, /* #####-- */
744:      0x00, /* ----- */
745:      0x00, /* ----- */
746:      0x00, /* ----- */
747:
748:      /* 0x3e ('>') */
749:      0x00, /* ----- */
750:      0x60, /* -##----- */
751:      0x30, /* --##----- */
752:      0x18, /* ----##---- */
753:      0x0c, /* -----##-- */
754:      0x18, /* ----##---- */
755:      0x30, /* --##----- */
756:      0x60, /* -##----- */
757:      0x00, /* ----- */
758:      0x00, /* ----- */
759:
760:      /* 0x3f ('?') */
761:      0x00, /* ----- */
762:      0x78, /* -####---- */
763:      0xcc, /* ##---##-- */
764:      0x0c, /* -----##-- */
765:      0x18, /* ----##---- */
766:      0x30, /* --##----- */
767:      0x00, /* ----- */
768:      0x30, /* --##----- */
769:      0x00, /* ----- */
770:      0x00, /* ----- */
771:
772:      /* 0x40 ('@') */
773:      0x00, /* ----- */
774:      0x7c, /* -####---- */
775:      0xc6, /* ##---##-- */
776:      0xde, /* ##-####- */
777:      0xde, /* ##-####- */
778:      0xde, /* ##-####- */
779:      0xc0, /* ##----- */
780:      0x78, /* -####---- */
781:      0x00, /* ----- */
782:      0x00, /* ----- */
783:
784:      /* 0x41 ('A') */
785:      0x00, /* ----- */
786:      0x30, /* --##----- */
787:      0x78, /* -####---- */
788:      0xcc, /* ##---##-- */
789:      0xcc, /* ##---##-- */
790:      0xfc, /* #####-- */
791:      0xcc, /* ##---##-- */
792:      0xcc, /* ##---##-- */
793:      0x00, /* ----- */
794:      0x00, /* ----- */
795:
796:      /* 0x42 ('B') */
797:      0x00, /* ----- */
798:      0xfc, /* #####-- */
799:      0x66, /* -##---##- */
800:      0x66, /* -##---##- */
801:      0x7c, /* -####---- */
802:      0x66, /* -##---##- */
803:      0x66, /* -##---##- */
804:      0xfc, /* #####-- */

```

drivers/video/font-lat9-8x10.c

Page 13/47

```

805:      0x00, /* ----- */
806:      0x00, /* ----- */
807:
808:      /* 0x43 ('C') */
809:      0x00, /* ----- */
810:      0x3c, /* --####-- */
811:      0x66, /* -##--##- */
812:      0xc0, /* ##----- */
813:      0xc0, /* ##----- */
814:      0xc0, /* ##----- */
815:      0x66, /* -##--##- */
816:      0x3c, /* --####-- */
817:      0x00, /* ----- */
818:      0x00, /* ----- */
819:
820:      /* 0x44 ('D') */
821:      0x00, /* ----- */
822:      0xf8, /* #####--- */
823:      0x6c, /* -##-##-- */
824:      0x66, /* -##--##- */
825:      0x66, /* -##--##- */
826:      0x66, /* -##--##- */
827:      0x6c, /* -##-##-- */
828:      0xf8, /* #####--- */
829:      0x00, /* ----- */
830:      0x00, /* ----- */
831:
832:      /* 0x45 ('E') */
833:      0x00, /* ----- */
834:      0xfe, /* #####--- */
835:      0x62, /* -##---#- */
836:      0x68, /* -##-#--- */
837:      0x78, /* -####--- */
838:      0x68, /* -##-#--- */
839:      0x62, /* -##---#- */
840:      0xfe, /* #####--- */
841:      0x00, /* ----- */
842:      0x00, /* ----- */
843:
844:      /* 0x46 ('F') */
845:      0x00, /* ----- */
846:      0xfe, /* #####--- */
847:      0x62, /* -##---#- */
848:      0x68, /* -##-#--- */
849:      0x78, /* -####--- */
850:      0x68, /* -##-#--- */
851:      0x60, /* -##----- */
852:      0xf0, /* #####--- */
853:      0x00, /* ----- */
854:      0x00, /* ----- */
855:
856:      /* 0x47 ('G') */
857:      0x00, /* ----- */
858:      0x3c, /* --####-- */
859:      0x66, /* -##--##- */
860:      0xc0, /* ##----- */
861:      0xc0, /* ##----- */
862:      0xce, /* ##--##-- */
863:      0x66, /* -##--##- */
864:      0x3e, /* --#####- */
865:      0x00, /* ----- */
866:      0x00, /* ----- */
867:
868:      /* 0x48 ('H') */
869:      0x00, /* ----- */
870:      0xcc, /* ##--##-- */
871:      0xcc, /* ##--##-- */

```


drivers/video/font-lat9-8x10.c

Page 14/47

```

872:      0xcc, /* ##--##-- */
873:      0xfc, /* #####-- */
874:      0xcc, /* ##--##-- */
875:      0xcc, /* ##--##-- */
876:      0xcc, /* ##--##-- */
877:      0x00, /* ----- */
878:      0x00, /* ----- */
879:
880:          /* 0x49 ('I') */
881:      0x00, /* ----- */
882:      0x78, /* -####-- */
883:      0x30, /* --##---- */
884:      0x30, /* --##---- */
885:      0x30, /* --##---- */
886:      0x30, /* --##---- */
887:      0x30, /* --##---- */
888:      0x78, /* -####-- */
889:      0x00, /* ----- */
890:      0x00, /* ----- */
891:
892:          /* 0x4a ('J') */
893:      0x00, /* ----- */
894:      0x1e, /* ---####- */
895:      0x0c, /* ----##-- */
896:      0x0c, /* ----##-- */
897:      0x0c, /* ----##-- */
898:      0xcc, /* ##--##-- */
899:      0xcc, /* ##--##-- */
900:      0x78, /* -####-- */
901:      0x00, /* ----- */
902:      0x00, /* ----- */
903:
904:          /* 0x4b ('K') */
905:      0x00, /* ----- */
906:      0xe6, /* ###--##- */
907:      0x66, /* -##--##- */
908:      0x6c, /* -##--##- */
909:      0x78, /* -####-- */
910:      0x6c, /* -##--##- */
911:      0x66, /* -##--##- */
912:      0xe6, /* ###--##- */
913:      0x00, /* ----- */
914:      0x00, /* ----- */
915:
916:          /* 0x4c ('L') */
917:      0x00, /* ----- */
918:      0xf0, /* #####-- */
919:      0x60, /* -##----- */
920:      0x60, /* -##----- */
921:      0x60, /* -##----- */
922:      0x62, /* -##---#- */
923:      0x66, /* -##--##- */
924:      0xfe, /* #######- */
925:      0x00, /* ----- */
926:      0x00, /* ----- */
927:
928:          /* 0x4d ('M') */
929:      0x00, /* ----- */
930:      0xc6, /* ##---##- */
931:      0xee, /* ###-###- */
932:      0xfe, /* #######- */
933:      0xfe, /* #######- */
934:      0xd6, /* ##-##-##- */
935:      0xc6, /* ##---##- */
936:      0xc6, /* ##---##- */
937:      0x00, /* ----- */
938:      0x00, /* ----- */

```

drivers/video/font-lat9-8x10.c

Page 15/47

```

939:
940:          /* 0x4e ('N') */
941:    0x00,  /* ----- */
942:    0xc6,  /* ##---##- */
943:    0xe6,  /* ###--##- */
944:    0xf6,  /* ####-##- */
945:    0xde,  /* ##-####- */
946:    0xce,  /* ##---##- */
947:    0xc6,  /* ##---##- */
948:    0xc6,  /* ##---##- */
949:    0x00,  /* ----- */
950:    0x00,  /* ----- */
951:
952:          /* 0x4f ('O') */
953:    0x00,  /* ----- */
954:    0x38,  /* --###--- */
955:    0x6c,  /* -##-##- */
956:    0xc6,  /* ##---##- */
957:    0xc6,  /* ##---##- */
958:    0xc6,  /* ##---##- */
959:    0x6c,  /* -##-##- */
960:    0x38,  /* --###--- */
961:    0x00,  /* ----- */
962:    0x00,  /* ----- */
963:
964:          /* 0x50 ('P') */
965:    0x00,  /* ----- */
966:    0xfc,  /* #####-- */
967:    0x66,  /* -##-##- */
968:    0x66,  /* -##-##- */
969:    0x7c,  /* -#####-- */
970:    0x60,  /* -##----- */
971:    0x60,  /* -##----- */
972:    0xf0,  /* #####---- */
973:    0x00,  /* ----- */
974:    0x00,  /* ----- */
975:
976:          /* 0x51 ('Q') */
977:    0x00,  /* ----- */
978:    0x78,  /* -####--- */
979:    0xcc,  /* ##--##-- */
980:    0xcc,  /* ##--##-- */
981:    0xcc,  /* ##--##-- */
982:    0xdc,  /* ##-###-- */
983:    0x78,  /* -####--- */
984:    0x1c,  /* ---###-- */
985:    0x00,  /* ----- */
986:    0x00,  /* ----- */
987:
988:          /* 0x52 ('R') */
989:    0x00,  /* ----- */
990:    0xfc,  /* #####-- */
991:    0x66,  /* -##-##- */
992:    0x66,  /* -##-##- */
993:    0x7c,  /* -#####-- */
994:    0x6c,  /* -##-##-- */
995:    0x66,  /* -##-##-- */
996:    0xe6,  /* ###--##- */
997:    0x00,  /* ----- */
998:    0x00,  /* ----- */
999:
1000:          /* 0x53 ('S') */
1001:    0x00,  /* ----- */
1002:    0x78,  /* -####--- */
1003:    0xcc,  /* ##--##-- */
1004:    0xe0,  /* ###----- */
1005:    0x70,  /* -###----- */

```

drivers/video/font-lat9-8x10.c

Page 16/47

```

1006:      0x1c,    /* ---###-- */
1007:      0xcc,    /* ##---##-- */
1008:      0x78,    /* -####--- */
1009:      0x00,    /* ----- */
1010:      0x00,    /* ----- */
1011:
1012:          /* 0x54 ('T') */
1013:      0x00,    /* ----- */
1014:      0xfc,    /* #####--- */
1015:      0xb4,    /* #-##-#-- */
1016:      0x30,    /* --##---- */
1017:      0x30,    /* --##---- */
1018:      0x30,    /* --##---- */
1019:      0x30,    /* --##---- */
1020:      0x78,    /* -####--- */
1021:      0x00,    /* ----- */
1022:      0x00,    /* ----- */
1023:
1024:          /* 0x55 ('U') */
1025:      0x00,    /* ----- */
1026:      0xcc,    /* ##---##-- */
1027:      0xcc,    /* ##---##-- */
1028:      0xcc,    /* ##---##-- */
1029:      0xcc,    /* ##---##-- */
1030:      0xcc,    /* ##---##-- */
1031:      0xcc,    /* ##---##-- */
1032:      0x78,    /* -####--- */
1033:      0x00,    /* ----- */
1034:      0x00,    /* ----- */
1035:
1036:          /* 0x56 ('V') */
1037:      0x00,    /* ----- */
1038:      0xcc,    /* ##---##-- */
1039:      0xcc,    /* ##---##-- */
1040:      0xcc,    /* ##---##-- */
1041:      0xcc,    /* ##---##-- */
1042:      0xcc,    /* ##---##-- */
1043:      0x78,    /* -####--- */
1044:      0x30,    /* --##---- */
1045:      0x00,    /* ----- */
1046:      0x00,    /* ----- */
1047:
1048:          /* 0x57 ('W') */
1049:      0x00,    /* ----- */
1050:      0xc6,    /* ##---##- */
1051:      0xc6,    /* ##---##- */
1052:      0xc6,    /* ##---##- */
1053:      0xd6,    /* ##-##-##- */
1054:      0xfe,    /* #####--- */
1055:      0xee,    /* ###-###- */
1056:      0xc6,    /* ##---##- */
1057:      0x00,    /* ----- */
1058:      0x00,    /* ----- */
1059:
1060:          /* 0x58 ('X') */
1061:      0x00,    /* ----- */
1062:      0xc6,    /* ##---##- */
1063:      0xc6,    /* ##---##- */
1064:      0x6c,    /* -##-##-#- */
1065:      0x38,    /* --###---- */
1066:      0x6c,    /* -##-##-#- */
1067:      0xc6,    /* ##---##- */
1068:      0xc6,    /* ##---##- */
1069:      0x00,    /* ----- */
1070:      0x00,    /* ----- */
1071:
1072:          /* 0x59 ('Y') */

```

drivers/video/font-lat9-8x10.c

Page 17/47

```

1073:      0x00, /* ----- */
1074:      0xcc, /* ##---##-- */
1075:      0xcc, /* ##---##-- */
1076:      0xcc, /* ##---##-- */
1077:      0x78, /* -####--- */
1078:      0x30, /* --##----- */
1079:      0x30, /* --##----- */
1080:      0x78, /* -####--- */
1081:      0x00, /* ----- */
1082:      0x00, /* ----- */
1083:
1084:          /* 0x5a ('Z') */
1085:      0x00, /* ----- */
1086:      0xfe, /* #####--- */
1087:      0xc6, /* ##---##-- */
1088:      0x0c, /* ----##--- */
1089:      0x18, /* ---##---- */
1090:      0x30, /* --##----- */
1091:      0x66, /* -##-##- */
1092:      0xfe, /* #####--- */
1093:      0x00, /* ----- */
1094:      0x00, /* ----- */
1095:
1096:          /* 0x5b ('[') */
1097:      0x00, /* ----- */
1098:      0x78, /* -####--- */
1099:      0x60, /* -##----- */
1100:      0x60, /* -##----- */
1101:      0x60, /* -##----- */
1102:      0x60, /* -##----- */
1103:      0x60, /* -##----- */
1104:      0x78, /* -####--- */
1105:      0x00, /* ----- */
1106:      0x00, /* ----- */
1107:
1108:          /* 0x5c ('\') */
1109:      0x00, /* ----- */
1110:      0x00, /* ----- */
1111:      0xc0, /* ##----- */
1112:      0x60, /* -##----- */
1113:      0x30, /* --##----- */
1114:      0x18, /* ---##---- */
1115:      0x0c, /* ----##--- */
1116:      0x06, /* -----##- */
1117:      0x00, /* ----- */
1118:      0x00, /* ----- */
1119:
1120:          /* 0x5d (']') */
1121:      0x00, /* ----- */
1122:      0x78, /* -####--- */
1123:      0x18, /* ---##---- */
1124:      0x18, /* ---##---- */
1125:      0x18, /* ---##---- */
1126:      0x18, /* ---##---- */
1127:      0x18, /* ---##---- */
1128:      0x78, /* -####--- */
1129:      0x00, /* ----- */
1130:      0x00, /* ----- */
1131:
1132:          /* 0x5e ('^') */
1133:      0x10, /* ----#----- */
1134:      0x38, /* --###----- */
1135:      0x6c, /* -##-##- */
1136:      0xc6, /* ##---##-- */
1137:      0x00, /* ----- */
1138:      0x00, /* ----- */
1139:      0x00, /* ----- */

```

drivers/video/font-lat9-8x10.c

Page 18/47

```

1140:      0x00, /* ----- */
1141:      0x00, /* ----- */
1142:      0x00, /* ----- */
1143:
1144:          /* 0x5f ('_') */
1145:      0x00, /* ----- */
1146:      0x00, /* ----- */
1147:      0x00, /* ----- */
1148:      0x00, /* ----- */
1149:      0x00, /* ----- */
1150:      0x00, /* ----- */
1151:      0x00, /* ----- */
1152:      0x00, /* ----- */
1153:      0x00, /* ----- */
1154:      0xff, /* ##### */
1155:
1156:          /* 0x60 ('`') */
1157:      0x30, /* --##---- */
1158:      0x30, /* --##---- */
1159:      0x18, /* ----##---- */
1160:      0x00, /* ----- */
1161:      0x00, /* ----- */
1162:      0x00, /* ----- */
1163:      0x00, /* ----- */
1164:      0x00, /* ----- */
1165:      0x00, /* ----- */
1166:      0x00, /* ----- */
1167:
1168:          /* 0x61 ('a') */
1169:      0x00, /* ----- */
1170:      0x00, /* ----- */
1171:      0x00, /* ----- */
1172:      0x78, /* -####--- */
1173:      0x0c, /* ----##-- */
1174:      0x7c, /* -#####-- */
1175:      0xcc, /* ##--##-- */
1176:      0x76, /* -##-##- */
1177:      0x00, /* ----- */
1178:      0x00, /* ----- */
1179:
1180:          /* 0x62 ('b') */
1181:      0x00, /* ----- */
1182:      0xe0, /* ###----- */
1183:      0x60, /* -##----- */
1184:      0x60, /* -##----- */
1185:      0x7c, /* -#####-- */
1186:      0x66, /* -##-##- */
1187:      0x66, /* -##-##- */
1188:      0xdc, /* ##-##- */
1189:      0x00, /* ----- */
1190:      0x00, /* ----- */
1191:
1192:          /* 0x63 ('c') */
1193:      0x00, /* ----- */
1194:      0x00, /* ----- */
1195:      0x00, /* ----- */
1196:      0x78, /* -####--- */
1197:      0xcc, /* ##--##-- */
1198:      0xc0, /* ##----- */
1199:      0xcc, /* ##--##-- */
1200:      0x78, /* -####--- */
1201:      0x00, /* ----- */
1202:      0x00, /* ----- */
1203:
1204:          /* 0x64 ('d') */
1205:      0x00, /* ----- */
1206:      0x1c, /* ----##-- */

```

drivers/video/font-lat9-8x10.c

Page 19/47

```

1207:      0x0c,    /* ----##-- */
1208:      0x0c,    /* ----##-- */
1209:      0x7c,    /* -#####-- */
1210:      0xc0,    /* ##--##-- */
1211:      0xc0,    /* ##--##-- */
1212:      0x76,    /* -###-##- */
1213:      0x00,    /* ----- */
1214:      0x00,    /* ----- */
1215:
1216:          /* 0x65 ('e') */
1217:      0x00,    /* ----- */
1218:      0x00,    /* ----- */
1219:      0x00,    /* ----- */
1220:      0x78,    /* -####--- */
1221:      0xc0,    /* ##--##-- */
1222:      0xf0,    /* #####--- */
1223:      0xc0,    /* ##----- */
1224:      0x78,    /* -####--- */
1225:      0x00,    /* ----- */
1226:      0x00,    /* ----- */
1227:
1228:          /* 0x66 ('f') */
1229:      0x00,    /* ----- */
1230:      0x38,    /* --###--- */
1231:      0x6c,    /* -##-##-- */
1232:      0x60,    /* -##----- */
1233:      0xf0,    /* #####--- */
1234:      0x60,    /* -##----- */
1235:      0x60,    /* -##----- */
1236:      0xf0,    /* #####--- */
1237:      0x00,    /* ----- */
1238:      0x00,    /* ----- */
1239:
1240:          /* 0x67 ('g') */
1241:      0x00,    /* ----- */
1242:      0x00,    /* ----- */
1243:      0x00,    /* ----- */
1244:      0x76,    /* -###-##- */
1245:      0xc0,    /* ##--##-- */
1246:      0xc0,    /* ##--##-- */
1247:      0x7c,    /* -#####-- */
1248:      0x0c,    /* ----##-- */
1249:      0xf8,    /* #####--- */
1250:      0x00,    /* ----- */
1251:
1252:          /* 0x68 ('h') */
1253:      0x00,    /* ----- */
1254:      0xe0,    /* ###----- */
1255:      0x60,    /* -##----- */
1256:      0x6c,    /* -##-##-- */
1257:      0x76,    /* -###-##- */
1258:      0x66,    /* -##-##-- */
1259:      0x66,    /* -##-##-- */
1260:      0xe6,    /* ###-##-- */
1261:      0x00,    /* ----- */
1262:      0x00,    /* ----- */
1263:
1264:          /* 0x69 ('i') */
1265:      0x00,    /* ----- */
1266:      0x30,    /* --##----- */
1267:      0x00,    /* ----- */
1268:      0x70,    /* -###----- */
1269:      0x30,    /* --##----- */
1270:      0x30,    /* --##----- */
1271:      0x30,    /* --##----- */
1272:      0x78,    /* -####--- */
1273:      0x00,    /* ----- */

```

drivers/video/font-lat9-8x10.c

Page 20/47

```

1274:      0x00,    /* ----- */
1275:
1276:      /* 0x6a ('j') */
1277:      0x00,    /* ----- */
1278:      0x0c,    /* ----##-- */
1279:      0x00,    /* ----- */
1280:      0x0c,    /* ----##-- */
1281:      0x0c,    /* ----##-- */
1282:      0x0c,    /* ----##-- */
1283:      0xcc,    /* ##--##-- */
1284:      0xcc,    /* ##--##-- */
1285:      0x78,    /* -####--- */
1286:      0x00,    /* ----- */
1287:
1288:      /* 0x6b ('k') */
1289:      0x00,    /* ----- */
1290:      0xe0,    /* ###----- */
1291:      0x60,    /* -##----- */
1292:      0x66,    /* -##-##- */
1293:      0x6c,    /* -##-##- */
1294:      0x78,    /* -####--- */
1295:      0x6c,    /* -##-##- */
1296:      0xe6,    /* ###-##- */
1297:      0x00,    /* ----- */
1298:      0x00,    /* ----- */
1299:
1300:      /* 0x6c ('l') */
1301:      0x00,    /* ----- */
1302:      0x70,    /* -###----- */
1303:      0x30,    /* --##----- */
1304:      0x30,    /* --##----- */
1305:      0x30,    /* --##----- */
1306:      0x30,    /* --##----- */
1307:      0x30,    /* --##----- */
1308:      0x78,    /* -####--- */
1309:      0x00,    /* ----- */
1310:      0x00,    /* ----- */
1311:
1312:      /* 0x6d ('m') */
1313:      0x00,    /* ----- */
1314:      0x00,    /* ----- */
1315:      0x00,    /* ----- */
1316:      0xcc,    /* ##--##-- */
1317:      0xfe,    /* #####--- */
1318:      0xfe,    /* #####--- */
1319:      0xd6,    /* #-#-##- */
1320:      0xc6,    /* #-#-##- */
1321:      0x00,    /* ----- */
1322:      0x00,    /* ----- */
1323:
1324:      /* 0x6e ('n') */
1325:      0x00,    /* ----- */
1326:      0x00,    /* ----- */
1327:      0x00,    /* ----- */
1328:      0xf8,    /* #####--- */
1329:      0xcc,    /* ##--##-- */
1330:      0xcc,    /* ##--##-- */
1331:      0xcc,    /* ##--##-- */
1332:      0xcc,    /* ##--##-- */
1333:      0x00,    /* ----- */
1334:      0x00,    /* ----- */
1335:
1336:      /* 0x6f ('o') */
1337:      0x00,    /* ----- */
1338:      0x00,    /* ----- */
1339:      0x00,    /* ----- */
1340:      0x78,    /* -####--- */

```

drivers/video/font-lat9-8x10.c

Page 21/47

```

1341:      0xcc, /* ##--##-- */
1342:      0xcc, /* ##--##-- */
1343:      0xcc, /* ##--##-- */
1344:      0x78, /* -####--- */
1345:      0x00, /* ----- */
1346:      0x00, /* ----- */
1347:
1348:          /* 0x70 ('p') */
1349:      0x00, /* ----- */
1350:      0x00, /* ----- */
1351:      0x00, /* ----- */
1352:      0xdc, /* #-###-- */
1353:      0x66, /* -##--##- */
1354:      0x66, /* -##--##- */
1355:      0x7c, /* -#####-- */
1356:      0x60, /* -##----- */
1357:      0xf0, /* ####----- */
1358:      0x00, /* ----- */
1359:
1360:          /* 0x71 ('q') */
1361:      0x00, /* ----- */
1362:      0x00, /* ----- */
1363:      0x00, /* ----- */
1364:      0x76, /* -###-##- */
1365:      0xcc, /* ##--##-- */
1366:      0xcc, /* ##--##-- */
1367:      0x7c, /* -#####-- */
1368:      0x0c, /* ----##-- */
1369:      0x1e, /* ---####- */
1370:      0x00, /* ----- */
1371:
1372:          /* 0x72 ('r') */
1373:      0x00, /* ----- */
1374:      0x00, /* ----- */
1375:      0x00, /* ----- */
1376:      0xdc, /* #-###-- */
1377:      0x76, /* -###-##- */
1378:      0x66, /* -##--##- */
1379:      0x60, /* -##----- */
1380:      0xf0, /* ####----- */
1381:      0x00, /* ----- */
1382:      0x00, /* ----- */
1383:
1384:          /* 0x73 ('s') */
1385:      0x00, /* ----- */
1386:      0x00, /* ----- */
1387:      0x00, /* ----- */
1388:      0x7c, /* -#####-- */
1389:      0xc0, /* ##----- */
1390:      0x78, /* -####--- */
1391:      0x0c, /* ----##-- */
1392:      0xf8, /* #####----- */
1393:      0x00, /* ----- */
1394:      0x00, /* ----- */
1395:
1396:          /* 0x74 ('t') */
1397:      0x00, /* ----- */
1398:      0x10, /* ---#----- */
1399:      0x30, /* --##----- */
1400:      0x7c, /* -#####-- */
1401:      0x30, /* --##----- */
1402:      0x30, /* --##----- */
1403:      0x34, /* --##-#----- */
1404:      0x18, /* ---##----- */
1405:      0x00, /* ----- */
1406:      0x00, /* ----- */
1407:

```


drivers/video/font-lat9-8x10.c

Page 22/47

```

1408:                /* 0x75 ('u') */
1409:                0x00, /* ----- */
1410:                0x00, /* ----- */
1411:                0x00, /* ----- */
1412:                0xcc, /* ##---##- */
1413:                0xcc, /* ##---##- */
1414:                0xcc, /* ##---##- */
1415:                0xcc, /* ##---##- */
1416:                0x76, /* -###-##- */
1417:                0x00, /* ----- */
1418:                0x00, /* ----- */
1419:
1420:                /* 0x76 ('v') */
1421:                0x00, /* ----- */
1422:                0x00, /* ----- */
1423:                0x00, /* ----- */
1424:                0xcc, /* ##---##- */
1425:                0xcc, /* ##---##- */
1426:                0xcc, /* ##---##- */
1427:                0x78, /* -####- */
1428:                0x30, /* --##---- */
1429:                0x00, /* ----- */
1430:                0x00, /* ----- */
1431:
1432:                /* 0x77 ('w') */
1433:                0x00, /* ----- */
1434:                0x00, /* ----- */
1435:                0x00, /* ----- */
1436:                0xc6, /* ##---##- */
1437:                0xd6, /* ##-##-##- */
1438:                0xfe, /* #####- */
1439:                0xfe, /* #####- */
1440:                0x6c, /* -##-##- */
1441:                0x00, /* ----- */
1442:                0x00, /* ----- */
1443:
1444:                /* 0x78 ('x') */
1445:                0x00, /* ----- */
1446:                0x00, /* ----- */
1447:                0x00, /* ----- */
1448:                0xc6, /* ##---##- */
1449:                0x6c, /* -##-##- */
1450:                0x38, /* --###---- */
1451:                0x6c, /* -##-##- */
1452:                0xc6, /* ##---##- */
1453:                0x00, /* ----- */
1454:                0x00, /* ----- */
1455:
1456:                /* 0x79 ('y') */
1457:                0x00, /* ----- */
1458:                0x00, /* ----- */
1459:                0x00, /* ----- */
1460:                0xcc, /* ##---##- */
1461:                0xcc, /* ##---##- */
1462:                0xcc, /* ##---##- */
1463:                0x7c, /* -#####- */
1464:                0x0c, /* ----##- */
1465:                0xf8, /* #####- */
1466:                0x00, /* ----- */
1467:
1468:                /* 0x7a ('z') */
1469:                0x00, /* ----- */
1470:                0x00, /* ----- */
1471:                0x00, /* ----- */
1472:                0xfc, /* #####- */
1473:                0x98, /* #-##---- */
1474:                0x30, /* --##---- */

```

drivers/video/font-lat9-8x10.c

Page 23/47

```

1475:      0x64,      /* ---#---#--- */
1476:      0xfc,      /* #####--- */
1477:      0x00,      /* ----- */
1478:      0x00,      /* ----- */
1479:
1480:          /* 0x7b ('{') */
1481:      0x00,      /* ----- */
1482:      0x1c,      /* ---###--- */
1483:      0x30,      /* --##----- */
1484:      0x30,      /* --##----- */
1485:      0xe0,      /* ###----- */
1486:      0x30,      /* --##----- */
1487:      0x30,      /* --##----- */
1488:      0x1c,      /* ---###--- */
1489:      0x00,      /* ----- */
1490:      0x00,      /* ----- */
1491:
1492:          /* 0x7c ('|') */
1493:      0x00,      /* ----- */
1494:      0x30,      /* --##----- */
1495:      0x30,      /* --##----- */
1496:      0x30,      /* --##----- */
1497:      0x30,      /* --##----- */
1498:      0x30,      /* --##----- */
1499:      0x30,      /* --##----- */
1500:      0x30,      /* --##----- */
1501:      0x00,      /* ----- */
1502:      0x00,      /* ----- */
1503:
1504:          /* 0x7d ('}') */
1505:      0x00,      /* ----- */
1506:      0xe0,      /* ###----- */
1507:      0x30,      /* --##----- */
1508:      0x30,      /* --##----- */
1509:      0x1c,      /* ---###--- */
1510:      0x30,      /* --##----- */
1511:      0x30,      /* --##----- */
1512:      0xe0,      /* ###----- */
1513:      0x00,      /* ----- */
1514:      0x00,      /* ----- */
1515:
1516:          /* 0x7e ('~') */
1517:      0x76,      /* -###-##- */
1518:      0xdc,      /* ##-###--- */
1519:      0x00,      /* ----- */
1520:      0x00,      /* ----- */
1521:      0x00,      /* ----- */
1522:      0x00,      /* ----- */
1523:      0x00,      /* ----- */
1524:      0x00,      /* ----- */
1525:      0x00,      /* ----- */
1526:      0x00,      /* ----- */
1527:
1528:          /* 0x7f */
1529:      0xcc,      /* ##---##- */
1530:      0xcc,      /* ##---##- */
1531:      0x00,      /* ----- */
1532:      0xcc,      /* ##---##- */
1533:      0xcc,      /* ##---##- */
1534:      0x78,      /* -###--- */
1535:      0x30,      /* --##----- */
1536:      0x78,      /* -###--- */
1537:      0x00,      /* ----- */
1538:      0x00,      /* ----- */
1539:
1540:          /* 0x80 */
1541:      0x00,      /* ----- */

```

drivers/video/font-lat9-8x10.c Page 24/47

```

1542:      0x00, /* ----- */
1543:      0xff, /* ##### */
1544:      0x00, /* ----- */
1545:      0x00, /* ----- */
1546:      0x00, /* ----- */
1547:      0x00, /* ----- */
1548:      0x00, /* ----- */
1549:      0x00, /* ----- */
1550:      0x00, /* ----- */
1551:
1552:      /* 0x81 */
1553:      0x18, /* ---#--- */
1554:      0x18, /* ---#--- */
1555:      0x18, /* ---#--- */
1556:      0x18, /* ---#--- */
1557:      0x18, /* ---#--- */
1558:      0x00, /* ----- */
1559:      0x00, /* ----- */
1560:      0x00, /* ----- */
1561:      0x00, /* ----- */
1562:      0x00, /* ----- */
1563:
1564:      /* 0x82 */
1565:      0x00, /* ----- */
1566:      0x00, /* ----- */
1567:      0x00, /* ----- */
1568:      0x00, /* ----- */
1569:      0x1f, /* ----##### */
1570:      0x00, /* ----- */
1571:      0x00, /* ----- */
1572:      0x00, /* ----- */
1573:      0x00, /* ----- */
1574:      0x00, /* ----- */
1575:
1576:      /* 0x83 */
1577:      0x18, /* ---#--- */
1578:      0x18, /* ---#--- */
1579:      0x18, /* ---#--- */
1580:      0x18, /* ---#--- */
1581:      0x1f, /* ----##### */
1582:      0x00, /* ----- */
1583:      0x00, /* ----- */
1584:      0x00, /* ----- */
1585:      0x00, /* ----- */
1586:      0x00, /* ----- */
1587:
1588:      /* 0x84 */
1589:      0x00, /* ----- */
1590:      0x00, /* ----- */
1591:      0x00, /* ----- */
1592:      0x00, /* ----- */
1593:      0x00, /* ----- */
1594:      0x18, /* ---#--- */
1595:      0x18, /* ---#--- */
1596:      0x18, /* ---#--- */
1597:      0x18, /* ---#--- */
1598:      0x18, /* ---#--- */
1599:
1600:      /* 0x85 */
1601:      0x18, /* ---#--- */
1602:      0x18, /* ---#--- */
1603:      0x18, /* ---#--- */
1604:      0x18, /* ---#--- */
1605:      0x18, /* ---#--- */
1606:      0x18, /* ---#--- */
1607:      0x18, /* ---#--- */
1608:      0x18, /* ---#--- */

```

drivers/video/font-lat9-8x10.c

Page 25/47

```

1609:      0x18, /* ---##--- */
1610:      0x18, /* ---##--- */
1611:
1612:          /* 0x86 */
1613:      0x00, /* ----- */
1614:      0x00, /* ----- */
1615:      0x00, /* ----- */
1616:      0x00, /* ----- */
1617:      0x1f, /* ---##### */
1618:      0x18, /* ---##--- */
1619:      0x18, /* ---##--- */
1620:      0x18, /* ---##--- */
1621:      0x18, /* ---##--- */
1622:      0x18, /* ---##--- */
1623:
1624:          /* 0x87 */
1625:      0x18, /* ---##--- */
1626:      0x18, /* ---##--- */
1627:      0x18, /* ---##--- */
1628:      0x18, /* ---##--- */
1629:      0x1f, /* ---##### */
1630:      0x18, /* ---##--- */
1631:      0x18, /* ---##--- */
1632:      0x18, /* ---##--- */
1633:      0x18, /* ---##--- */
1634:      0x18, /* ---##--- */
1635:
1636:          /* 0x88 */
1637:      0x00, /* ----- */
1638:      0x00, /* ----- */
1639:      0x00, /* ----- */
1640:      0x00, /* ----- */
1641:      0xf8, /* #####--- */
1642:      0x00, /* ----- */
1643:      0x00, /* ----- */
1644:      0x00, /* ----- */
1645:      0x00, /* ----- */
1646:      0x00, /* ----- */
1647:
1648:          /* 0x89 */
1649:      0x18, /* ---##--- */
1650:      0x18, /* ---##--- */
1651:      0x18, /* ---##--- */
1652:      0x18, /* ---##--- */
1653:      0xf8, /* #####--- */
1654:      0x00, /* ----- */
1655:      0x00, /* ----- */
1656:      0x00, /* ----- */
1657:      0x00, /* ----- */
1658:      0x00, /* ----- */
1659:
1660:          /* 0x8a */
1661:      0x00, /* ----- */
1662:      0x00, /* ----- */
1663:      0x00, /* ----- */
1664:      0x00, /* ----- */
1665:      0xff, /* #####--- */
1666:      0x00, /* ----- */
1667:      0x00, /* ----- */
1668:      0x00, /* ----- */
1669:      0x00, /* ----- */
1670:      0x00, /* ----- */
1671:
1672:          /* 0x8b */
1673:      0x18, /* ---##--- */
1674:      0x18, /* ---##--- */
1675:      0x18, /* ---##--- */

```

drivers/video/font-lat9-8x10.c

Page 26/47

```

1676:      0x18,    /* ----##---- */
1677:      0xff,    /* ##### */
1678:      0x00,    /* ----- */
1679:      0x00,    /* ----- */
1680:      0x00,    /* ----- */
1681:      0x00,    /* ----- */
1682:      0x00,    /* ----- */
1683:
1684:          /* 0x8c */
1685:      0x00,    /* ----- */
1686:      0x00,    /* ----- */
1687:      0x00,    /* ----- */
1688:      0x00,    /* ----- */
1689:      0xf8,    /* #####---- */
1690:      0x18,    /* ----##---- */
1691:      0x18,    /* ----##---- */
1692:      0x18,    /* ----##---- */
1693:      0x18,    /* ----##---- */
1694:      0x18,    /* ----##---- */
1695:
1696:          /* 0x8d */
1697:      0x18,    /* ----##---- */
1698:      0x18,    /* ----##---- */
1699:      0x18,    /* ----##---- */
1700:      0x18,    /* ----##---- */
1701:      0xf8,    /* #####---- */
1702:      0x18,    /* ----##---- */
1703:      0x18,    /* ----##---- */
1704:      0x18,    /* ----##---- */
1705:      0x18,    /* ----##---- */
1706:      0x18,    /* ----##---- */
1707:
1708:          /* 0x8e */
1709:      0x00,    /* ----- */
1710:      0x00,    /* ----- */
1711:      0x00,    /* ----- */
1712:      0x00,    /* ----- */
1713:      0xff,    /* ##### */
1714:      0x18,    /* ----##---- */
1715:      0x18,    /* ----##---- */
1716:      0x18,    /* ----##---- */
1717:      0x18,    /* ----##---- */
1718:      0x18,    /* ----##---- */
1719:
1720:          /* 0x8f */
1721:      0x18,    /* ----##---- */
1722:      0x18,    /* ----##---- */
1723:      0x18,    /* ----##---- */
1724:      0x18,    /* ----##---- */
1725:      0xff,    /* ##### */
1726:      0x18,    /* ----##---- */
1727:      0x18,    /* ----##---- */
1728:      0x18,    /* ----##---- */
1729:      0x18,    /* ----##---- */
1730:      0x18,    /* ----##---- */
1731:
1732:          /* 0x90 */
1733:      0x00,    /* ----- */
1734:      0x00,    /* ----- */
1735:      0x00,    /* ----- */
1736:      0x00,    /* ----- */
1737:      0x00,    /* ----- */
1738:      0x00,    /* ----- */
1739:      0xff,    /* ##### */
1740:      0x00,    /* ----- */
1741:      0x00,    /* ----- */
1742:      0x00,    /* ----- */

```

drivers/video/font-lat9-8x10.c

Page 27/47

```

1743:
1744:          /* 0x91          */
1745:    0x6c, /* -##-##-- */
1746:    0x6c, /* -##-##-- */
1747:    0x6c, /* -##-##-- */
1748:    0x6c, /* -##-##-- */
1749:    0x6c, /* -##-##-- */
1750:    0x7c, /* -#####-- */
1751:    0x00, /* ----- */
1752:    0x00, /* ----- */
1753:    0x00, /* ----- */
1754:    0x00, /* ----- */
1755:
1756:          /* 0x92          */
1757:    0x00, /* ----- */
1758:    0x00, /* ----- */
1759:    0x00, /* ----- */
1760:    0x7f, /* -##### */
1761:    0x60, /* -##----- */
1762:    0x7f, /* -##### */
1763:    0x00, /* ----- */
1764:    0x00, /* ----- */
1765:    0x00, /* ----- */
1766:    0x00, /* ----- */
1767:
1768:          /* 0x93          */
1769:    0x6c, /* -##-##-- */
1770:    0x6c, /* -##-##-- */
1771:    0x6c, /* -##-##-- */
1772:    0x6f, /* -##-#### */
1773:    0x60, /* -##----- */
1774:    0x7f, /* -##### */
1775:    0x00, /* ----- */
1776:    0x00, /* ----- */
1777:    0x00, /* ----- */
1778:    0x00, /* ----- */
1779:
1780:          /* 0x94          */
1781:    0x00, /* ----- */
1782:    0x00, /* ----- */
1783:    0x00, /* ----- */
1784:    0x7c, /* -#####-- */
1785:    0x6c, /* -##-##-- */
1786:    0x6c, /* -##-##-- */
1787:    0x6c, /* -##-##-- */
1788:    0x6c, /* -##-##-- */
1789:    0x6c, /* -##-##-- */
1790:    0x6c, /* -##-##-- */
1791:
1792:          /* 0x95          */
1793:    0x6c, /* -##-##-- */
1794:    0x6c, /* -##-##-- */
1795:    0x6c, /* -##-##-- */
1796:    0x6c, /* -##-##-- */
1797:    0x6c, /* -##-##-- */
1798:    0x6c, /* -##-##-- */
1799:    0x6c, /* -##-##-- */
1800:    0x6c, /* -##-##-- */
1801:    0x6c, /* -##-##-- */
1802:    0x6c, /* -##-##-- */
1803:
1804:          /* 0x96          */
1805:    0x00, /* ----- */
1806:    0x00, /* ----- */
1807:    0x00, /* ----- */
1808:    0x7f, /* -##### */
1809:    0x60, /* -##----- */

```

drivers/video/font-lat9-8x10.c

Page 28/47

```

1810:      0x6f,    /* ---### */
1811:      0x6c,    /* ---### */
1812:      0x6c,    /* ---### */
1813:      0x6c,    /* ---### */
1814:      0x6c,    /* ---### */
1815:
1816:          /* 0x97 */
1817:      0x6c,    /* ---### */
1818:      0x6c,    /* ---### */
1819:      0x6c,    /* ---### */
1820:      0x6f,    /* ---### */
1821:      0x60,    /* --- */
1822:      0x6f,    /* ---### */
1823:      0x6c,    /* ---### */
1824:      0x6c,    /* ---### */
1825:      0x6c,    /* ---### */
1826:      0x6c,    /* ---### */
1827:
1828:          /* 0x98 */
1829:      0x00,    /* ----- */
1830:      0x00,    /* ----- */
1831:      0x00,    /* ----- */
1832:      0xfc,    /* #####-- */
1833:      0x0c,    /* ----##-- */
1834:      0xfc,    /* #####-- */
1835:      0x00,    /* ----- */
1836:      0x00,    /* ----- */
1837:      0x00,    /* ----- */
1838:      0x00,    /* ----- */
1839:
1840:          /* 0x99 */
1841:      0x6c,    /* ---### */
1842:      0x6c,    /* ---### */
1843:      0x6c,    /* ---### */
1844:      0xec,    /* ###-##-- */
1845:      0x0c,    /* ----##-- */
1846:      0xfc,    /* #####-- */
1847:      0x00,    /* ----- */
1848:      0x00,    /* ----- */
1849:      0x00,    /* ----- */
1850:      0x00,    /* ----- */
1851:
1852:          /* 0x9a */
1853:      0x00,    /* ----- */
1854:      0x00,    /* ----- */
1855:      0x00,    /* ----- */
1856:      0xff,    /* ##### */
1857:      0x00,    /* ----- */
1858:      0xff,    /* ##### */
1859:      0x00,    /* ----- */
1860:      0x00,    /* ----- */
1861:      0x00,    /* ----- */
1862:      0x00,    /* ----- */
1863:
1864:          /* 0x9b */
1865:      0x6c,    /* ---### */
1866:      0x6c,    /* ---### */
1867:      0x6c,    /* ---### */
1868:      0xef,    /* ###-### */
1869:      0x00,    /* ----- */
1870:      0xff,    /* ##### */
1871:      0x00,    /* ----- */
1872:      0x00,    /* ----- */
1873:      0x00,    /* ----- */
1874:      0x00,    /* ----- */
1875:
1876:          /* 0x9c */

```

drivers/video/font-lat9-8x10.c

Page 29/47

```

1877:      0x00,    /* ----- */
1878:      0x00,    /* ----- */
1879:      0x00,    /* ----- */
1880:      0xfc,    /* #####- */
1881:      0x0c,    /* ----##- */
1882:      0xec,    /* ###-##- */
1883:      0x6c,    /* -##-##- */
1884:      0x6c,    /* -##-##- */
1885:      0x6c,    /* -##-##- */
1886:      0x6c,    /* -##-##- */
1887:
1888:          /* 0x9d      */
1889:      0x6c,    /* -##-##- */
1890:      0x6c,    /* -##-##- */
1891:      0x6c,    /* -##-##- */
1892:      0xec,    /* ###-##- */
1893:      0x0c,    /* ----##- */
1894:      0xec,    /* ###-##- */
1895:      0x6c,    /* -##-##- */
1896:      0x6c,    /* -##-##- */
1897:      0x6c,    /* -##-##- */
1898:      0x6c,    /* -##-##- */
1899:
1900:          /* 0x9e      */
1901:      0x00,    /* ----- */
1902:      0x00,    /* ----- */
1903:      0x00,    /* ----- */
1904:      0xff,    /* ##### */
1905:      0x00,    /* ----- */
1906:      0xef,    /* ###-### */
1907:      0x6c,    /* -##-##- */
1908:      0x6c,    /* -##-##- */
1909:      0x6c,    /* -##-##- */
1910:      0x6c,    /* -##-##- */
1911:
1912:          /* 0x9f      */
1913:      0x6c,    /* -##-##- */
1914:      0x6c,    /* -##-##- */
1915:      0x6c,    /* -##-##- */
1916:      0xef,    /* ###-### */
1917:      0x00,    /* ----- */
1918:      0xef,    /* ###-### */
1919:      0x6c,    /* -##-##- */
1920:      0x6c,    /* -##-##- */
1921:      0x6c,    /* -##-##- */
1922:      0x6c,    /* -##-##- */
1923:
1924:          /* 0xa0      */
1925:      0x00,    /* ----- */
1926:      0x00,    /* ----- */
1927:      0x00,    /* ----- */
1928:      0x00,    /* ----- */
1929:      0x00,    /* ----- */
1930:      0x00,    /* ----- */
1931:      0x00,    /* ----- */
1932:      0xc6,    /* ##---##- */
1933:      0xfe,    /* #####- */
1934:      0x00,    /* ----- */
1935:
1936:          /* 0xa1      */
1937:      0x00,    /* ----- */
1938:      0x00,    /* ----- */
1939:      0x00,    /* ----- */
1940:      0x30,    /* ---##- */
1941:      0x00,    /* ----- */
1942:      0x30,    /* ---##- */
1943:      0x30,    /* ---##- */

```


drivers/video/font-lat9-8x10.c

Page 30/47

```

1944:      0x78, /* -###- --- */
1945:      0x78, /* -###- --- */
1946:      0x30, /* --##- --- */
1947:
1948:          /* 0xa2 */
1949:      0x00, /* ----- */
1950:      0x00, /* ----- */
1951:      0x30, /* --##- --- */
1952:      0x78, /* -###- --- */
1953:      0xcc, /* ##-##- --- */
1954:      0xc0, /* ##- --- */
1955:      0xcc, /* ##-##- --- */
1956:      0x78, /* -###- --- */
1957:      0x30, /* --##- --- */
1958:      0x00, /* ----- */
1959:
1960:          /* 0xa3 */
1961:      0x00, /* ----- */
1962:      0x38, /* --###- --- */
1963:      0x6c, /* -##-##- --- */
1964:      0x64, /* -##-##- --- */
1965:      0xf0, /* ####- --- */
1966:      0x60, /* -##- --- */
1967:      0xe6, /* ###-##- --- */
1968:      0xfc, /* #####- --- */
1969:      0x00, /* ----- */
1970:      0x00, /* ----- */
1971:
1972:          /* 0xa4 */
1973:      0x00, /* ----- */
1974:      0x38, /* --###- --- */
1975:      0x64, /* -##-##- --- */
1976:      0xf0, /* ####- --- */
1977:      0x60, /* -##- --- */
1978:      0xf0, /* ####- --- */
1979:      0x64, /* -##-##- --- */
1980:      0x38, /* --###- --- */
1981:      0x00, /* ----- */
1982:      0x00, /* ----- */
1983:
1984:          /* 0xa5 */
1985:      0x00, /* ----- */
1986:      0xcc, /* ##-##- --- */
1987:      0xcc, /* ##-##- --- */
1988:      0x78, /* -###- --- */
1989:      0xfc, /* #####- --- */
1990:      0x30, /* --##- --- */
1991:      0xfc, /* #####- --- */
1992:      0x30, /* --##- --- */
1993:      0x30, /* --##- --- */
1994:      0x00, /* ----- */
1995:
1996:          /* 0xa6 */
1997:      0x78, /* -###- --- */
1998:      0x00, /* ----- */
1999:      0x78, /* -###- --- */
2000:      0xc4, /* ##-##- --- */
2001:      0x60, /* -##- --- */
2002:      0x38, /* --###- --- */
2003:      0x8c, /* #---##- --- */
2004:      0x78, /* -###- --- */
2005:      0x00, /* ----- */
2006:      0x00, /* ----- */
2007:
2008:          /* 0xa7 */
2009:      0x38, /* --###- --- */
2010:      0x6c, /* -##-##- --- */

```

drivers/video/font-lat9-8x10.c

Page 31/47

```

2011:      0x30,    /* --##----- */
2012:      0x38,    /* ---###---- */
2013:      0x6c,    /* -##-##-- */
2014:      0x6c,    /* -##-##-- */
2015:      0x38,    /* ---###---- */
2016:      0x18,    /* ----##---- */
2017:      0x6c,    /* -##-##-- */
2018:      0x38,    /* ---###---- */
2019:
2020:      /* 0xa8          */
2021:      0xcc,    /* ##--##-- */
2022:      0x78,    /* -####---- */
2023:      0x00,    /* ----- */
2024:      0x7c,    /* -#####-- */
2025:      0xc0,    /* ##----- */
2026:      0x78,    /* -####---- */
2027:      0x0c,    /* -----##-- */
2028:      0xf8,    /* #####---- */
2029:      0x00,    /* ----- */
2030:      0x00,    /* ----- */
2031:
2032:      /* 0xa9          */
2033:      0x7c,    /* -#####-- */
2034:      0x82,    /* #-----#- */
2035:      0x9a,    /* #--##-#- */
2036:      0xa2,    /* #-#----#- */
2037:      0xa2,    /* #-#----#- */
2038:      0x9a,    /* #--##-#- */
2039:      0x82,    /* #-----#- */
2040:      0x7c,    /* -#####-- */
2041:      0x00,    /* ----- */
2042:      0x00,    /* ----- */
2043:
2044:      /* 0xaa          */
2045:      0x3c,    /* --####-- */
2046:      0x6c,    /* -##-##-- */
2047:      0x6c,    /* -##-##-- */
2048:      0x3e,    /* --#####- */
2049:      0x00,    /* ----- */
2050:      0x7e,    /* -#####-- */
2051:      0x00,    /* ----- */
2052:      0x00,    /* ----- */
2053:      0x00,    /* ----- */
2054:      0x00,    /* ----- */
2055:
2056:      /* 0xab          */
2057:      0x00,    /* ----- */
2058:      0x00,    /* ----- */
2059:      0x33,    /* --##--## */
2060:      0x66,    /* -##--##- */
2061:      0xcc,    /* ##--##-- */
2062:      0x66,    /* -##--##- */
2063:      0x33,    /* --##--## */
2064:      0x00,    /* ----- */
2065:      0x00,    /* ----- */
2066:      0x00,    /* ----- */
2067:
2068:      /* 0xac          */
2069:      0x00,    /* ----- */
2070:      0x00,    /* ----- */
2071:      0x00,    /* ----- */
2072:      0x00,    /* ----- */
2073:      0xfc,    /* #####-- */
2074:      0x0c,    /* -----##-- */
2075:      0x0c,    /* -----##-- */
2076:      0x00,    /* ----- */
2077:      0x00,    /* ----- */

```

drivers/video/font-lat9-8x10.c

Page 32/47

```

2078:      0x00, /* ----- */
2079:
2080:      /* 0xad */
2081:      0x00, /* ----- */
2082:      0x00, /* ----- */
2083:      0x00, /* ----- */
2084:      0x00, /* ----- */
2085:      0x7c, /* #####-- */
2086:      0x00, /* ----- */
2087:      0x00, /* ----- */
2088:      0x00, /* ----- */
2089:      0x00, /* ----- */
2090:      0x00, /* ----- */
2091:
2092:      /* 0xae */
2093:      0x7c, /* #####-- */
2094:      0x82, /* #-----#- */
2095:      0xb2, /* #-##--#- */
2096:      0xaa, /* #-#-#-#- */
2097:      0xb2, /* #-##--#- */
2098:      0xaa, /* #-#-#-#- */
2099:      0x82, /* #-----#- */
2100:      0x7c, /* #####-- */
2101:      0x00, /* ----- */
2102:      0x00, /* ----- */
2103:
2104:      /* 0xaf */
2105:      0xff, /* ##### */
2106:      0x00, /* ----- */
2107:      0x00, /* ----- */
2108:      0x00, /* ----- */
2109:      0x00, /* ----- */
2110:      0x00, /* ----- */
2111:      0x00, /* ----- */
2112:      0x00, /* ----- */
2113:      0x00, /* ----- */
2114:      0x00, /* ----- */
2115:
2116:      /* 0xb0 */
2117:      0x70, /* -###----- */
2118:      0xd8, /* ##-##---- */
2119:      0x70, /* -###----- */
2120:      0x00, /* ----- */
2121:      0x00, /* ----- */
2122:      0x00, /* ----- */
2123:      0x00, /* ----- */
2124:      0x00, /* ----- */
2125:      0x00, /* ----- */
2126:      0x00, /* ----- */
2127:
2128:      /* 0xb1 */
2129:      0x00, /* ----- */
2130:      0x30, /* --##----- */
2131:      0x30, /* --##----- */
2132:      0xfc, /* #####-- */
2133:      0x30, /* --##----- */
2134:      0x30, /* --##----- */
2135:      0x00, /* ----- */
2136:      0xfc, /* #####-- */
2137:      0x00, /* ----- */
2138:      0x00, /* ----- */
2139:
2140:      /* 0xb2 */
2141:      0x70, /* -###----- */
2142:      0xd8, /* ##-##---- */
2143:      0x30, /* --##----- */
2144:      0x60, /* -##----- */

```

drivers/video/font-lat9-8x10.c

Page 33/47

```

2145:      0xf8,    /* #####--- */
2146:      0x00,    /* ----- */
2147:      0x00,    /* ----- */
2148:      0x00,    /* ----- */
2149:      0x00,    /* ----- */
2150:      0x00,    /* ----- */
2151:
2152:          /* 0xb3 */
2153:      0x70,    /* -###----- */
2154:      0xd8,    /* #-##----- */
2155:      0x30,    /* --##----- */
2156:      0xd8,    /* #-##----- */
2157:      0x70,    /* -###----- */
2158:      0x00,    /* ----- */
2159:      0x00,    /* ----- */
2160:      0x00,    /* ----- */
2161:      0x00,    /* ----- */
2162:      0x00,    /* ----- */
2163:
2164:          /* 0xb4 */
2165:      0x7c,    /* -#####-- */
2166:      0x00,    /* ----- */
2167:      0xfe,    /* #####--- */
2168:      0xcc,    /* #-##----- */
2169:      0x18,    /* ---##----- */
2170:      0x30,    /* --##----- */
2171:      0x66,    /* -##-##----- */
2172:      0xfe,    /* #####--- */
2173:      0x00,    /* ----- */
2174:      0x00,    /* ----- */
2175:
2176:          /* 0xb5 */
2177:      0x00,    /* ----- */
2178:      0x00,    /* ----- */
2179:      0x00,    /* ----- */
2180:      0xcc,    /* #-##----- */
2181:      0xcc,    /* #-##----- */
2182:      0xcc,    /* #-##----- */
2183:      0xcc,    /* #-##----- */
2184:      0xf6,    /* #####-##- */
2185:      0xc0,    /* #-##----- */
2186:      0xc0,    /* #-##----- */
2187:
2188:          /* 0xb6 */
2189:      0x00,    /* ----- */
2190:      0x7f,    /* -##### */
2191:      0xdb,    /* #-##-##- */
2192:      0x7b,    /* -####-##- */
2193:      0x3b,    /* --##-##- */
2194:      0x1b,    /* ---##-##- */
2195:      0x1b,    /* ---##-##- */
2196:      0x1b,    /* ---##-##- */
2197:      0x1b,    /* ---##-##- */
2198:      0x00,    /* ----- */
2199:
2200:          /* 0xb7 */
2201:      0x00,    /* ----- */
2202:      0x00,    /* ----- */
2203:      0x00,    /* ----- */
2204:      0x00,    /* ----- */
2205:      0x18,    /* ---##----- */
2206:      0x18,    /* ---##----- */
2207:      0x00,    /* ----- */
2208:      0x00,    /* ----- */
2209:      0x00,    /* ----- */
2210:      0x00,    /* ----- */
2211:

```

drivers/video/font-lat9-8x10.c

Page 34/47

```

2212:                /* 0xb8          */
2213:    0xcc,          /* ##---##--- */
2214:    0x78,          /* -####--- */
2215:    0x00,          /* ----- */
2216:    0xfc,          /* #####--- */
2217:    0x98,          /* #---##--- */
2218:    0x30,          /* --##----- */
2219:    0x64,          /* -##---##--- */
2220:    0xfc,          /* #####--- */
2221:    0x00,          /* ----- */
2222:    0x00,          /* ----- */
2223:
2224:                /* 0xb9          */
2225:    0x60,          /* -##----- */
2226:    0xe0,          /* ###----- */
2227:    0x60,          /* -##----- */
2228:    0x60,          /* -##----- */
2229:    0xf0,          /* ####----- */
2230:    0x00,          /* ----- */
2231:    0x00,          /* ----- */
2232:    0x00,          /* ----- */
2233:    0x00,          /* ----- */
2234:    0x00,          /* ----- */
2235:
2236:                /* 0xba          */
2237:    0x38,          /* ---###---- */
2238:    0x6c,          /* -##-##--- */
2239:    0x6c,          /* -##-##--- */
2240:    0x38,          /* ---###---- */
2241:    0x00,          /* ----- */
2242:    0x7c,          /* -#####--- */
2243:    0x00,          /* ----- */
2244:    0x00,          /* ----- */
2245:    0x00,          /* ----- */
2246:    0x00,          /* ----- */
2247:
2248:                /* 0xbb          */
2249:    0x00,          /* ----- */
2250:    0x00,          /* ----- */
2251:    0xcc,          /* ##---##--- */
2252:    0x66,          /* -##-##--- */
2253:    0x33,          /* --##---## */
2254:    0x66,          /* -##-##--- */
2255:    0xcc,          /* ##---##--- */
2256:    0x00,          /* ----- */
2257:    0x00,          /* ----- */
2258:    0x00,          /* ----- */
2259:
2260:                /* 0xbc          */
2261:    0x00,          /* ----- */
2262:    0x7e,          /* -#####- */
2263:    0xd8,          /* ##-##--- */
2264:    0xd8,          /* ##-##--- */
2265:    0xdc,          /* ##-##--- */
2266:    0xd8,          /* ##-##--- */
2267:    0xd8,          /* ##-##--- */
2268:    0x7e,          /* -#####- */
2269:    0x00,          /* ----- */
2270:    0x00,          /* ----- */
2271:
2272:                /* 0xbd          */
2273:    0x00,          /* ----- */
2274:    0x00,          /* ----- */
2275:    0x00,          /* ----- */
2276:    0x7e,          /* -#####- */
2277:    0xdb,          /* ##-##-## */
2278:    0xde,          /* ##-##-##- */

```

drivers/video/font-lat9-8x10.c

Page 35/47

```

2279:      0xd8,    /* ##-##--- */
2280:      0x7e,    /* -#####- */
2281:      0x00,    /* ----- */
2282:      0x00,    /* ----- */
2283:
2284:          /* 0xbe */
2285:      0xcc,    /* ##--##-- */
2286:      0xcc,    /* ##--##-- */
2287:      0x00,    /* ----- */
2288:      0xcc,    /* ##--##-- */
2289:      0x78,    /* -####--- */
2290:      0x30,    /* --##---- */
2291:      0x30,    /* --##---- */
2292:      0x78,    /* -####--- */
2293:      0x00,    /* ----- */
2294:      0x00,    /* ----- */
2295:
2296:          /* 0xbf */
2297:      0x00,    /* ----- */
2298:      0x00,    /* ----- */
2299:      0x00,    /* ----- */
2300:      0x18,    /* ---##--- */
2301:      0x00,    /* ----- */
2302:      0x18,    /* ---##--- */
2303:      0x30,    /* --##---- */
2304:      0x60,    /* -##----- */
2305:      0x66,    /* -##-##- */
2306:      0x3c,    /* --####-- */
2307:
2308:          /* 0xc0 */
2309:      0x60,    /* -##----- */
2310:      0x30,    /* --##---- */
2311:      0x00,    /* ----- */
2312:      0x78,    /* -####--- */
2313:      0xcc,    /* ##--##-- */
2314:      0xfc,    /* #####--- */
2315:      0xcc,    /* ##--##-- */
2316:      0xcc,    /* ##--##-- */
2317:      0x00,    /* ----- */
2318:      0x00,    /* ----- */
2319:
2320:          /* 0xc1 */
2321:      0x18,    /* ---##--- */
2322:      0x30,    /* --##---- */
2323:      0x00,    /* ----- */
2324:      0x78,    /* -####--- */
2325:      0xcc,    /* ##--##-- */
2326:      0xfc,    /* #####--- */
2327:      0xcc,    /* ##--##-- */
2328:      0xcc,    /* ##--##-- */
2329:      0x00,    /* ----- */
2330:      0x00,    /* ----- */
2331:
2332:          /* 0xc2 */
2333:      0x78,    /* -####--- */
2334:      0xcc,    /* ##--##-- */
2335:      0x00,    /* ----- */
2336:      0x78,    /* -####--- */
2337:      0xcc,    /* ##--##-- */
2338:      0xfc,    /* #####--- */
2339:      0xcc,    /* ##--##-- */
2340:      0xcc,    /* ##--##-- */
2341:      0x00,    /* ----- */
2342:      0x00,    /* ----- */
2343:
2344:          /* 0xc3 */
2345:      0x76,    /* -##-##- */

```

drivers/video/font-lat9-8x10.c

Page 36/47

```

2346:      0xdc,    /* ##-##-# */
2347:      0x00,    /* ----- */
2348:      0x78,    /* -###-# */
2349:      0xcc,    /* ##-##-# */
2350:      0xfc,    /* #####-# */
2351:      0xcc,    /* ##-##-# */
2352:      0xcc,    /* ##-##-# */
2353:      0x00,    /* ----- */
2354:      0x00,    /* ----- */
2355:
2356:          /* 0xc4      */
2357:      0xcc,    /* ##-##-# */
2358:      0xcc,    /* ##-##-# */
2359:      0x00,    /* ----- */
2360:      0x78,    /* -###-# */
2361:      0xcc,    /* ##-##-# */
2362:      0xfc,    /* #####-# */
2363:      0xcc,    /* ##-##-# */
2364:      0xcc,    /* ##-##-# */
2365:      0x00,    /* ----- */
2366:      0x00,    /* ----- */
2367:
2368:          /* 0xc5      */
2369:      0x78,    /* -###-# */
2370:      0xcc,    /* ##-##-# */
2371:      0x78,    /* -###-# */
2372:      0x78,    /* -###-# */
2373:      0xcc,    /* ##-##-# */
2374:      0xfc,    /* #####-# */
2375:      0xcc,    /* ##-##-# */
2376:      0xcc,    /* ##-##-# */
2377:      0x00,    /* ----- */
2378:      0x00,    /* ----- */
2379:
2380:          /* 0xc6      */
2381:      0x00,    /* ----- */
2382:      0x3e,    /* --##### */
2383:      0x78,    /* -###-# */
2384:      0xd8,    /* #-##-# */
2385:      0xfc,    /* #####-# */
2386:      0xd8,    /* #-##-# */
2387:      0xd8,    /* #-##-# */
2388:      0xde,    /* #-##### */
2389:      0x00,    /* ----- */
2390:      0x00,    /* ----- */
2391:
2392:          /* 0xc7      */
2393:      0x00,    /* ----- */
2394:      0x3c,    /* --##### */
2395:      0x66,    /* -##-##-# */
2396:      0xc0,    /* ##-#-#-# */
2397:      0xc0,    /* ##-#-#-# */
2398:      0xc0,    /* ##-#-#-# */
2399:      0x66,    /* -##-##-# */
2400:      0x3c,    /* --##### */
2401:      0x0c,    /* ----##-# */
2402:      0x78,    /* -###-# */
2403:
2404:          /* 0xc8      */
2405:      0x60,    /* -##-#-#-# */
2406:      0x30,    /* --##-#-#-# */
2407:      0x00,    /* ----- */
2408:      0xfe,    /* #####-#-# */
2409:      0x62,    /* -##-##-#-# */
2410:      0x78,    /* -###-#-#-# */
2411:      0x62,    /* -##-##-#-# */
2412:      0xfe,    /* #####-#-# */

```

drivers/video/font-lat9-8x10.c

Page 37/47

```

2413:      0x00, /* ----- */
2414:      0x00, /* ----- */
2415:
2416:          /* 0xc9 */
2417:      0x0c, /* ----##-- */
2418:      0x18, /* ---##--- */
2419:      0x00, /* ----- */
2420:      0xfe, /* #####- */
2421:      0x62, /* -##---#- */
2422:      0x78, /* -####--- */
2423:      0x62, /* -##---#- */
2424:      0xfe, /* #####- */
2425:      0x00, /* ----- */
2426:      0x00, /* ----- */
2427:
2428:          /* 0xca */
2429:      0x38, /* --###--- */
2430:      0x6c, /* -##-##-- */
2431:      0x00, /* ----- */
2432:      0xfe, /* #####- */
2433:      0x62, /* -##---#- */
2434:      0x78, /* -####--- */
2435:      0x62, /* -##---#- */
2436:      0xfe, /* #####- */
2437:      0x00, /* ----- */
2438:      0x00, /* ----- */
2439:
2440:          /* 0xcb */
2441:      0x6c, /* -##-##-- */
2442:      0x6c, /* -##-##-- */
2443:      0x00, /* ----- */
2444:      0xfe, /* #####- */
2445:      0x62, /* -##---#- */
2446:      0x78, /* -####--- */
2447:      0x62, /* -##---#- */
2448:      0xfe, /* #####- */
2449:      0x00, /* ----- */
2450:      0x00, /* ----- */
2451:
2452:          /* 0xcc */
2453:      0x60, /* -##----- */
2454:      0x30, /* --##----- */
2455:      0x00, /* ----- */
2456:      0x78, /* -####--- */
2457:      0x30, /* --##----- */
2458:      0x30, /* --##----- */
2459:      0x30, /* --##----- */
2460:      0x78, /* -####--- */
2461:      0x00, /* ----- */
2462:      0x00, /* ----- */
2463:
2464:          /* 0xcd */
2465:      0x18, /* ---##--- */
2466:      0x30, /* --##----- */
2467:      0x00, /* ----- */
2468:      0x78, /* -####--- */
2469:      0x30, /* --##----- */
2470:      0x30, /* --##----- */
2471:      0x30, /* --##----- */
2472:      0x78, /* -####--- */
2473:      0x00, /* ----- */
2474:      0x00, /* ----- */
2475:
2476:          /* 0xce */
2477:      0x78, /* -####--- */
2478:      0xcc, /* ##--##-- */
2479:      0x00, /* ----- */

```


drivers/video/font-lat9-8x10.c

Page 38/47

```

2480:      0x78,    /* #####--- */
2481:      0x30,    /* ---#----- */
2482:      0x30,    /* ---#----- */
2483:      0x30,    /* ---#----- */
2484:      0x78,    /* #####--- */
2485:      0x00,    /* ----- */
2486:      0x00,    /* ----- */
2487:
2488:          /* 0xcf          */
2489:      0xcc,    /* ##--##-- */
2490:      0xcc,    /* ##--##-- */
2491:      0x00,    /* ----- */
2492:      0x78,    /* #####--- */
2493:      0x30,    /* ---#----- */
2494:      0x30,    /* ---#----- */
2495:      0x30,    /* ---#----- */
2496:      0x78,    /* #####--- */
2497:      0x00,    /* ----- */
2498:      0x00,    /* ----- */
2499:
2500:          /* 0xd0          */
2501:      0x00,    /* ----- */
2502:      0xf8,    /* #####--- */
2503:      0x6c,    /* -##-##-- */
2504:      0x66,    /* -##--##- */
2505:      0xf6,    /* #####-##- */
2506:      0x66,    /* -##--##- */
2507:      0x6c,    /* -##-##-- */
2508:      0xf8,    /* #####--- */
2509:      0x00,    /* ----- */
2510:      0x00,    /* ----- */
2511:
2512:          /* 0xd1          */
2513:      0x76,    /* -##-##- */
2514:      0xdc,    /* #-###-- */
2515:      0x00,    /* ----- */
2516:      0xe6,    /* ###--##- */
2517:      0xf6,    /* #####-##- */
2518:      0xde,    /* #-##### */
2519:      0xce,    /* ##--###- */
2520:      0xc6,    /* ##---##- */
2521:      0x00,    /* ----- */
2522:      0x00,    /* ----- */
2523:
2524:          /* 0xd2          */
2525:      0x60,    /* -##----- */
2526:      0x30,    /* ---#----- */
2527:      0x00,    /* ----- */
2528:      0x78,    /* #####--- */
2529:      0xcc,    /* ##--##-- */
2530:      0xcc,    /* ##--##-- */
2531:      0xcc,    /* ##--##-- */
2532:      0x78,    /* #####--- */
2533:      0x00,    /* ----- */
2534:      0x00,    /* ----- */
2535:
2536:          /* 0xd3          */
2537:      0x18,    /* ---#----- */
2538:      0x30,    /* ---#----- */
2539:      0x00,    /* ----- */
2540:      0x78,    /* #####--- */
2541:      0xcc,    /* ##--##-- */
2542:      0xcc,    /* ##--##-- */
2543:      0xcc,    /* ##--##-- */
2544:      0x78,    /* #####--- */
2545:      0x00,    /* ----- */
2546:      0x00,    /* ----- */

```

drivers/video/font-lat9-8x10.c

Page 39/47

```

2547:
2548:          /* 0xd4          */
2549:    0x78, /* -####--- */
2550:    0xcc, /* ##--##-- */
2551:    0x00, /* ----- */
2552:    0x78, /* -####--- */
2553:    0xcc, /* ##--##-- */
2554:    0xcc, /* ##--##-- */
2555:    0xcc, /* ##--##-- */
2556:    0x78, /* -####--- */
2557:    0x00, /* ----- */
2558:    0x00, /* ----- */
2559:
2560:          /* 0xd5          */
2561:    0x76, /* -###-##- */
2562:    0xdc, /* #-###-- */
2563:    0x00, /* ----- */
2564:    0x78, /* -####--- */
2565:    0xcc, /* ##--##-- */
2566:    0xcc, /* ##--##-- */
2567:    0xcc, /* ##--##-- */
2568:    0x78, /* -####--- */
2569:    0x00, /* ----- */
2570:    0x00, /* ----- */
2571:
2572:          /* 0xd6          */
2573:    0xcc, /* ##--##-- */
2574:    0xcc, /* ##--##-- */
2575:    0x00, /* ----- */
2576:    0x78, /* -####--- */
2577:    0xcc, /* ##--##-- */
2578:    0xcc, /* ##--##-- */
2579:    0xcc, /* ##--##-- */
2580:    0x78, /* -####--- */
2581:    0x00, /* ----- */
2582:    0x00, /* ----- */
2583:
2584:          /* 0xd7          */
2585:    0x00, /* ----- */
2586:    0x00, /* ----- */
2587:    0x6c, /* -##-##- */
2588:    0x38, /* --###--- */
2589:    0x38, /* --###--- */
2590:    0x6c, /* -##-##- */
2591:    0x00, /* ----- */
2592:    0x00, /* ----- */
2593:    0x00, /* ----- */
2594:    0x00, /* ----- */
2595:
2596:          /* 0xd8          */
2597:    0x00, /* ----- */
2598:    0x3e, /* --##### */
2599:    0x6c, /* -##-##- */
2600:    0xde, /* #-##### */
2601:    0xd6, /* #-##-##- */
2602:    0xf6, /* #####-##- */
2603:    0x6c, /* -##-##- */
2604:    0xf8, /* #####--- */
2605:    0x00, /* ----- */
2606:    0x00, /* ----- */
2607:
2608:          /* 0xd9          */
2609:    0x60, /* -##----- */
2610:    0x30, /* --##----- */
2611:    0x00, /* ----- */
2612:    0xcc, /* ##--##-- */
2613:    0xcc, /* ##--##-- */

```

drivers/video/font-lat9-8x10.c

Page 40/47

```

2614:      0xcc, /* ##--##-- */
2615:      0xcc, /* ##--##-- */
2616:      0x78, /* -###---- */
2617:      0x00, /* ----- */
2618:      0x00, /* ----- */
2619:
2620:          /* 0xda */
2621:      0x18, /* ---##---- */
2622:      0x30, /* --##----- */
2623:      0x00, /* ----- */
2624:      0xcc, /* ##--##-- */
2625:      0xcc, /* ##--##-- */
2626:      0xcc, /* ##--##-- */
2627:      0xcc, /* ##--##-- */
2628:      0x78, /* -###---- */
2629:      0x00, /* ----- */
2630:      0x00, /* ----- */
2631:
2632:          /* 0xdb */
2633:      0x78, /* -###---- */
2634:      0xcc, /* ##--##-- */
2635:      0x00, /* ----- */
2636:      0xcc, /* ##--##-- */
2637:      0xcc, /* ##--##-- */
2638:      0xcc, /* ##--##-- */
2639:      0xcc, /* ##--##-- */
2640:      0x78, /* -###---- */
2641:      0x00, /* ----- */
2642:      0x00, /* ----- */
2643:
2644:          /* 0xdc */
2645:      0xcc, /* ##--##-- */
2646:      0xcc, /* ##--##-- */
2647:      0x00, /* ----- */
2648:      0xcc, /* ##--##-- */
2649:      0xcc, /* ##--##-- */
2650:      0xcc, /* ##--##-- */
2651:      0xcc, /* ##--##-- */
2652:      0x78, /* -###---- */
2653:      0x00, /* ----- */
2654:      0x00, /* ----- */
2655:
2656:          /* 0xdd */
2657:      0x18, /* ---##---- */
2658:      0x30, /* --##----- */
2659:      0x00, /* ----- */
2660:      0xcc, /* ##--##-- */
2661:      0xcc, /* ##--##-- */
2662:      0x78, /* -###---- */
2663:      0x30, /* --##----- */
2664:      0x78, /* -###---- */
2665:      0x00, /* ----- */
2666:      0x00, /* ----- */
2667:
2668:          /* 0xde */
2669:      0x00, /* ----- */
2670:      0xf0, /* #####---- */
2671:      0x60, /* -##----- */
2672:      0x7c, /* -#####-- */
2673:      0x66, /* -##--##-- */
2674:      0x7c, /* -#####-- */
2675:      0x60, /* -##----- */
2676:      0xf0, /* #####---- */
2677:      0x00, /* ----- */
2678:      0x00, /* ----- */
2679:
2680:          /* 0xdf */

```

drivers/video/font-lat9-8x10.c

Page 41/47

```

2681:      0x00,    /* ----- */
2682:      0x7c,    /* -##### */
2683:      0xc6,    /* ##---## */
2684:      0xc6,    /* ##---## */
2685:      0xcc,    /* ##---## */
2686:      0xc6,    /* ##---## */
2687:      0xd6,    /* ##-##-# */
2688:      0xdc,    /* ##-#### */
2689:      0x80,    /* #----- */
2690:      0x00,    /* ----- */
2691:
2692:          /* 0xe0      */
2693:      0x60,    /* -##----- */
2694:      0x30,    /* --##----- */
2695:      0x00,    /* ----- */
2696:      0x78,    /* -##### */
2697:      0x0c,    /* -----## */
2698:      0x7c,    /* -##### */
2699:      0xcc,    /* ##---## */
2700:      0x76,    /* -##-##-# */
2701:      0x00,    /* ----- */
2702:      0x00,    /* ----- */
2703:
2704:          /* 0xe1      */
2705:      0x18,    /* ---##---- */
2706:      0x30,    /* --##----- */
2707:      0x00,    /* ----- */
2708:      0x78,    /* -##### */
2709:      0x0c,    /* -----## */
2710:      0x7c,    /* -##### */
2711:      0xcc,    /* ##---## */
2712:      0x76,    /* -##-##-# */
2713:      0x00,    /* ----- */
2714:      0x00,    /* ----- */
2715:
2716:          /* 0xe2      */
2717:      0x38,    /* --###---- */
2718:      0x6c,    /* -##-##-# */
2719:      0x00,    /* ----- */
2720:      0x78,    /* -##### */
2721:      0x0c,    /* -----## */
2722:      0x7c,    /* -##### */
2723:      0xcc,    /* ##---## */
2724:      0x76,    /* -##-##-# */
2725:      0x00,    /* ----- */
2726:      0x00,    /* ----- */
2727:
2728:          /* 0xe3      */
2729:      0x76,    /* -##-##-# */
2730:      0xdc,    /* ##-#### */
2731:      0x00,    /* ----- */
2732:      0x78,    /* -##### */
2733:      0x0c,    /* -----## */
2734:      0x7c,    /* -##### */
2735:      0xcc,    /* ##---## */
2736:      0x76,    /* -##-##-# */
2737:      0x00,    /* ----- */
2738:      0x00,    /* ----- */
2739:
2740:          /* 0xe4      */
2741:      0x6c,    /* -##-##-# */
2742:      0x6c,    /* -##-##-# */
2743:      0x00,    /* ----- */
2744:      0x78,    /* -##### */
2745:      0x0c,    /* -----## */
2746:      0x7c,    /* -##### */
2747:      0xcc,    /* ##---## */

```

drivers/video/font-lat9-8x10.c

Page 42/47

```

2748:      0x76,    /* ---###--- */
2749:      0x00,    /* ----- */
2750:      0x00,    /* ----- */
2751:
2752:          /* 0xe5 */
2753:      0x38,    /* --###--- */
2754:      0x6c,    /* -##-##-- */
2755:      0x38,    /* --###--- */
2756:      0x78,    /* -####--- */
2757:      0x0c,    /* ----##-- */
2758:      0x7c,    /* -#####-- */
2759:      0xcc,    /* ##--##-- */
2760:      0x76,    /* -###-##- */
2761:      0x00,    /* ----- */
2762:      0x00,    /* ----- */
2763:
2764:          /* 0xe6 */
2765:      0x00,    /* ----- */
2766:      0x00,    /* ----- */
2767:      0x00,    /* ----- */
2768:      0x7e,    /* -#####- */
2769:      0x1b,    /* ---##-## */
2770:      0x7e,    /* -#####- */
2771:      0xd8,    /* #-##--- */
2772:      0x7e,    /* -#####- */
2773:      0x00,    /* ----- */
2774:      0x00,    /* ----- */
2775:
2776:          /* 0xe7 */
2777:      0x00,    /* ----- */
2778:      0x00,    /* ----- */
2779:      0x00,    /* ----- */
2780:      0x78,    /* -####--- */
2781:      0xcc,    /* ##--##-- */
2782:      0xc0,    /* #------ */
2783:      0xcc,    /* ##--##-- */
2784:      0x78,    /* -####--- */
2785:      0x18,    /* ---##--- */
2786:      0x70,    /* -###---- */
2787:
2788:          /* 0xe8 */
2789:      0x60,    /* -##----- */
2790:      0x30,    /* --##----- */
2791:      0x00,    /* ----- */
2792:      0x78,    /* -####--- */
2793:      0xcc,    /* ##--##-- */
2794:      0xfc,    /* #####--- */
2795:      0xc0,    /* #------ */
2796:      0x78,    /* -####--- */
2797:      0x00,    /* ----- */
2798:      0x00,    /* ----- */
2799:
2800:          /* 0xe9 */
2801:      0x0c,    /* ----##-- */
2802:      0x18,    /* ---##--- */
2803:      0x00,    /* ----- */
2804:      0x78,    /* -####--- */
2805:      0xcc,    /* ##--##-- */
2806:      0xfc,    /* #####--- */
2807:      0xc0,    /* #------ */
2808:      0x78,    /* -####--- */
2809:      0x00,    /* ----- */
2810:      0x00,    /* ----- */
2811:
2812:          /* 0xea */
2813:      0x78,    /* -####--- */
2814:      0xcc,    /* ##--##-- */

```

drivers/video/font-lat9-8x10.c

Page 43/47

```

2815:      0x00, /* ----- */
2816:      0x78, /* -###---- */
2817:      0xcc, /* ##--##-- */
2818:      0xfc, /* #####--- */
2819:      0xc0, /* ##----- */
2820:      0x78, /* -###---- */
2821:      0x00, /* ----- */
2822:      0x00, /* ----- */
2823:
2824:          /* 0xeb */
2825:      0xcc, /* ##--##-- */
2826:      0xcc, /* ##--##-- */
2827:      0x00, /* ----- */
2828:      0x78, /* -###---- */
2829:      0xcc, /* ##--##-- */
2830:      0xfc, /* #####--- */
2831:      0xc0, /* ##----- */
2832:      0x78, /* -###---- */
2833:      0x00, /* ----- */
2834:      0x00, /* ----- */
2835:
2836:          /* 0xec */
2837:      0x00, /* ----- */
2838:      0x60, /* -##----- */
2839:      0x30, /* --##----- */
2840:      0x00, /* ----- */
2841:      0x70, /* -###---- */
2842:      0x30, /* --##----- */
2843:      0x30, /* --##----- */
2844:      0x78, /* -###---- */
2845:      0x00, /* ----- */
2846:      0x00, /* ----- */
2847:
2848:          /* 0xed */
2849:      0x00, /* ----- */
2850:      0x18, /* ---##---- */
2851:      0x30, /* --##----- */
2852:      0x00, /* ----- */
2853:      0x70, /* -###---- */
2854:      0x30, /* --##----- */
2855:      0x30, /* --##----- */
2856:      0x78, /* -###---- */
2857:      0x00, /* ----- */
2858:      0x00, /* ----- */
2859:
2860:          /* 0xee */
2861:      0x20, /* --#----- */
2862:      0x70, /* -###---- */
2863:      0xd8, /* ##-##---- */
2864:      0x00, /* ----- */
2865:      0x70, /* -###---- */
2866:      0x30, /* --##----- */
2867:      0x30, /* --##----- */
2868:      0x78, /* -###---- */
2869:      0x00, /* ----- */
2870:      0x00, /* ----- */
2871:
2872:          /* 0xef */
2873:      0x00, /* ----- */
2874:      0xd8, /* ##-##---- */
2875:      0xd8, /* ##-##---- */
2876:      0x00, /* ----- */
2877:      0x70, /* -###---- */
2878:      0x30, /* --##----- */
2879:      0x30, /* --##----- */
2880:      0x78, /* -###---- */
2881:      0x00, /* ----- */

```

drivers/video/font-lat9-8x10.c

Page 44/47

```

2882:      0x00,    /* ----- */
2883:
2884:      /* 0xf0 */
2885:      0x00,    /* ----- */
2886:      0x78,    /* -####- */
2887:      0x70,    /* -###- */
2888:      0x18,    /* ---##- */
2889:      0x7c,    /* -#####- */
2890:      0xcc,    /* ##--##- */
2891:      0xcc,    /* ##--##- */
2892:      0x78,    /* -####- */
2893:      0x00,    /* ----- */
2894:      0x00,    /* ----- */
2895:
2896:      /* 0xf1 */
2897:      0x76,    /* -###-##- */
2898:      0xdc,    /* ##-###- */
2899:      0x00,    /* ----- */
2900:      0xf8,    /* #####- */
2901:      0xcc,    /* ##--##- */
2902:      0xcc,    /* ##--##- */
2903:      0xcc,    /* ##--##- */
2904:      0xcc,    /* ##--##- */
2905:      0x00,    /* ----- */
2906:      0x00,    /* ----- */
2907:
2908:      /* 0xf2 */
2909:      0x00,    /* ----- */
2910:      0x60,    /* -##- */
2911:      0x30,    /* --##- */
2912:      0x00,    /* ----- */
2913:      0x78,    /* -####- */
2914:      0xcc,    /* ##--##- */
2915:      0xcc,    /* ##--##- */
2916:      0x78,    /* -####- */
2917:      0x00,    /* ----- */
2918:      0x00,    /* ----- */
2919:
2920:      /* 0xf3 */
2921:      0x00,    /* ----- */
2922:      0x18,    /* ---##- */
2923:      0x30,    /* --##- */
2924:      0x00,    /* ----- */
2925:      0x78,    /* -####- */
2926:      0xcc,    /* ##--##- */
2927:      0xcc,    /* ##--##- */
2928:      0x78,    /* -####- */
2929:      0x00,    /* ----- */
2930:      0x00,    /* ----- */
2931:
2932:      /* 0xf4 */
2933:      0x30,    /* --##- */
2934:      0x78,    /* -####- */
2935:      0xcc,    /* ##--##- */
2936:      0x00,    /* ----- */
2937:      0x78,    /* -####- */
2938:      0xcc,    /* ##--##- */
2939:      0xcc,    /* ##--##- */
2940:      0x78,    /* -####- */
2941:      0x00,    /* ----- */
2942:      0x00,    /* ----- */
2943:
2944:      /* 0xf5 */
2945:      0x00,    /* ----- */
2946:      0x76,    /* -###-##- */
2947:      0xdc,    /* ##-###- */
2948:      0x00,    /* ----- */

```

drivers/video/font-lat9-8x10.c

Page 45/47

```

2949:      0x78,    /* #####--- */
2950:      0xcc,    /* ##---##--- */
2951:      0xcc,    /* ##---##--- */
2952:      0x78,    /* #####--- */
2953:      0x00,    /* ----- */
2954:      0x00,    /* ----- */
2955:
2956:          /* 0xf6          */
2957:      0x00,    /* ----- */
2958:      0xcc,    /* ##---##--- */
2959:      0xcc,    /* ##---##--- */
2960:      0x00,    /* ----- */
2961:      0x78,    /* #####--- */
2962:      0xcc,    /* ##---##--- */
2963:      0xcc,    /* ##---##--- */
2964:      0x78,    /* #####--- */
2965:      0x00,    /* ----- */
2966:      0x00,    /* ----- */
2967:
2968:          /* 0xf7          */
2969:      0x00,    /* ----- */
2970:      0x30,    /* --##----- */
2971:      0x30,    /* --##----- */
2972:      0x00,    /* ----- */
2973:      0xfc,    /* #####--- */
2974:      0x00,    /* ----- */
2975:      0x30,    /* --##----- */
2976:      0x30,    /* --##----- */
2977:      0x00,    /* ----- */
2978:      0x00,    /* ----- */
2979:
2980:          /* 0xf8          */
2981:      0x00,    /* ----- */
2982:      0x00,    /* ----- */
2983:      0x00,    /* ----- */
2984:      0x7c,    /* #####--- */
2985:      0xdc,    /* ##---##--- */
2986:      0xfc,    /* #####--- */
2987:      0xec,    /* ##---##--- */
2988:      0xf8,    /* #####--- */
2989:      0x00,    /* ----- */
2990:      0x00,    /* ----- */
2991:
2992:          /* 0xf9          */
2993:      0x60,    /* -##----- */
2994:      0x30,    /* --##----- */
2995:      0x00,    /* ----- */
2996:      0xcc,    /* ##---##--- */
2997:      0xcc,    /* ##---##--- */
2998:      0xcc,    /* ##---##--- */
2999:      0xcc,    /* ##---##--- */
3000:      0x76,    /* -##-##- */
3001:      0x00,    /* ----- */
3002:      0x00,    /* ----- */
3003:
3004:          /* 0xfa          */
3005:      0x18,    /* ---##----- */
3006:      0x30,    /* --##----- */
3007:      0x00,    /* ----- */
3008:      0xcc,    /* ##---##--- */
3009:      0xcc,    /* ##---##--- */
3010:      0xcc,    /* ##---##--- */
3011:      0xcc,    /* ##---##--- */
3012:      0x76,    /* -##-##- */
3013:      0x00,    /* ----- */
3014:      0x00,    /* ----- */
3015:

```


drivers/video/font-lat9-8x10.c

Page 46/47

```

3016:                /* 0xfb */
3017:    0x78,          /* -###---- */
3018:    0xcc,          /* ##--##-- */
3019:    0x00,          /* ----- */
3020:    0xcc,          /* ##--##-- */
3021:    0xcc,          /* ##--##-- */
3022:    0xcc,          /* ##--##-- */
3023:    0xcc,          /* ##--##-- */
3024:    0x76,          /* -###-##- */
3025:    0x00,          /* ----- */
3026:    0x00,          /* ----- */
3027:
3028:                /* 0xfc */
3029:    0xcc,          /* ##--##-- */
3030:    0xcc,          /* ##--##-- */
3031:    0x00,          /* ----- */
3032:    0xcc,          /* ##--##-- */
3033:    0xcc,          /* ##--##-- */
3034:    0xcc,          /* ##--##-- */
3035:    0xcc,          /* ##--##-- */
3036:    0x76,          /* -###-##- */
3037:    0x00,          /* ----- */
3038:    0x00,          /* ----- */
3039:
3040:                /* 0xfd */
3041:    0x18,          /* ----##---- */
3042:    0x30,          /* --##----- */
3043:    0x00,          /* ----- */
3044:    0xcc,          /* ##--##-- */
3045:    0xcc,          /* ##--##-- */
3046:    0xcc,          /* ##--##-- */
3047:    0x7c,          /* -#####-- */
3048:    0x0c,          /* ----##-- */
3049:    0xf8,          /* #####----- */
3050:    0x00,          /* ----- */
3051:
3052:                /* 0xfe */
3053:    0x00,          /* ----- */
3054:    0xf0,          /* #####----- */
3055:    0x60,          /* -##----- */
3056:    0x78,          /* -#####-- */
3057:    0x6c,          /* -##-##-- */
3058:    0x6c,          /* -##-##-- */
3059:    0x78,          /* -#####-- */
3060:    0x60,          /* -##----- */
3061:    0xf0,          /* #####----- */
3062:    0x00,          /* ----- */
3063:
3064:                /* 0xff */
3065:    0xcc,          /* ##--##-- */
3066:    0xcc,          /* ##--##-- */
3067:    0x00,          /* ----- */
3068:    0xcc,          /* ##--##-- */
3069:    0xcc,          /* ##--##-- */
3070:    0xcc,          /* ##--##-- */
3071:    0x7c,          /* -#####-- */
3072:    0x0c,          /* ----##-- */
3073:    0xf8,          /* #####----- */
3074:    0x00,          /* ----- */
3075:
3076: };
3077:
3078: static unsigned char cursorshape_8x10[] = {
3079:    0x00,          /* ----- */
3080:    0x00,          /* ----- */
3081:    0x00,          /* ----- */
3082:    0x00,          /* ----- */

```

drivers/video/font-lat9-8x10.c

Page 47/47

```
3083:         0x00,    /* ----- */
3084:         0x00,    /* ----- */
3085:         0x00,    /* ----- */
3086:         0x00,    /* ----- */
3087:         0xFF,    /* ##### */
3088:         0xFF,    /* ##### */
3089: };
3090:
3091: struct fbcon_font_desc font_lat9_8x10 = {
3092:     "VGA8x10",
3093:     8,
3094:     10,
3095:     fontdata_8x10,
3096:     cursorshape_8x10
3097: };
```

drivers/video/font-lat9-8x12.c

Page 1/54

```

1: #include <fiwix/font.h>
2:
3: static unsigned char fontdata_8x12[] = {
4:     /* 0x00 */
5:     0x7e, /* -#####- */
6:     0xc3, /* ##----## */
7:     0x99, /* #--##--# */
8:     0x99, /* #--##--# */
9:     0xf3, /* #####--## */
10:    0xe7, /* #####--## */
11:    0xe7, /* #####--## */
12:    0xff, /* ######## */
13:    0xe7, /* #####--## */
14:    0xe7, /* #####--## */
15:    0x7e, /* -#####- */
16:    0x00, /* ----- */
17:
18:    /* 0x01 */
19:    0x00, /* ----- */
20:    0x00, /* ----- */
21:    0x00, /* ----- */
22:    0x76, /* -###-##- */
23:    0xdc, /* #-###-#- */
24:    0x00, /* ----- */
25:    0x76, /* -###-##- */
26:    0xdc, /* #-###-#- */
27:    0x00, /* ----- */
28:    0x00, /* ----- */
29:    0x00, /* ----- */
30:    0x00, /* ----- */
31:
32:    /* 0x02 */
33:    0x6e, /* -##-###- */
34:    0xd8, /* #-##-###- */
35:    0xd8, /* #-##-###- */
36:    0xd8, /* #-##-###- */
37:    0xd8, /* #-##-###- */
38:    0xde, /* #-#####- */
39:    0xd8, /* #-##-###- */
40:    0xd8, /* #-##-###- */
41:    0xd8, /* #-##-###- */
42:    0x6e, /* -##-###- */
43:    0x00, /* ----- */
44:    0x00, /* ----- */
45:
46:    /* 0x03 */
47:    0x00, /* ----- */
48:    0x00, /* ----- */
49:    0x00, /* ----- */
50:    0x6e, /* -##-###- */
51:    0xdb, /* #-##-###- */
52:    0xdb, /* #-##-###- */
53:    0xdf, /* #-#####- */
54:    0xd8, /* #-##-###- */
55:    0xdb, /* #-##-###- */
56:    0x6e, /* -##-###- */
57:    0x00, /* ----- */
58:    0x00, /* ----- */
59:
60:    /* 0x04 */
61:    0x00, /* ----- */
62:    0x00, /* ----- */
63:    0x10, /* ----#---- */
64:    0x38, /* --###---- */
65:    0x7c, /* -#####- */
66:    0xfe, /* ########- */
67:    0x7c, /* -#####- */

```

drivers/video/font-lat9-8x12.c

Page 2/54

```

68:      0x38,    /* ---###--- */
69:      0x10,    /* ----#---- */
70:      0x00,    /* ----- */
71:      0x00,    /* ----- */
72:      0x00,    /* ----- */
73:
74:      /* 0x05 */
75:      0x88,    /* #---#--- */
76:      0x88,    /* #---#--- */
77:      0xf8,    /* #####--- */
78:      0x88,    /* #---#--- */
79:      0x88,    /* #---#--- */
80:      0x00,    /* ----- */
81:      0x3e,    /* ---##### */
82:      0x08,    /* ----#---- */
83:      0x08,    /* ----#---- */
84:      0x08,    /* ----#---- */
85:      0x08,    /* ----#---- */
86:      0x00,    /* ----- */
87:
88:      /* 0x06 */
89:      0xf8,    /* #####--- */
90:      0x80,    /* #----- */
91:      0xe0,    /* ##----- */
92:      0x80,    /* #----- */
93:      0x80,    /* #----- */
94:      0x00,    /* ----- */
95:      0x3e,    /* ---##### */
96:      0x20,    /* --#----- */
97:      0x38,    /* --###---- */
98:      0x20,    /* --#----- */
99:      0x20,    /* --#----- */
100:     0x00,    /* ----- */
101:
102:     /* 0x07 */
103:     0x78,    /* -##### */
104:     0x80,    /* #----- */
105:     0x80,    /* #----- */
106:     0x80,    /* #----- */
107:     0x78,    /* -##### */
108:     0x00,    /* ----- */
109:     0x3c,    /* ---####-- */
110:     0x22,    /* --#---#- */
111:     0x3e,    /* ---##### */
112:     0x24,    /* --#---#- */
113:     0x22,    /* --#---#- */
114:     0x00,    /* ----- */
115:
116:     /* 0x08 */
117:     0x80,    /* #----- */
118:     0x80,    /* #----- */
119:     0x80,    /* #----- */
120:     0x80,    /* #----- */
121:     0xf8,    /* #####--- */
122:     0x00,    /* ----- */
123:     0x3e,    /* ---##### */
124:     0x20,    /* --#----- */
125:     0x38,    /* --###---- */
126:     0x20,    /* --#----- */
127:     0x20,    /* --#----- */
128:     0x00,    /* ----- */
129:
130:     /* 0x09 */
131:     0x22,    /* --#---#- */
132:     0x88,    /* #---#--- */
133:     0x22,    /* --#---#- */
134:     0x88,    /* #---#--- */

```

drivers/video/font-lat9-8x12.c Page 3/54

```

135:      0x22, /* --#---#- */
136:      0x88, /* #---#--- */
137:      0x22, /* --#---#- */
138:      0x88, /* #---#--- */
139:      0x22, /* --#---#- */
140:      0x88, /* #---#--- */
141:      0x22, /* --#---#- */
142:      0x88, /* #---#--- */
143:
144:      /* 0x0a */
145:      0x55, /* -#-#-#-# */
146:      0xaa, /* #-#-#-#- */
147:      0x55, /* -#-#-#-# */
148:      0xaa, /* #-#-#-#- */
149:      0x55, /* -#-#-#-# */
150:      0xaa, /* #-#-#-#- */
151:      0x55, /* -#-#-#-# */
152:      0xaa, /* #-#-#-#- */
153:      0x55, /* -#-#-#-# */
154:      0xaa, /* #-#-#-#- */
155:      0x55, /* -#-#-#-# */
156:      0xaa, /* #-#-#-#- */
157:
158:      /* 0x0b */
159:      0xee, /* ###-###- */
160:      0xbb, /* #-###-## */
161:      0xee, /* ###-###- */
162:      0xbb, /* #-###-## */
163:      0xee, /* ###-###- */
164:      0xbb, /* #-###-## */
165:      0xee, /* ###-###- */
166:      0xbb, /* #-###-## */
167:      0xee, /* ###-###- */
168:      0xbb, /* #-###-## */
169:      0xee, /* ###-###- */
170:      0xbb, /* #-###-## */
171:
172:      /* 0x0c */
173:      0xff, /* ##### */
174:      0xff, /* ##### */
175:      0xff, /* ##### */
176:      0xff, /* ##### */
177:      0xff, /* ##### */
178:      0xff, /* ##### */
179:      0xff, /* ##### */
180:      0xff, /* ##### */
181:      0xff, /* ##### */
182:      0xff, /* ##### */
183:      0xff, /* ##### */
184:      0xff, /* ##### */
185:
186:      /* 0x0d */
187:      0x00, /* ----- */
188:      0x00, /* ----- */
189:      0x00, /* ----- */
190:      0x00, /* ----- */
191:      0x00, /* ----- */
192:      0x00, /* ----- */
193:      0xff, /* ##### */
194:      0xff, /* ##### */
195:      0xff, /* ##### */
196:      0xff, /* ##### */
197:      0xff, /* ##### */
198:      0xff, /* ##### */
199:
200:      /* 0x0e */
201:      0xff, /* ##### */

```

drivers/video/font-lat9-8x12.c

```

202:      0xff, /* ##### */
203:      0xff, /* ##### */
204:      0xff, /* ##### */
205:      0xff, /* ##### */
206:      0xff, /* ##### */
207:      0x00, /* ----- */
208:      0x00, /* ----- */
209:      0x00, /* ----- */
210:      0x00, /* ----- */
211:      0x00, /* ----- */
212:      0x00, /* ----- */
213:
214:          /* 0x0f */
215:      0xf0, /* ####---- */
216:      0xf0, /* ####---- */
217:      0xf0, /* ####---- */
218:      0xf0, /* ####---- */
219:      0xf0, /* ####---- */
220:      0xf0, /* ####---- */
221:      0xf0, /* ####---- */
222:      0xf0, /* ####---- */
223:      0xf0, /* ####---- */
224:      0xf0, /* ####---- */
225:      0xf0, /* ####---- */
226:      0xf0, /* ####---- */
227:
228:          /* 0x10 */
229:      0x0f, /* ----#### */
230:      0x0f, /* ----#### */
231:      0x0f, /* ----#### */
232:      0x0f, /* ----#### */
233:      0x0f, /* ----#### */
234:      0x0f, /* ----#### */
235:      0x0f, /* ----#### */
236:      0x0f, /* ----#### */
237:      0x0f, /* ----#### */
238:      0x0f, /* ----#### */
239:      0x0f, /* ----#### */
240:      0x0f, /* ----#### */
241:
242:          /* 0x11 */
243:      0x88, /* #---#--- */
244:      0xc8, /* ##--#--- */
245:      0xa8, /* #-#-#--- */
246:      0x98, /* #--##--- */
247:      0x88, /* #---#--- */
248:      0x00, /* ----- */
249:      0x20, /* --#----- */
250:      0x20, /* --#----- */
251:      0x20, /* --#----- */
252:      0x20, /* --#----- */
253:      0x3e, /* --##### */
254:      0x00, /* ----- */
255:
256:          /* 0x12 */
257:      0x88, /* #---#--- */
258:      0x88, /* #---#--- */
259:      0x50, /* -#-#---- */
260:      0x50, /* -#-#---- */
261:      0x20, /* --#----- */
262:      0x00, /* ----- */
263:      0x3e, /* --##### */
264:      0x08, /* ----#---- */
265:      0x08, /* ----#---- */
266:      0x08, /* ----#---- */
267:      0x08, /* ----#---- */
268:      0x00, /* ----- */

```

drivers/video/font-lat9-8x12.c

Page 5/54

```

269:
270:          /* 0x13          */
271:    0x00, /* ----- */
272:    0x00, /* ----- */
273:    0x06, /* -----##- */
274:    0x0c, /* -----##- */
275:    0x18, /* ----##---- */
276:    0x30, /* --##----- */
277:    0x7e, /* -#####- */
278:    0x00, /* ----- */
279:    0x7e, /* -#####- */
280:    0x00, /* ----- */
281:    0x00, /* ----- */
282:    0x00, /* ----- */
283:
284:          /* 0x14          */
285:    0x00, /* ----- */
286:    0x00, /* ----- */
287:    0x60, /* -##----- */
288:    0x30, /* --##----- */
289:    0x18, /* ----##---- */
290:    0x0c, /* -----##- */
291:    0x7e, /* -#####- */
292:    0x00, /* ----- */
293:    0x7e, /* -#####- */
294:    0x00, /* ----- */
295:    0x00, /* ----- */
296:    0x00, /* ----- */
297:
298:          /* 0x15          */
299:    0x00, /* ----- */
300:    0x00, /* ----- */
301:    0x06, /* -----##- */
302:    0x0c, /* -----##- */
303:    0xfe, /* #####-#- */
304:    0x38, /* --##----- */
305:    0xfe, /* #####-#- */
306:    0x60, /* -##----- */
307:    0xc0, /* ##----- */
308:    0x00, /* ----- */
309:    0x00, /* ----- */
310:    0x00, /* ----- */
311:
312:          /* 0x16          */
313:    0x00, /* ----- */
314:    0x02, /* -----#- */
315:    0x0e, /* ----###- */
316:    0x3e, /* --#####- */
317:    0x7e, /* -#####- */
318:    0xfe, /* #####-#- */
319:    0x7e, /* -#####- */
320:    0x3e, /* --#####- */
321:    0x0e, /* ----###- */
322:    0x02, /* -----#- */
323:    0x00, /* ----- */
324:    0x00, /* ----- */
325:
326:          /* 0x17          */
327:    0x00, /* ----- */
328:    0x80, /* #----- */
329:    0xe0, /* ###----- */
330:    0xf0, /* ####----- */
331:    0xfc, /* #####--- */
332:    0xfe, /* #####-#- */
333:    0xfc, /* #####--- */
334:    0xf0, /* ####----- */
335:    0xe0, /* ###----- */

```

drivers/video/font-lat9-8x12.c

Page 6/54

```

336:      0x80, /* #----- */
337:      0x00, /* ----- */
338:      0x00, /* ----- */
339:
340:          /* 0x18 */
341:      0x00, /* ----- */
342:      0x18, /* ---##--- */
343:      0x3c, /* --####-- */
344:      0x7e, /* -#####- */
345:      0x18, /* ---##--- */
346:      0x18, /* ---##--- */
347:      0x18, /* ---##--- */
348:      0x18, /* ---##--- */
349:      0x18, /* ---##--- */
350:      0x18, /* ---##--- */
351:      0x00, /* ----- */
352:      0x00, /* ----- */
353:
354:          /* 0x19 */
355:      0x00, /* ----- */
356:      0x18, /* ---##--- */
357:      0x18, /* ---##--- */
358:      0x18, /* ---##--- */
359:      0x18, /* ---##--- */
360:      0x18, /* ---##--- */
361:      0x18, /* ---##--- */
362:      0x7e, /* -#####- */
363:      0x3c, /* --####-- */
364:      0x18, /* ---##--- */
365:      0x00, /* ----- */
366:      0x00, /* ----- */
367:
368:          /* 0x1a */
369:      0x00, /* ----- */
370:      0x00, /* ----- */
371:      0x00, /* ----- */
372:      0x18, /* ---##--- */
373:      0x0c, /* ----##-- */
374:      0xfe, /* #####-- */
375:      0x0c, /* ----##-- */
376:      0x18, /* ---##--- */
377:      0x00, /* ----- */
378:      0x00, /* ----- */
379:      0x00, /* ----- */
380:      0x00, /* ----- */
381:
382:          /* 0x1b */
383:      0x00, /* ----- */
384:      0x00, /* ----- */
385:      0x00, /* ----- */
386:      0x30, /* --##--- */
387:      0x60, /* -##----- */
388:      0xfe, /* #####-- */
389:      0x60, /* -##----- */
390:      0x30, /* --##--- */
391:      0x00, /* ----- */
392:      0x00, /* ----- */
393:      0x00, /* ----- */
394:      0x00, /* ----- */
395:
396:          /* 0x1c */
397:      0x00, /* ----- */
398:      0x18, /* ---##--- */
399:      0x3c, /* --####-- */
400:      0x7e, /* -#####- */
401:      0x18, /* ---##--- */
402:      0x18, /* ---##--- */

```


drivers/video/font-lat9-8x12.c

Page 7/54

```

403:      0x18, /* ----##---- */
404:      0x7e, /* -#####- */
405:      0x3c, /* --####-- */
406:      0x18, /* ----##---- */
407:      0x00, /* ----- */
408:      0x00, /* ----- */
409:
410:          /* 0x1d */
411:      0x00, /* ----- */
412:      0x00, /* ----- */
413:      0x00, /* ----- */
414:      0x28, /* --#-#-#- */
415:      0x6c, /* -##-##- */
416:      0xfe, /* #####- */
417:      0x6c, /* -##-##- */
418:      0x28, /* --#-#-#- */
419:      0x00, /* ----- */
420:      0x00, /* ----- */
421:      0x00, /* ----- */
422:      0x00, /* ----- */
423:
424:          /* 0x1e */
425:      0x00, /* ----- */
426:      0x06, /* -----##- */
427:      0x06, /* -----##- */
428:      0x36, /* --##-##- */
429:      0x66, /* -##-##- */
430:      0xfe, /* #####- */
431:      0x60, /* -##-#-#- */
432:      0x30, /* --##-#-#- */
433:      0x00, /* ----- */
434:      0x00, /* ----- */
435:      0x00, /* ----- */
436:      0x00, /* ----- */
437:
438:          /* 0x1f */
439:      0x00, /* ----- */
440:      0x00, /* ----- */
441:      0x00, /* ----- */
442:      0xc0, /* ##----- */
443:      0x7c, /* -#####- */
444:      0x6e, /* -##-##-#- */
445:      0x6c, /* -##-##-#- */
446:      0x6c, /* -##-##-#- */
447:      0x6c, /* -##-##-#- */
448:      0x00, /* ----- */
449:      0x00, /* ----- */
450:      0x00, /* ----- */
451:
452:          /* 0x20 (' ') */
453:      0x00, /* ----- */
454:      0x00, /* ----- */
455:      0x00, /* ----- */
456:      0x00, /* ----- */
457:      0x00, /* ----- */
458:      0x00, /* ----- */
459:      0x00, /* ----- */
460:      0x00, /* ----- */
461:      0x00, /* ----- */
462:      0x00, /* ----- */
463:      0x00, /* ----- */
464:      0x00, /* ----- */
465:
466:          /* 0x21 ('!') */
467:      0x00, /* ----- */
468:      0x18, /* ----##---- */
469:      0x3c, /* --####-- */

```

drivers/video/font-lat9-8x12.c

Page 8/54

```

470:      0x3c, /* ---###-- */
471:      0x3c, /* ---###-- */
472:      0x18, /* ----##---- */
473:      0x18, /* ----##---- */
474:      0x00, /* ----- */
475:      0x18, /* ----##---- */
476:      0x18, /* ----##---- */
477:      0x00, /* ----- */
478:      0x00, /* ----- */
479:
480:      /* 0x22 ('"') */
481:      0x00, /* ----- */
482:      0x36, /* --##-##- */
483:      0x36, /* --##-##- */
484:      0x14, /* ----##---- */
485:      0x00, /* ----- */
486:      0x00, /* ----- */
487:      0x00, /* ----- */
488:      0x00, /* ----- */
489:      0x00, /* ----- */
490:      0x00, /* ----- */
491:      0x00, /* ----- */
492:      0x00, /* ----- */
493:
494:      /* 0x23 ('#') */
495:      0x00, /* ----- */
496:      0x6c, /* -##-##- */
497:      0xfe, /* #####- */
498:      0x6c, /* -##-##- */
499:      0x6c, /* -##-##- */
500:      0xfe, /* #####- */
501:      0x6c, /* -##-##- */
502:      0x00, /* ----- */
503:      0x00, /* ----- */
504:      0x00, /* ----- */
505:      0x00, /* ----- */
506:      0x00, /* ----- */
507:
508:      /* 0x24 ('$') */
509:      0x00, /* ----- */
510:      0x10, /* ----#---- */
511:      0x7c, /* -#####- */
512:      0xd6, /* ##-##-##- */
513:      0x70, /* -###----- */
514:      0x38, /* --###----- */
515:      0x1c, /* ----###----- */
516:      0xd6, /* ##-##-##- */
517:      0x7c, /* -#####- */
518:      0x10, /* ----#---- */
519:      0x00, /* ----- */
520:      0x00, /* ----- */
521:
522:      /* 0x25 ('%') */
523:      0x00, /* ----- */
524:      0x00, /* ----- */
525:      0x00, /* ----- */
526:      0x62, /* -##---#- */
527:      0x66, /* -##---##- */
528:      0x0c, /* ----##----- */
529:      0x18, /* ----##----- */
530:      0x30, /* --##----- */
531:      0x66, /* -##---##- */
532:      0xc6, /* ##---##- */
533:      0x00, /* ----- */
534:      0x00, /* ----- */
535:
536:      /* 0x26 ('&') */

```

drivers/video/font-lat9-8x12.c

Page 9/54

```

537:      0x00, /* ----- */
538:      0x38, /* ---###--- */
539:      0x6c, /* -##-##-- */
540:      0x38, /* ---###--- */
541:      0x38, /* ---###--- */
542:      0x76, /* -###-##- */
543:      0xf6, /* #####-##- */
544:      0xce, /* ##--###- */
545:      0xcc, /* ##--##-- */
546:      0x76, /* -###-##- */
547:      0x00, /* ----- */
548:      0x00, /* ----- */
549:
550:          /* 0x27 ('''') */
551:      0x1c, /* ----###--- */
552:      0x1c, /* ----###--- */
553:      0x0c, /* -----##-- */
554:      0x18, /* ----##---- */
555:      0x00, /* ----- */
556:      0x00, /* ----- */
557:      0x00, /* ----- */
558:      0x00, /* ----- */
559:      0x00, /* ----- */
560:      0x00, /* ----- */
561:      0x00, /* ----- */
562:      0x00, /* ----- */
563:
564:          /* 0x28 ('(') */
565:      0x00, /* ----- */
566:      0x0c, /* -----##-- */
567:      0x18, /* ----##---- */
568:      0x30, /* --##----- */
569:      0x30, /* --##----- */
570:      0x30, /* --##----- */
571:      0x30, /* --##----- */
572:      0x30, /* --##----- */
573:      0x18, /* ----##---- */
574:      0x0c, /* -----##-- */
575:      0x00, /* ----- */
576:      0x00, /* ----- */
577:
578:          /* 0x29 ('') */
579:      0x00, /* ----- */
580:      0x30, /* --##----- */
581:      0x18, /* ----##---- */
582:      0x0c, /* -----##-- */
583:      0x0c, /* -----##-- */
584:      0x0c, /* -----##-- */
585:      0x0c, /* -----##-- */
586:      0x0c, /* -----##-- */
587:      0x18, /* ----##---- */
588:      0x30, /* --##----- */
589:      0x00, /* ----- */
590:      0x00, /* ----- */
591:
592:          /* 0x2a ('*') */
593:      0x00, /* ----- */
594:      0x00, /* ----- */
595:      0x00, /* ----- */
596:      0x6c, /* -##-##-- */
597:      0x38, /* ---###--- */
598:      0xf6, /* #####-##- */
599:      0x38, /* ---###--- */
600:      0x6c, /* -##-##-- */
601:      0x00, /* ----- */
602:      0x00, /* ----- */
603:      0x00, /* ----- */

```

drivers/video/font-lat9-8x12.c

Page 10/54

```

604:      0x00,    /* ----- */
605:
606:      /* 0x2b ('+') */
607:      0x00,    /* ----- */
608:      0x00,    /* ----- */
609:      0x00,    /* ----- */
610:      0x18,    /* ---##--- */
611:      0x18,    /* ---##--- */
612:      0x7e,    /* -#####- */
613:      0x18,    /* ---##--- */
614:      0x18,    /* ---##--- */
615:      0x00,    /* ----- */
616:      0x00,    /* ----- */
617:      0x00,    /* ----- */
618:      0x00,    /* ----- */
619:
620:      /* 0x2c (',') */
621:      0x00,    /* ----- */
622:      0x00,    /* ----- */
623:      0x00,    /* ----- */
624:      0x00,    /* ----- */
625:      0x00,    /* ----- */
626:      0x00,    /* ----- */
627:      0x00,    /* ----- */
628:      0x0c,    /* ----##-- */
629:      0x0c,    /* ----##-- */
630:      0x0c,    /* ----##-- */
631:      0x18,    /* ---##--- */
632:      0x00,    /* ----- */
633:
634:      /* 0x2d ('-') */
635:      0x00,    /* ----- */
636:      0x00,    /* ----- */
637:      0x00,    /* ----- */
638:      0x00,    /* ----- */
639:      0x00,    /* ----- */
640:      0xfe,    /* #####-#- */
641:      0x00,    /* ----- */
642:      0x00,    /* ----- */
643:      0x00,    /* ----- */
644:      0x00,    /* ----- */
645:      0x00,    /* ----- */
646:      0x00,    /* ----- */
647:
648:      /* 0x2e ('.') */
649:      0x00,    /* ----- */
650:      0x00,    /* ----- */
651:      0x00,    /* ----- */
652:      0x00,    /* ----- */
653:      0x00,    /* ----- */
654:      0x00,    /* ----- */
655:      0x00,    /* ----- */
656:      0x00,    /* ----- */
657:      0x18,    /* ---##--- */
658:      0x18,    /* ---##--- */
659:      0x00,    /* ----- */
660:      0x00,    /* ----- */
661:
662:      /* 0x2f ('/') */
663:      0x00,    /* ----- */
664:      0x00,    /* ----- */
665:      0x00,    /* ----- */
666:      0x06,    /* ----##-- */
667:      0x0c,    /* ----##-- */
668:      0x18,    /* ---##--- */
669:      0x30,    /* --##----- */
670:      0x60,    /* -##----- */

```

drivers/video/font-lat9-8x12.c

Page 11/54

```

671:      0xc0, /* ##----- */
672:      0x00, /* ----- */
673:      0x00, /* ----- */
674:      0x00, /* ----- */
675:
676:      /* 0x30 ('0') */
677:      0x00, /* ----- */
678:      0x7c, /* -#####- */
679:      0xc6, /* ##---##- */
680:      0xc6, /* ##---##- */
681:      0xc6, /* ##---##- */
682:      0xd6, /* #-#-##- */
683:      0xc6, /* ##---##- */
684:      0xc6, /* ##---##- */
685:      0xc6, /* ##---##- */
686:      0x7c, /* -#####- */
687:      0x00, /* ----- */
688:      0x00, /* ----- */
689:
690:      /* 0x31 ('1') */
691:      0x00, /* ----- */
692:      0x18, /* ---##--- */
693:      0x78, /* -#####- */
694:      0x18, /* ---##--- */
695:      0x18, /* ---##--- */
696:      0x18, /* ---##--- */
697:      0x18, /* ---##--- */
698:      0x18, /* ---##--- */
699:      0x18, /* ---##--- */
700:      0x7e, /* -#####- */
701:      0x00, /* ----- */
702:      0x00, /* ----- */
703:
704:      /* 0x32 ('2') */
705:      0x00, /* ----- */
706:      0x7c, /* -#####- */
707:      0xc6, /* ##---##- */
708:      0xc6, /* ##---##- */
709:      0x0c, /* ----##-- */
710:      0x18, /* ---##--- */
711:      0x30, /* --##----- */
712:      0x60, /* -##----- */
713:      0xc6, /* ##---##- */
714:      0xfe, /* #####--- */
715:      0x00, /* ----- */
716:      0x00, /* ----- */
717:
718:      /* 0x33 ('3') */
719:      0x00, /* ----- */
720:      0x7c, /* -#####- */
721:      0xc6, /* ##---##- */
722:      0x06, /* -----##- */
723:      0x06, /* -----##- */
724:      0x3c, /* --#####- */
725:      0x06, /* -----##- */
726:      0x06, /* -----##- */
727:      0xc6, /* ##---##- */
728:      0x7c, /* -#####- */
729:      0x00, /* ----- */
730:      0x00, /* ----- */
731:
732:      /* 0x34 ('4') */
733:      0x00, /* ----- */
734:      0x0c, /* ----##-- */
735:      0x1c, /* ---###--- */
736:      0x3c, /* --#####- */
737:      0x6c, /* -##-##- */

```

drivers/video/font-lat9-8x12.c

Page 12/54

```

738:      0xcc,    /* ##--##-- */
739:      0xfe,    /* #####- */
740:      0x0c,    /* ----##-- */
741:      0x0c,    /* ----##-- */
742:      0x0c,    /* ----##-- */
743:      0x00,    /* ----- */
744:      0x00,    /* ----- */
745:
746:          /* 0x35 ('5') */
747:      0x00,    /* ----- */
748:      0xfe,    /* #####- */
749:      0xc0,    /* ##----- */
750:      0xc0,    /* ##----- */
751:      0xc0,    /* ##----- */
752:      0xfc,    /* #####- */
753:      0x06,    /* ----##- */
754:      0x06,    /* ----##- */
755:      0xc6,    /* ##---##- */
756:      0x7c,    /* -#####- */
757:      0x00,    /* ----- */
758:      0x00,    /* ----- */
759:
760:          /* 0x36 ('6') */
761:      0x00,    /* ----- */
762:      0x7c,    /* -#####- */
763:      0xc6,    /* ##---##- */
764:      0xc0,    /* ##----- */
765:      0xc0,    /* ##----- */
766:      0xfc,    /* #####- */
767:      0xc6,    /* ##---##- */
768:      0xc6,    /* ##---##- */
769:      0xc6,    /* ##---##- */
770:      0x7c,    /* -#####- */
771:      0x00,    /* ----- */
772:      0x00,    /* ----- */
773:
774:          /* 0x37 ('7') */
775:      0x00,    /* ----- */
776:      0xfe,    /* #####- */
777:      0xc6,    /* ##---##- */
778:      0x0c,    /* ----##-- */
779:      0x18,    /* ---##---- */
780:      0x30,    /* --##----- */
781:      0x30,    /* --##----- */
782:      0x30,    /* --##----- */
783:      0x30,    /* --##----- */
784:      0x30,    /* --##----- */
785:      0x00,    /* ----- */
786:      0x00,    /* ----- */
787:
788:          /* 0x38 ('8') */
789:      0x00,    /* ----- */
790:      0x7c,    /* -#####- */
791:      0xc6,    /* ##---##- */
792:      0xc6,    /* ##---##- */
793:      0xc6,    /* ##---##- */
794:      0x7c,    /* -#####- */
795:      0xc6,    /* ##---##- */
796:      0xc6,    /* ##---##- */
797:      0xc6,    /* ##---##- */
798:      0x7c,    /* -#####- */
799:      0x00,    /* ----- */
800:      0x00,    /* ----- */
801:
802:          /* 0x39 ('9') */
803:      0x00,    /* ----- */
804:      0x7c,    /* -#####- */

```

drivers/video/font-lat9-8x12.c

Page 13/54

```

805:      0xc6, /* ##---##- */
806:      0xc6, /* ##---##- */
807:      0xc6, /* ##---##- */
808:      0x7e, /* -#####- */
809:      0x06, /* -----##- */
810:      0x06, /* -----##- */
811:      0xc6, /* ##---##- */
812:      0x7c, /* -#####- */
813:      0x00, /* ----- */
814:      0x00, /* ----- */
815:
816:          /* 0x3a (':') */
817:      0x00, /* ----- */
818:      0x00, /* ----- */
819:      0x00, /* ----- */
820:      0x0c, /* ----##- */
821:      0x0c, /* ----##- */
822:      0x00, /* ----- */
823:      0x00, /* ----- */
824:      0x0c, /* ----##- */
825:      0x0c, /* ----##- */
826:      0x00, /* ----- */
827:      0x00, /* ----- */
828:      0x00, /* ----- */
829:
830:          /* 0x3b (';') */
831:      0x00, /* ----- */
832:      0x00, /* ----- */
833:      0x00, /* ----- */
834:      0x0c, /* ----##- */
835:      0x0c, /* ----##- */
836:      0x00, /* ----- */
837:      0x00, /* ----- */
838:      0x0c, /* ----##- */
839:      0x0c, /* ----##- */
840:      0x0c, /* ----##- */
841:      0x18, /* ----##- */
842:      0x00, /* ----- */
843:
844:          /* 0x3c ('<') */
845:      0x00, /* ----- */
846:      0x0c, /* ----##- */
847:      0x18, /* ----##- */
848:      0x30, /* --##---- */
849:      0x60, /* -##----- */
850:      0xc0, /* ##----- */
851:      0x60, /* -##----- */
852:      0x30, /* --##---- */
853:      0x18, /* ----##- */
854:      0x0c, /* ----##- */
855:      0x00, /* ----- */
856:      0x00, /* ----- */
857:
858:          /* 0x3d ('=') */
859:      0x00, /* ----- */
860:      0x00, /* ----- */
861:      0x00, /* ----- */
862:      0x00, /* ----- */
863:      0xfe, /* #####- */
864:      0x00, /* ----- */
865:      0xfe, /* #####- */
866:      0x00, /* ----- */
867:      0x00, /* ----- */
868:      0x00, /* ----- */
869:      0x00, /* ----- */
870:      0x00, /* ----- */
871:

```

drivers/video/font-lat9-8x12.c

Page 14/54

```

872:                /* 0x3e ('>') */
873:    0x00,          /* ----- */
874:    0x60,          /* -##----- */
875:    0x30,          /* --##----- */
876:    0x18,          /* ----##----- */
877:    0x0c,          /* -----##-- */
878:    0x06,          /* -----##- */
879:    0x0c,          /* -----##-- */
880:    0x18,          /* ----##----- */
881:    0x30,          /* --##----- */
882:    0x60,          /* -##----- */
883:    0x00,          /* ----- */
884:    0x00,          /* ----- */
885:
886:                /* 0x3f ('?') */
887:    0x00,          /* ----- */
888:    0x7c,          /* -#####-- */
889:    0xc6,          /* ##---##- */
890:    0xc6,          /* ##---##- */
891:    0x0c,          /* -----##-- */
892:    0x18,          /* ----##----- */
893:    0x18,          /* ----##----- */
894:    0x00,          /* ----- */
895:    0x18,          /* ----##----- */
896:    0x18,          /* ----##----- */
897:    0x00,          /* ----- */
898:    0x00,          /* ----- */
899:
900:                /* 0x40 ('@') */
901:    0x00,          /* ----- */
902:    0x7c,          /* -#####-- */
903:    0xc6,          /* ##---##- */
904:    0xc6,          /* ##---##- */
905:    0xde,          /* ##-#####- */
906:    0xde,          /* ##-#####- */
907:    0xde,          /* ##-#####- */
908:    0xdc,          /* ##-#####- */
909:    0xc0,          /* ##----- */
910:    0x7e,          /* -#####-- */
911:    0x00,          /* ----- */
912:    0x00,          /* ----- */
913:
914:                /* 0x41 ('A') */
915:    0x00,          /* ----- */
916:    0x38,          /* --###----- */
917:    0x6c,          /* -##-##----- */
918:    0xc6,          /* ##---##- */
919:    0xc6,          /* ##---##- */
920:    0xc6,          /* ##---##- */
921:    0xfe,          /* #####--- */
922:    0xc6,          /* ##---##- */
923:    0xc6,          /* ##---##- */
924:    0xc6,          /* ##---##- */
925:    0x00,          /* ----- */
926:    0x00,          /* ----- */
927:
928:                /* 0x42 ('B') */
929:    0x00,          /* ----- */
930:    0xfc,          /* #####--- */
931:    0x66,          /* -##-##----- */
932:    0x66,          /* -##-##----- */
933:    0x66,          /* -##-##----- */
934:    0x7c,          /* -#####-- */
935:    0x66,          /* -##-##----- */
936:    0x66,          /* -##-##----- */
937:    0x66,          /* -##-##----- */
938:    0xfc,          /* #####---

```


drivers/video/font-lat9-8x12.c

Page 15/54

```

939:      0x00, /* ----- */
940:      0x00, /* ----- */
941:
942:      /* 0x43 ('C') */
943:      0x00, /* ----- */
944:      0x3c, /* ---####- */
945:      0x66, /* -##--##- */
946:      0xc0, /* ##----- */
947:      0xc0, /* ##----- */
948:      0xc0, /* ##----- */
949:      0xc0, /* ##----- */
950:      0xc0, /* ##----- */
951:      0x66, /* -##--##- */
952:      0x3c, /* ---####- */
953:      0x00, /* ----- */
954:      0x00, /* ----- */
955:
956:      /* 0x44 ('D') */
957:      0x00, /* ----- */
958:      0xf8, /* #####--- */
959:      0x6c, /* -##-##-- */
960:      0x66, /* -##-##-- */
961:      0x66, /* -##-##-- */
962:      0x66, /* -##-##-- */
963:      0x66, /* -##-##-- */
964:      0x66, /* -##-##-- */
965:      0x6c, /* -##-##-- */
966:      0xf8, /* #####--- */
967:      0x00, /* ----- */
968:      0x00, /* ----- */
969:
970:      /* 0x45 ('E') */
971:      0x00, /* ----- */
972:      0xfe, /* #####--- */
973:      0x66, /* -##-##-- */
974:      0x60, /* -##----- */
975:      0x60, /* -##----- */
976:      0x7c, /* -#####-- */
977:      0x60, /* -##----- */
978:      0x60, /* -##----- */
979:      0x66, /* -##-##-- */
980:      0xfe, /* #####--- */
981:      0x00, /* ----- */
982:      0x00, /* ----- */
983:
984:      /* 0x46 ('F') */
985:      0x00, /* ----- */
986:      0xfe, /* #####--- */
987:      0x66, /* -##-##-- */
988:      0x60, /* -##----- */
989:      0x60, /* -##----- */
990:      0x7c, /* -#####-- */
991:      0x60, /* -##----- */
992:      0x60, /* -##----- */
993:      0x60, /* -##----- */
994:      0xf0, /* ####---- */
995:      0x00, /* ----- */
996:      0x00, /* ----- */
997:
998:      /* 0x47 ('G') */
999:      0x00, /* ----- */
1000:     0x7c, /* -#####-- */
1001:     0xc6, /* ##----##- */
1002:     0xc6, /* ##----##- */
1003:     0xc0, /* ##----- */
1004:     0xc0, /* ##----- */
1005:     0xce, /* ##--###- */

```

drivers/video/font-lat9-8x12.c

Page 16/54

```

1006:      0xc6, /* ##---##- */
1007:      0xc6, /* ##---##- */
1008:      0x7c, /* -#####- */
1009:      0x00, /* ----- */
1010:      0x00, /* ----- */
1011:
1012:          /* 0x48 ('H') */
1013:      0x00, /* ----- */
1014:      0xc6, /* ##---##- */
1015:      0xc6, /* ##---##- */
1016:      0xc6, /* ##---##- */
1017:      0xc6, /* ##---##- */
1018:      0xfe, /* #####- */
1019:      0xc6, /* ##---##- */
1020:      0xc6, /* ##---##- */
1021:      0xc6, /* ##---##- */
1022:      0xc6, /* ##---##- */
1023:      0x00, /* ----- */
1024:      0x00, /* ----- */
1025:
1026:          /* 0x49 ('I') */
1027:      0x00, /* ----- */
1028:      0x3c, /* --#####- */
1029:      0x18, /* ---##--- */
1030:      0x18, /* ---##--- */
1031:      0x18, /* ---##--- */
1032:      0x18, /* ---##--- */
1033:      0x18, /* ---##--- */
1034:      0x18, /* ---##--- */
1035:      0x18, /* ---##--- */
1036:      0x3c, /* --#####- */
1037:      0x00, /* ----- */
1038:      0x00, /* ----- */
1039:
1040:          /* 0x4a ('J') */
1041:      0x00, /* ----- */
1042:      0x3c, /* --#####- */
1043:      0x18, /* ---##--- */
1044:      0x18, /* ---##--- */
1045:      0x18, /* ---##--- */
1046:      0x18, /* ---##--- */
1047:      0x18, /* ---##--- */
1048:      0xd8, /* ##-##--- */
1049:      0xd8, /* ##-##--- */
1050:      0x70, /* -###----- */
1051:      0x00, /* ----- */
1052:      0x00, /* ----- */
1053:
1054:          /* 0x4b ('K') */
1055:      0x00, /* ----- */
1056:      0xc6, /* ##---##- */
1057:      0xcc, /* ##-##--- */
1058:      0xd8, /* ##-##--- */
1059:      0xf0, /* #####----- */
1060:      0xf0, /* #####----- */
1061:      0xd8, /* ##-##--- */
1062:      0xcc, /* ##-##--- */
1063:      0xc6, /* ##---##- */
1064:      0xc6, /* ##---##- */
1065:      0x00, /* ----- */
1066:      0x00, /* ----- */
1067:
1068:          /* 0x4c ('L') */
1069:      0x00, /* ----- */
1070:      0xf0, /* #####----- */
1071:      0x60, /* -##----- */
1072:      0x60, /* -##----- */

```

drivers/video/font-lat9-8x12.c

Page 17/54

```

1073:      0x60, /* -##----- */
1074:      0x60, /* -##----- */
1075:      0x60, /* -##----- */
1076:      0x62, /* -##---#- */
1077:      0x66, /* -##--##- */
1078:      0xfe, /* #####- */
1079:      0x00, /* ----- */
1080:      0x00, /* ----- */
1081:
1082:          /* 0x4d ('M') */
1083:      0x00, /* ----- */
1084:      0xc6, /* ##---##- */
1085:      0xc6, /* ##---##- */
1086:      0xee, /* ###-###- */
1087:      0xfe, /* #####- */
1088:      0xd6, /* #-#-##- */
1089:      0xd6, /* #-#-##- */
1090:      0xd6, /* #-#-##- */
1091:      0xc6, /* ##---##- */
1092:      0xc6, /* ##---##- */
1093:      0x00, /* ----- */
1094:      0x00, /* ----- */
1095:
1096:          /* 0x4e ('N') */
1097:      0x00, /* ----- */
1098:      0xc6, /* ##---##- */
1099:      0xc6, /* ##---##- */
1100:      0xe6, /* ###-##- */
1101:      0xe6, /* ###-##- */
1102:      0xf6, /* ####-##- */
1103:      0xde, /* #-#####- */
1104:      0xce, /* #-#####- */
1105:      0xce, /* #-#####- */
1106:      0xc6, /* ##---##- */
1107:      0x00, /* ----- */
1108:      0x00, /* ----- */
1109:
1110:          /* 0x4f ('O') */
1111:      0x00, /* ----- */
1112:      0x7c, /* -#####- */
1113:      0xc6, /* ##---##- */
1114:      0xc6, /* ##---##- */
1115:      0xc6, /* ##---##- */
1116:      0xc6, /* ##---##- */
1117:      0xc6, /* ##---##- */
1118:      0xc6, /* ##---##- */
1119:      0xc6, /* ##---##- */
1120:      0x7c, /* -#####- */
1121:      0x00, /* ----- */
1122:      0x00, /* ----- */
1123:
1124:          /* 0x50 ('P') */
1125:      0x00, /* ----- */
1126:      0xfc, /* #####- */
1127:      0x66, /* -##--##- */
1128:      0x66, /* -##--##- */
1129:      0x66, /* -##--##- */
1130:      0x7c, /* -#####- */
1131:      0x60, /* -##----- */
1132:      0x60, /* -##----- */
1133:      0x60, /* -##----- */
1134:      0xf0, /* ####---- */
1135:      0x00, /* ----- */
1136:      0x00, /* ----- */
1137:
1138:          /* 0x51 ('Q') */
1139:      0x00, /* ----- */

```

drivers/video/font-lat9-8x12.c

Page 18/54

```

1140:      0x7c, /* #####-- */
1141:      0xc6, /* ##----##- */
1142:      0xc6, /* ##----##- */
1143:      0xc6, /* ##----##- */
1144:      0xc6, /* ##----##- */
1145:      0xc6, /* ##----##- */
1146:      0xc6, /* ##----##- */
1147:      0xd6, /* ##-##-##- */
1148:      0x7c, /* #####-- */
1149:      0x06, /* -----##- */
1150:      0x00, /* ----- */
1151:
1152:          /* 0x52 ('R') */
1153:      0x00, /* ----- */
1154:      0xfc, /* #####-- */
1155:      0x66, /* -##--##- */
1156:      0x66, /* -##--##- */
1157:      0x66, /* -##--##- */
1158:      0x7c, /* #####-- */
1159:      0x78, /* #####-- */
1160:      0x6c, /* -##-##-- */
1161:      0x66, /* -##-##-- */
1162:      0xe6, /* ###--##- */
1163:      0x00, /* ----- */
1164:      0x00, /* ----- */
1165:
1166:          /* 0x53 ('S') */
1167:      0x00, /* ----- */
1168:      0x7c, /* #####-- */
1169:      0xc6, /* ##----##- */
1170:      0xc0, /* ##----- */
1171:      0x60, /* -##----- */
1172:      0x38, /* --###---- */
1173:      0x0c, /* -----##- */
1174:      0x06, /* -----##- */
1175:      0xc6, /* ##----##- */
1176:      0x7c, /* #####-- */
1177:      0x00, /* ----- */
1178:      0x00, /* ----- */
1179:
1180:          /* 0x54 ('T') */
1181:      0x00, /* ----- */
1182:      0x7e, /* #######- */
1183:      0x5a, /* -##-##-#- */
1184:      0x18, /* ---##---- */
1185:      0x18, /* ---##---- */
1186:      0x18, /* ---##---- */
1187:      0x18, /* ---##---- */
1188:      0x18, /* ---##---- */
1189:      0x18, /* ---##---- */
1190:      0x3c, /* --####-- */
1191:      0x00, /* ----- */
1192:      0x00, /* ----- */
1193:
1194:          /* 0x55 ('U') */
1195:      0x00, /* ----- */
1196:      0xc6, /* ##----##- */
1197:      0xc6, /* ##----##- */
1198:      0xc6, /* ##----##- */
1199:      0xc6, /* ##----##- */
1200:      0xc6, /* ##----##- */
1201:      0xc6, /* ##----##- */
1202:      0xc6, /* ##----##- */
1203:      0xc6, /* ##----##- */
1204:      0x7c, /* #####-- */
1205:      0x00, /* ----- */
1206:      0x00, /* ----- */

```

drivers/video/font-lat9-8x12.c

Page 19/54

```

1207:
1208:          /* 0x56 ('V') */
1209:    0x00,  /* ----- */
1210:    0xc6,  /* ##---##- */
1211:    0xc6,  /* ##---##- */
1212:    0xc6,  /* ##---##- */
1213:    0xc6,  /* ##---##- */
1214:    0xc6,  /* ##---##- */
1215:    0xc6,  /* ##---##- */
1216:    0x6c,  /* -##-##-- */
1217:    0x38,  /* --###---- */
1218:    0x10,  /* ---#----- */
1219:    0x00,  /* ----- */
1220:    0x00,  /* ----- */
1221:
1222:          /* 0x57 ('W') */
1223:    0x00,  /* ----- */
1224:    0xc6,  /* ##---##- */
1225:    0xc6,  /* ##---##- */
1226:    0xd6,  /* ##-##-##- */
1227:    0xd6,  /* ##-##-##- */
1228:    0xd6,  /* ##-##-##- */
1229:    0xfe,  /* #####-#- */
1230:    0xee,  /* ##-###-#- */
1231:    0xc6,  /* ##---##- */
1232:    0xc6,  /* ##---##- */
1233:    0x00,  /* ----- */
1234:    0x00,  /* ----- */
1235:
1236:          /* 0x58 ('X') */
1237:    0x00,  /* ----- */
1238:    0xc6,  /* ##---##- */
1239:    0xc6,  /* ##---##- */
1240:    0x6c,  /* -##-##-- */
1241:    0x38,  /* --###---- */
1242:    0x38,  /* --###---- */
1243:    0x38,  /* --###---- */
1244:    0x6c,  /* -##-##-- */
1245:    0xc6,  /* ##---##- */
1246:    0xc6,  /* ##---##- */
1247:    0x00,  /* ----- */
1248:    0x00,  /* ----- */
1249:
1250:          /* 0x59 ('Y') */
1251:    0x00,  /* ----- */
1252:    0x66,  /* -##--##- */
1253:    0x66,  /* -##--##- */
1254:    0x66,  /* -##--##- */
1255:    0x66,  /* -##--##- */
1256:    0x3c,  /* --####-#- */
1257:    0x18,  /* ---##---- */
1258:    0x18,  /* ---##---- */
1259:    0x18,  /* ---##---- */
1260:    0x3c,  /* --####-#- */
1261:    0x00,  /* ----- */
1262:    0x00,  /* ----- */
1263:
1264:          /* 0x5a ('Z') */
1265:    0x00,  /* ----- */
1266:    0xfe,  /* #####-#- */
1267:    0xc6,  /* ##---##- */
1268:    0x8c,  /* #---##-- */
1269:    0x18,  /* ---##---- */
1270:    0x30,  /* --##----- */
1271:    0x60,  /* -##----- */
1272:    0xc2,  /* ##-----#- */
1273:    0xc6,  /* ##---##- */

```

drivers/video/font-lat9-8x12.c

Page 20/54

```

1274:      0xfe, /* #####- */
1275:      0x00, /* ----- */
1276:      0x00, /* ----- */
1277:
1278:          /* 0x5b ('[') */
1279:      0x00, /* ----- */
1280:      0x7c, /* -#####- */
1281:      0x60, /* -##----- */
1282:      0x60, /* -##----- */
1283:      0x60, /* -##----- */
1284:      0x60, /* -##----- */
1285:      0x60, /* -##----- */
1286:      0x60, /* -##----- */
1287:      0x60, /* -##----- */
1288:      0x7c, /* -#####- */
1289:      0x00, /* ----- */
1290:      0x00, /* ----- */
1291:
1292:          /* 0x5c ('\') */
1293:      0x00, /* ----- */
1294:      0x00, /* ----- */
1295:      0x00, /* ----- */
1296:      0xc0, /* ##----- */
1297:      0x60, /* -##----- */
1298:      0x30, /* --##----- */
1299:      0x18, /* ----##---- */
1300:      0x0c, /* -----##-- */
1301:      0x06, /* -----##- */
1302:      0x00, /* ----- */
1303:      0x00, /* ----- */
1304:      0x00, /* ----- */
1305:
1306:          /* 0x5d (']') */
1307:      0x00, /* ----- */
1308:      0x7c, /* -#####- */
1309:      0x0c, /* ----##-- */
1310:      0x0c, /* ----##-- */
1311:      0x0c, /* ----##-- */
1312:      0x0c, /* ----##-- */
1313:      0x0c, /* ----##-- */
1314:      0x0c, /* ----##-- */
1315:      0x0c, /* ----##-- */
1316:      0x7c, /* -#####- */
1317:      0x00, /* ----- */
1318:      0x00, /* ----- */
1319:
1320:          /* 0x5e ('^') */
1321:      0x00, /* ----- */
1322:      0x18, /* ----##---- */
1323:      0x3c, /* --#####- */
1324:      0x66, /* -##--##- */
1325:      0x00, /* ----- */
1326:      0x00, /* ----- */
1327:      0x00, /* ----- */
1328:      0x00, /* ----- */
1329:      0x00, /* ----- */
1330:      0x00, /* ----- */
1331:      0x00, /* ----- */
1332:      0x00, /* ----- */
1333:
1334:          /* 0x5f ('_') */
1335:      0x00, /* ----- */
1336:      0x00, /* ----- */
1337:      0x00, /* ----- */
1338:      0x00, /* ----- */
1339:      0x00, /* ----- */
1340:      0x00, /* ----- */

```

drivers/video/font-lat9-8x12.c

Page 21/54

```

1341:      0x00, /* ----- */
1342:      0x00, /* ----- */
1343:      0x00, /* ----- */
1344:      0x00, /* ----- */
1345:      0x00, /* ----- */
1346:      0xff, /* ##### */
1347:
1348:          /* 0x60 ('') */
1349:      0x1c, /* ---###-- */
1350:      0x1c, /* ---###-- */
1351:      0x18, /* ---##--- */
1352:      0x0c, /* ----##-- */
1353:      0x00, /* ----- */
1354:      0x00, /* ----- */
1355:      0x00, /* ----- */
1356:      0x00, /* ----- */
1357:      0x00, /* ----- */
1358:      0x00, /* ----- */
1359:      0x00, /* ----- */
1360:      0x00, /* ----- */
1361:
1362:          /* 0x61 ('a') */
1363:      0x00, /* ----- */
1364:      0x00, /* ----- */
1365:      0x00, /* ----- */
1366:      0x00, /* ----- */
1367:      0x78, /* -####--- */
1368:      0x0c, /* ----##-- */
1369:      0x7c, /* -#####-- */
1370:      0xcc, /* ##---##-- */
1371:      0xdc, /* ##-###-- */
1372:      0x76, /* -###-##- */
1373:      0x00, /* ----- */
1374:      0x00, /* ----- */
1375:
1376:          /* 0x62 ('b') */
1377:      0x00, /* ----- */
1378:      0xe0, /* ###----- */
1379:      0x60, /* -##----- */
1380:      0x60, /* -##----- */
1381:      0x7c, /* -#####-- */
1382:      0x66, /* -##---##- */
1383:      0x66, /* -##---##- */
1384:      0x66, /* -##---##- */
1385:      0x66, /* -##---##- */
1386:      0xfc, /* #####--- */
1387:      0x00, /* ----- */
1388:      0x00, /* ----- */
1389:
1390:          /* 0x63 ('c') */
1391:      0x00, /* ----- */
1392:      0x00, /* ----- */
1393:      0x00, /* ----- */
1394:      0x00, /* ----- */
1395:      0x7c, /* -#####-- */
1396:      0xc6, /* ##---##- */
1397:      0xc0, /* ##----- */
1398:      0xc0, /* ##----- */
1399:      0xc6, /* ##---##- */
1400:      0x7c, /* -#####-- */
1401:      0x00, /* ----- */
1402:      0x00, /* ----- */
1403:
1404:          /* 0x64 ('d') */
1405:      0x00, /* ----- */
1406:      0x1c, /* ---###-- */
1407:      0x0c, /* ----##-- */

```

drivers/video/font-lat9-8x12.c

Page 22/54

```

1408:      0x0c, /* ----##-- */
1409:      0x7c, /* -#####-- */
1410:      0xcc, /* ##--##-- */
1411:      0xcc, /* ##--##-- */
1412:      0xcc, /* ##--##-- */
1413:      0xcc, /* ##--##-- */
1414:      0x7e, /* -#####-- */
1415:      0x00, /* ----- */
1416:      0x00, /* ----- */
1417:
1418:          /* 0x65 ('e') */
1419:      0x00, /* ----- */
1420:      0x00, /* ----- */
1421:      0x00, /* ----- */
1422:      0x00, /* ----- */
1423:      0x7c, /* -#####-- */
1424:      0xc6, /* ##--##-- */
1425:      0xfe, /* #####--- */
1426:      0xc0, /* ##----- */
1427:      0xc6, /* ##--##-- */
1428:      0x7c, /* -#####-- */
1429:      0x00, /* ----- */
1430:      0x00, /* ----- */
1431:
1432:          /* 0x66 ('f') */
1433:      0x00, /* ----- */
1434:      0x1c, /* ---###-- */
1435:      0x36, /* --##-##- */
1436:      0x30, /* --##----- */
1437:      0x30, /* --##----- */
1438:      0xfc, /* #####--- */
1439:      0x30, /* --##----- */
1440:      0x30, /* --##----- */
1441:      0x30, /* --##----- */
1442:      0x78, /* -#####-- */
1443:      0x00, /* ----- */
1444:      0x00, /* ----- */
1445:
1446:          /* 0x67 ('g') */
1447:      0x00, /* ----- */
1448:      0x00, /* ----- */
1449:      0x00, /* ----- */
1450:      0x00, /* ----- */
1451:      0x76, /* -###-##- */
1452:      0xce, /* ##--##-- */
1453:      0xc6, /* ##--##-- */
1454:      0xc6, /* ##--##-- */
1455:      0x7e, /* -#####-- */
1456:      0x06, /* -----##- */
1457:      0xc6, /* ##--##-- */
1458:      0x7c, /* -#####-- */
1459:
1460:          /* 0x68 ('h') */
1461:      0x00, /* ----- */
1462:      0xe0, /* ###----- */
1463:      0x60, /* -##----- */
1464:      0x60, /* -##----- */
1465:      0x6c, /* -##-##-- */
1466:      0x76, /* -##-##-- */
1467:      0x66, /* -##-##-- */
1468:      0x66, /* -##-##-- */
1469:      0x66, /* -##-##-- */
1470:      0xe6, /* ###-##-- */
1471:      0x00, /* ----- */
1472:      0x00, /* ----- */
1473:
1474:          /* 0x69 ('i') */

```


drivers/video/font-lat9-8x12.c

Page 23/54

```

1475:      0x00, /* ----- */
1476:      0x18, /* ----##---- */
1477:      0x18, /* ----##---- */
1478:      0x00, /* ----- */
1479:      0x38, /* ---###---- */
1480:      0x18, /* ----##---- */
1481:      0x18, /* ----##---- */
1482:      0x18, /* ----##---- */
1483:      0x18, /* ----##---- */
1484:      0x3c, /* ---####--- */
1485:      0x00, /* ----- */
1486:      0x00, /* ----- */
1487:
1488:          /* 0x6a ('j') */
1489:      0x00, /* ----- */
1490:      0x00, /* ----- */
1491:      0x0c, /* ----##---- */
1492:      0x0c, /* ----##---- */
1493:      0x00, /* ----- */
1494:      0x1c, /* ----###--- */
1495:      0x0c, /* ----##---- */
1496:      0x0c, /* ----##---- */
1497:      0x0c, /* ----##---- */
1498:      0xcc, /* ##--##-- */
1499:      0xcc, /* ##--##-- */
1500:      0x78, /* -####--- */
1501:
1502:          /* 0x6b ('k') */
1503:      0x00, /* ----- */
1504:      0xe0, /* ###----- */
1505:      0x60, /* -##----- */
1506:      0x60, /* -##----- */
1507:      0x66, /* -##--##-- */
1508:      0x6c, /* -##--##-- */
1509:      0x78, /* -####--- */
1510:      0x6c, /* -##--##-- */
1511:      0x66, /* -##--##-- */
1512:      0xe6, /* ###--##-- */
1513:      0x00, /* ----- */
1514:      0x00, /* ----- */
1515:
1516:          /* 0x6c ('l') */
1517:      0x00, /* ----- */
1518:      0x38, /* --###---- */
1519:      0x18, /* ----##---- */
1520:      0x18, /* ----##---- */
1521:      0x18, /* ----##---- */
1522:      0x18, /* ----##---- */
1523:      0x18, /* ----##---- */
1524:      0x18, /* ----##---- */
1525:      0x18, /* ----##---- */
1526:      0x3c, /* ---####--- */
1527:      0x00, /* ----- */
1528:      0x00, /* ----- */
1529:
1530:          /* 0x6d ('m') */
1531:      0x00, /* ----- */
1532:      0x00, /* ----- */
1533:      0x00, /* ----- */
1534:      0x00, /* ----- */
1535:      0x6c, /* -##--##-- */
1536:      0xfe, /* #####--- */
1537:      0xd6, /* ##-##-##- */
1538:      0xd6, /* ##-##-##- */
1539:      0xc6, /* ##---##- */
1540:      0xc6, /* ##---##- */
1541:      0x00, /* ----- */

```

drivers/video/font-lat9-8x12.c

Page 24/54

```

1542:      0x00,    /* ----- */
1543:
1544:      /* 0x6e ('n') */
1545:      0x00,    /* ----- */
1546:      0x00,    /* ----- */
1547:      0x00,    /* ----- */
1548:      0x00,    /* ----- */
1549:      0xdc,    /* ##-##-# */
1550:      0x66,    /* -##-##-# */
1551:      0x66,    /* -##-##-# */
1552:      0x66,    /* -##-##-# */
1553:      0x66,    /* -##-##-# */
1554:      0x66,    /* -##-##-# */
1555:      0x00,    /* ----- */
1556:      0x00,    /* ----- */
1557:
1558:      /* 0x6f ('o') */
1559:      0x00,    /* ----- */
1560:      0x00,    /* ----- */
1561:      0x00,    /* ----- */
1562:      0x00,    /* ----- */
1563:      0x7c,    /* -#####- */
1564:      0xc6,    /* ##-##-# */
1565:      0xc6,    /* ##-##-# */
1566:      0xc6,    /* ##-##-# */
1567:      0xc6,    /* ##-##-# */
1568:      0x7c,    /* -#####- */
1569:      0x00,    /* ----- */
1570:      0x00,    /* ----- */
1571:
1572:      /* 0x70 ('p') */
1573:      0x00,    /* ----- */
1574:      0x00,    /* ----- */
1575:      0x00,    /* ----- */
1576:      0x00,    /* ----- */
1577:      0xdc,    /* ##-##-# */
1578:      0x66,    /* -##-##-# */
1579:      0x66,    /* -##-##-# */
1580:      0x66,    /* -##-##-# */
1581:      0x7c,    /* -#####- */
1582:      0x60,    /* -##-##-# */
1583:      0x60,    /* -##-##-# */
1584:      0xf0,    /* #####-# */
1585:
1586:      /* 0x71 ('q') */
1587:      0x00,    /* ----- */
1588:      0x00,    /* ----- */
1589:      0x00,    /* ----- */
1590:      0x00,    /* ----- */
1591:      0x76,    /* -###-##-# */
1592:      0xcc,    /* ##-##-# */
1593:      0xcc,    /* ##-##-# */
1594:      0xcc,    /* ##-##-# */
1595:      0x7c,    /* -#####- */
1596:      0x0c,    /* ----##-# */
1597:      0x0c,    /* ----##-# */
1598:      0x1e,    /* ---#####- */
1599:
1600:      /* 0x72 ('r') */
1601:      0x00,    /* ----- */
1602:      0x00,    /* ----- */
1603:      0x00,    /* ----- */
1604:      0x00,    /* ----- */
1605:      0xdc,    /* ##-##-# */
1606:      0x66,    /* -##-##-# */
1607:      0x60,    /* -##-##-# */
1608:      0x60,    /* -##-##-# */

```

drivers/video/font-lat9-8x12.c

Page 25/54

```

1609:      0x60,    /* -##----- */
1610:      0xf0,    /* ###----- */
1611:      0x00,    /* ----- */
1612:      0x00,    /* ----- */
1613:
1614:          /* 0x73 ('s') */
1615:      0x00,    /* ----- */
1616:      0x00,    /* ----- */
1617:      0x00,    /* ----- */
1618:      0x00,    /* ----- */
1619:      0x7c,    /* -#####-- */
1620:      0xc6,    /* ##---##- */
1621:      0x70,    /* -##----- */
1622:      0x1c,    /* ----###-- */
1623:      0xc6,    /* ##---##- */
1624:      0x7c,    /* -#####-- */
1625:      0x00,    /* ----- */
1626:      0x00,    /* ----- */
1627:
1628:          /* 0x74 ('t') */
1629:      0x00,    /* ----- */
1630:      0x30,    /* --##----- */
1631:      0x30,    /* --##----- */
1632:      0x30,    /* --##----- */
1633:      0xfc,    /* #####--- */
1634:      0x30,    /* --##----- */
1635:      0x30,    /* --##----- */
1636:      0x30,    /* --##----- */
1637:      0x36,    /* --##-##- */
1638:      0x1c,    /* ----###-- */
1639:      0x00,    /* ----- */
1640:      0x00,    /* ----- */
1641:
1642:          /* 0x75 ('u') */
1643:      0x00,    /* ----- */
1644:      0x00,    /* ----- */
1645:      0x00,    /* ----- */
1646:      0x00,    /* ----- */
1647:      0xcc,    /* ##---##- */
1648:      0xcc,    /* ##---##- */
1649:      0xcc,    /* ##---##- */
1650:      0xcc,    /* ##---##- */
1651:      0xcc,    /* ##---##- */
1652:      0x76,    /* -##-##- */
1653:      0x00,    /* ----- */
1654:      0x00,    /* ----- */
1655:
1656:          /* 0x76 ('v') */
1657:      0x00,    /* ----- */
1658:      0x00,    /* ----- */
1659:      0x00,    /* ----- */
1660:      0x00,    /* ----- */
1661:      0xc6,    /* ##---##- */
1662:      0xc6,    /* ##---##- */
1663:      0xc6,    /* ##---##- */
1664:      0x6c,    /* -##-##- */
1665:      0x38,    /* --###----- */
1666:      0x10,    /* ---#----- */
1667:      0x00,    /* ----- */
1668:      0x00,    /* ----- */
1669:
1670:          /* 0x77 ('w') */
1671:      0x00,    /* ----- */
1672:      0x00,    /* ----- */
1673:      0x00,    /* ----- */
1674:      0x00,    /* ----- */
1675:      0xc6,    /* ##---##- */

```

drivers/video/font-lat9-8x12.c

Page 26/54

```

1676:      0xc6,    /* ##---##- */
1677:      0xd6,    /* #-#-##- */
1678:      0xd6,    /* #-#-##- */
1679:      0xfe,    /* #####- */
1680:      0x6c,    /* -##-##- */
1681:      0x00,    /* ----- */
1682:      0x00,    /* ----- */
1683:
1684:          /* 0x78 ('x') */
1685:      0x00,    /* ----- */
1686:      0x00,    /* ----- */
1687:      0x00,    /* ----- */
1688:      0x00,    /* ----- */
1689:      0xc6,    /* ##---##- */
1690:      0x6c,    /* -##-##- */
1691:      0x38,    /* ---###--- */
1692:      0x38,    /* ---###--- */
1693:      0x6c,    /* -##-##- */
1694:      0xc6,    /* ##---##- */
1695:      0x00,    /* ----- */
1696:      0x00,    /* ----- */
1697:
1698:          /* 0x79 ('y') */
1699:      0x00,    /* ----- */
1700:      0x00,    /* ----- */
1701:      0x00,    /* ----- */
1702:      0x00,    /* ----- */
1703:      0xc6,    /* ##---##- */
1704:      0xc6,    /* ##---##- */
1705:      0xc6,    /* ##---##- */
1706:      0xce,    /* ##---##- */
1707:      0x76,    /* -###-##- */
1708:      0x06,    /* -----##- */
1709:      0xc6,    /* ##---##- */
1710:      0x7c,    /* -#####- */
1711:
1712:          /* 0x7a ('z') */
1713:      0x00,    /* ----- */
1714:      0x00,    /* ----- */
1715:      0x00,    /* ----- */
1716:      0x00,    /* ----- */
1717:      0xfe,    /* #####- */
1718:      0x8c,    /* #---##- */
1719:      0x18,    /* ---##--- */
1720:      0x30,    /* --##---- */
1721:      0x62,    /* -##-##- */
1722:      0xfe,    /* #####- */
1723:      0x00,    /* ----- */
1724:      0x00,    /* ----- */
1725:
1726:          /* 0x7b ('{') */
1727:      0x00,    /* ----- */
1728:      0x0e,    /* ----###- */
1729:      0x18,    /* ---##--- */
1730:      0x18,    /* ---##--- */
1731:      0x18,    /* ---##--- */
1732:      0x70,    /* -###---- */
1733:      0x18,    /* ---##--- */
1734:      0x18,    /* ---##--- */
1735:      0x18,    /* ---##--- */
1736:      0x0e,    /* ----###- */
1737:      0x00,    /* ----- */
1738:      0x00,    /* ----- */
1739:
1740:          /* 0x7c ('|') */
1741:      0x00,    /* ----- */
1742:      0x18,    /* ---##--- */

```

drivers/video/font-lat9-8x12.c

Page 27/54

```

1743:      0x18, /* ----##---- */
1744:      0x18, /* ----##---- */
1745:      0x18, /* ----##---- */
1746:      0x18, /* ----##---- */
1747:      0x18, /* ----##---- */
1748:      0x18, /* ----##---- */
1749:      0x18, /* ----##---- */
1750:      0x18, /* ----##---- */
1751:      0x00, /* ----- */
1752:      0x00, /* ----- */
1753:
1754:          /* 0x7d ('}') */
1755:      0x00, /* ----- */
1756:      0x70, /* -###----- */
1757:      0x18, /* ----##---- */
1758:      0x18, /* ----##---- */
1759:      0x18, /* ----##---- */
1760:      0x0e, /* ----###- */
1761:      0x18, /* ----##---- */
1762:      0x18, /* ----##---- */
1763:      0x18, /* ----##---- */
1764:      0x70, /* -###----- */
1765:      0x00, /* ----- */
1766:      0x00, /* ----- */
1767:
1768:          /* 0x7e (~') */
1769:      0x00, /* ----- */
1770:      0x76, /* -###-##- */
1771:      0xdc, /* ##-###- */
1772:      0x00, /* ----- */
1773:      0x00, /* ----- */
1774:      0x00, /* ----- */
1775:      0x00, /* ----- */
1776:      0x00, /* ----- */
1777:      0x00, /* ----- */
1778:      0x00, /* ----- */
1779:      0x00, /* ----- */
1780:      0x00, /* ----- */
1781:
1782:          /* 0x7f */
1783:      0x66, /* -##--##- */
1784:      0x66, /* -##--##- */
1785:      0x00, /* ----- */
1786:      0x66, /* -##--##- */
1787:      0x66, /* -##--##- */
1788:      0x66, /* -##--##- */
1789:      0x3c, /* --#####- */
1790:      0x18, /* ----##---- */
1791:      0x18, /* ----##---- */
1792:      0x3c, /* --#####- */
1793:      0x00, /* ----- */
1794:      0x00, /* ----- */
1795:
1796:          /* 0x80 */
1797:      0x00, /* ----- */
1798:      0x00, /* ----- */
1799:      0xff, /* ##### */
1800:      0x00, /* ----- */
1801:      0x00, /* ----- */
1802:      0x00, /* ----- */
1803:      0x00, /* ----- */
1804:      0x00, /* ----- */
1805:      0x00, /* ----- */
1806:      0x00, /* ----- */
1807:      0x00, /* ----- */
1808:      0x00, /* ----- */
1809:

```

drivers/video/font-lat9-8x12.c

Page 28/54

```

1810:                /* 0x81          */
1811:    0x18,          /* ----##---- */
1812:    0x18,          /* ----##---- */
1813:    0x18,          /* ----##---- */
1814:    0x18,          /* ----##---- */
1815:    0x18,          /* ----##---- */
1816:    0x18,          /* ----##---- */
1817:    0x00,          /* ----- */
1818:    0x00,          /* ----- */
1819:    0x00,          /* ----- */
1820:    0x00,          /* ----- */
1821:    0x00,          /* ----- */
1822:    0x00,          /* ----- */
1823:
1824:                /* 0x82          */
1825:    0x00,          /* ----- */
1826:    0x00,          /* ----- */
1827:    0x00,          /* ----- */
1828:    0x00,          /* ----- */
1829:    0x00,          /* ----- */
1830:    0x1f,          /* ----##### */
1831:    0x00,          /* ----- */
1832:    0x00,          /* ----- */
1833:    0x00,          /* ----- */
1834:    0x00,          /* ----- */
1835:    0x00,          /* ----- */
1836:    0x00,          /* ----- */
1837:
1838:                /* 0x83          */
1839:    0x18,          /* ----##---- */
1840:    0x18,          /* ----##---- */
1841:    0x18,          /* ----##---- */
1842:    0x18,          /* ----##---- */
1843:    0x18,          /* ----##---- */
1844:    0x1f,          /* ----##### */
1845:    0x00,          /* ----- */
1846:    0x00,          /* ----- */
1847:    0x00,          /* ----- */
1848:    0x00,          /* ----- */
1849:    0x00,          /* ----- */
1850:    0x00,          /* ----- */
1851:
1852:                /* 0x84          */
1853:    0x00,          /* ----- */
1854:    0x00,          /* ----- */
1855:    0x00,          /* ----- */
1856:    0x00,          /* ----- */
1857:    0x00,          /* ----- */
1858:    0x18,          /* ----##---- */
1859:    0x18,          /* ----##---- */
1860:    0x18,          /* ----##---- */
1861:    0x18,          /* ----##---- */
1862:    0x18,          /* ----##---- */
1863:    0x18,          /* ----##---- */
1864:    0x18,          /* ----##---- */
1865:
1866:                /* 0x85          */
1867:    0x18,          /* ----##---- */
1868:    0x18,          /* ----##---- */
1869:    0x18,          /* ----##---- */
1870:    0x18,          /* ----##---- */
1871:    0x18,          /* ----##---- */
1872:    0x18,          /* ----##---- */
1873:    0x18,          /* ----##---- */
1874:    0x18,          /* ----##---- */
1875:    0x18,          /* ----##---- */
1876:    0x18,          /* ----##---- */

```

drivers/video/font-lat9-8x12.c

Page 29/54

```

1877:      0x18, /* ---##--- */
1878:      0x18, /* ---##--- */
1879:
1880:          /* 0x86 */
1881:      0x00, /* ----- */
1882:      0x00, /* ----- */
1883:      0x00, /* ----- */
1884:      0x00, /* ----- */
1885:      0x00, /* ----- */
1886:      0x1f, /* ---##### */
1887:      0x18, /* ---##--- */
1888:      0x18, /* ---##--- */
1889:      0x18, /* ---##--- */
1890:      0x18, /* ---##--- */
1891:      0x18, /* ---##--- */
1892:      0x18, /* ---##--- */
1893:
1894:          /* 0x87 */
1895:      0x18, /* ---##--- */
1896:      0x18, /* ---##--- */
1897:      0x18, /* ---##--- */
1898:      0x18, /* ---##--- */
1899:      0x18, /* ---##--- */
1900:      0x1f, /* ---##### */
1901:      0x18, /* ---##--- */
1902:      0x18, /* ---##--- */
1903:      0x18, /* ---##--- */
1904:      0x18, /* ---##--- */
1905:      0x18, /* ---##--- */
1906:      0x18, /* ---##--- */
1907:
1908:          /* 0x88 */
1909:      0x00, /* ----- */
1910:      0x00, /* ----- */
1911:      0x00, /* ----- */
1912:      0x00, /* ----- */
1913:      0x00, /* ----- */
1914:      0xf8, /* #####--- */
1915:      0x00, /* ----- */
1916:      0x00, /* ----- */
1917:      0x00, /* ----- */
1918:      0x00, /* ----- */
1919:      0x00, /* ----- */
1920:      0x00, /* ----- */
1921:
1922:          /* 0x89 */
1923:      0x18, /* ---##--- */
1924:      0x18, /* ---##--- */
1925:      0x18, /* ---##--- */
1926:      0x18, /* ---##--- */
1927:      0x18, /* ---##--- */
1928:      0xf8, /* #####--- */
1929:      0x00, /* ----- */
1930:      0x00, /* ----- */
1931:      0x00, /* ----- */
1932:      0x00, /* ----- */
1933:      0x00, /* ----- */
1934:      0x00, /* ----- */
1935:
1936:          /* 0x8a */
1937:      0x00, /* ----- */
1938:      0x00, /* ----- */
1939:      0x00, /* ----- */
1940:      0x00, /* ----- */
1941:      0x00, /* ----- */
1942:      0xff, /* #####--- */
1943:      0x00, /* ----- */

```

drivers/video/font-lat9-8x12.c

Page 30/54

```

1944:      0x00,    /* ----- */
1945:      0x00,    /* ----- */
1946:      0x00,    /* ----- */
1947:      0x00,    /* ----- */
1948:      0x00,    /* ----- */
1949:
1950:          /* 0x8b      */
1951:      0x18,    /* ----##---- */
1952:      0x18,    /* ----##---- */
1953:      0x18,    /* ----##---- */
1954:      0x18,    /* ----##---- */
1955:      0x18,    /* ----##---- */
1956:      0xff,    /* ##### */
1957:      0x00,    /* ----- */
1958:      0x00,    /* ----- */
1959:      0x00,    /* ----- */
1960:      0x00,    /* ----- */
1961:      0x00,    /* ----- */
1962:      0x00,    /* ----- */
1963:
1964:          /* 0x8c      */
1965:      0x00,    /* ----- */
1966:      0x00,    /* ----- */
1967:      0x00,    /* ----- */
1968:      0x00,    /* ----- */
1969:      0x00,    /* ----- */
1970:      0xf8,    /* #####---- */
1971:      0x18,    /* ----##---- */
1972:      0x18,    /* ----##---- */
1973:      0x18,    /* ----##---- */
1974:      0x18,    /* ----##---- */
1975:      0x18,    /* ----##---- */
1976:      0x18,    /* ----##---- */
1977:
1978:          /* 0x8d      */
1979:      0x18,    /* ----##---- */
1980:      0x18,    /* ----##---- */
1981:      0x18,    /* ----##---- */
1982:      0x18,    /* ----##---- */
1983:      0x18,    /* ----##---- */
1984:      0xf8,    /* #####---- */
1985:      0x18,    /* ----##---- */
1986:      0x18,    /* ----##---- */
1987:      0x18,    /* ----##---- */
1988:      0x18,    /* ----##---- */
1989:      0x18,    /* ----##---- */
1990:      0x18,    /* ----##---- */
1991:
1992:          /* 0x8e      */
1993:      0x00,    /* ----- */
1994:      0x00,    /* ----- */
1995:      0x00,    /* ----- */
1996:      0x00,    /* ----- */
1997:      0x00,    /* ----- */
1998:      0xff,    /* ##### */
1999:      0x18,    /* ----##---- */
2000:      0x18,    /* ----##---- */
2001:      0x18,    /* ----##---- */
2002:      0x18,    /* ----##---- */
2003:      0x18,    /* ----##---- */
2004:      0x18,    /* ----##---- */
2005:
2006:          /* 0x8f      */
2007:      0x18,    /* ----##---- */
2008:      0x18,    /* ----##---- */
2009:      0x18,    /* ----##---- */
2010:      0x18,    /* ----##---- */

```


drivers/video/font-lat9-8x12.c

Page 31/54

```

2011:      0x18, /* ----##---- */
2012:      0xff, /* ##### */
2013:      0x18, /* ----##---- */
2014:      0x18, /* ----##---- */
2015:      0x18, /* ----##---- */
2016:      0x18, /* ----##---- */
2017:      0x18, /* ----##---- */
2018:      0x18, /* ----##---- */
2019:
2020:      /* 0x90 */
2021:      0x00, /* ----- */
2022:      0x00, /* ----- */
2023:      0x00, /* ----- */
2024:      0x00, /* ----- */
2025:      0x00, /* ----- */
2026:      0x00, /* ----- */
2027:      0x00, /* ----- */
2028:      0xff, /* ##### */
2029:      0x00, /* ----- */
2030:      0x00, /* ----- */
2031:      0x00, /* ----- */
2032:      0x00, /* ----- */
2033:
2034:      /* 0x91 */
2035:      0x6c, /* -##-##-- */
2036:      0x6c, /* -##-##-- */
2037:      0x6c, /* -##-##-- */
2038:      0x6c, /* -##-##-- */
2039:      0x6c, /* -##-##-- */
2040:      0x6c, /* -##-##-- */
2041:      0x7c, /* -#####-- */
2042:      0x00, /* ----- */
2043:      0x00, /* ----- */
2044:      0x00, /* ----- */
2045:      0x00, /* ----- */
2046:      0x00, /* ----- */
2047:
2048:      /* 0x92 */
2049:      0x00, /* ----- */
2050:      0x00, /* ----- */
2051:      0x00, /* ----- */
2052:      0x00, /* ----- */
2053:      0x3f, /* --##### */
2054:      0x30, /* --##---- */
2055:      0x3f, /* --##### */
2056:      0x00, /* ----- */
2057:      0x00, /* ----- */
2058:      0x00, /* ----- */
2059:      0x00, /* ----- */
2060:      0x00, /* ----- */
2061:
2062:      /* 0x93 */
2063:      0x6c, /* -##-##-- */
2064:      0x6c, /* -##-##-- */
2065:      0x6c, /* -##-##-- */
2066:      0x6c, /* -##-##-- */
2067:      0x6f, /* -##-#### */
2068:      0x60, /* -##----- */
2069:      0x7f, /* -##### */
2070:      0x00, /* ----- */
2071:      0x00, /* ----- */
2072:      0x00, /* ----- */
2073:      0x00, /* ----- */
2074:      0x00, /* ----- */
2075:
2076:      /* 0x94 */
2077:      0x00, /* ----- */

```

drivers/video/font-lat9-8x12.c

Page 32/54

```

2078:      0x00,    /* ----- */
2079:      0x00,    /* ----- */
2080:      0x00,    /* ----- */
2081:      0x7c,    /* ##### */
2082:      0x6c,    /* ##-##- */
2083:      0x6c,    /* ##-##- */
2084:      0x6c,    /* ##-##- */
2085:      0x6c,    /* ##-##- */
2086:      0x6c,    /* ##-##- */
2087:      0x6c,    /* ##-##- */
2088:      0x6c,    /* ##-##- */
2089:
2090:          /* 0x95 */
2091:      0x6c,    /* ##-##- */
2092:      0x6c,    /* ##-##- */
2093:      0x6c,    /* ##-##- */
2094:      0x6c,    /* ##-##- */
2095:      0x6c,    /* ##-##- */
2096:      0x6c,    /* ##-##- */
2097:      0x6c,    /* ##-##- */
2098:      0x6c,    /* ##-##- */
2099:      0x6c,    /* ##-##- */
2100:      0x6c,    /* ##-##- */
2101:      0x6c,    /* ##-##- */
2102:      0x6c,    /* ##-##- */
2103:
2104:          /* 0x96 */
2105:      0x00,    /* ----- */
2106:      0x00,    /* ----- */
2107:      0x00,    /* ----- */
2108:      0x00,    /* ----- */
2109:      0x7f,    /* ##### */
2110:      0x60,    /* ##- */
2111:      0x6f,    /* ##-#### */
2112:      0x6c,    /* ##-##- */
2113:      0x6c,    /* ##-##- */
2114:      0x6c,    /* ##-##- */
2115:      0x6c,    /* ##-##- */
2116:      0x6c,    /* ##-##- */
2117:
2118:          /* 0x97 */
2119:      0x6c,    /* ##-##- */
2120:      0x6c,    /* ##-##- */
2121:      0x6c,    /* ##-##- */
2122:      0x6c,    /* ##-##- */
2123:      0x6f,    /* ##-#### */
2124:      0x60,    /* ##- */
2125:      0x6f,    /* ##-#### */
2126:      0x6c,    /* ##-##- */
2127:      0x6c,    /* ##-##- */
2128:      0x6c,    /* ##-##- */
2129:      0x6c,    /* ##-##- */
2130:      0x6c,    /* ##-##- */
2131:
2132:          /* 0x98 */
2133:      0x00,    /* ----- */
2134:      0x00,    /* ----- */
2135:      0x00,    /* ----- */
2136:      0x00,    /* ----- */
2137:      0xf0,    /* #####- */
2138:      0x0c,    /* ----##- */
2139:      0xf0,    /* #####- */
2140:      0x00,    /* ----- */
2141:      0x00,    /* ----- */
2142:      0x00,    /* ----- */
2143:      0x00,    /* ----- */
2144:      0x00,    /* ----- */

```

drivers/video/font-lat9-8x12.c

Page 33/54

```

2145:
2146:          /* 0x99          */
2147:    0x6c, /* -##-##-- */
2148:    0x6c, /* -##-##-- */
2149:    0x6c, /* -##-##-- */
2150:    0x6c, /* -##-##-- */
2151:    0xec, /* ###-##-- */
2152:    0x0c, /* ----##-- */
2153:    0xfc, /* #####-- */
2154:    0x00, /* ----- */
2155:    0x00, /* ----- */
2156:    0x00, /* ----- */
2157:    0x00, /* ----- */
2158:    0x00, /* ----- */
2159:
2160:          /* 0x9a          */
2161:    0x00, /* ----- */
2162:    0x00, /* ----- */
2163:    0x00, /* ----- */
2164:    0x00, /* ----- */
2165:    0xff, /* ##### */
2166:    0x00, /* ----- */
2167:    0xff, /* ##### */
2168:    0x00, /* ----- */
2169:    0x00, /* ----- */
2170:    0x00, /* ----- */
2171:    0x00, /* ----- */
2172:    0x00, /* ----- */
2173:
2174:          /* 0x9b          */
2175:    0x6c, /* -##-##-- */
2176:    0x6c, /* -##-##-- */
2177:    0x6c, /* -##-##-- */
2178:    0x6c, /* -##-##-- */
2179:    0xef, /* ###-#### */
2180:    0x00, /* ----- */
2181:    0xff, /* ##### */
2182:    0x00, /* ----- */
2183:    0x00, /* ----- */
2184:    0x00, /* ----- */
2185:    0x00, /* ----- */
2186:    0x00, /* ----- */
2187:
2188:          /* 0x9c          */
2189:    0x00, /* ----- */
2190:    0x00, /* ----- */
2191:    0x00, /* ----- */
2192:    0x00, /* ----- */
2193:    0xfc, /* #####-- */
2194:    0x0c, /* ----##-- */
2195:    0xec, /* ###-##-- */
2196:    0x6c, /* -##-##-- */
2197:    0x6c, /* -##-##-- */
2198:    0x6c, /* -##-##-- */
2199:    0x6c, /* -##-##-- */
2200:    0x6c, /* -##-##-- */
2201:
2202:          /* 0x9d          */
2203:    0x6c, /* -##-##-- */
2204:    0x6c, /* -##-##-- */
2205:    0x6c, /* -##-##-- */
2206:    0x6c, /* -##-##-- */
2207:    0xec, /* ###-##-- */
2208:    0x0c, /* ----##-- */
2209:    0xec, /* ###-##-- */
2210:    0x6c, /* -##-##-- */
2211:    0x6c, /* -##-##-- */

```

drivers/video/font-lat9-8x12.c

Page 34/54

```

2212:      0x6c, /* -##-##-- */
2213:      0x6c, /* -##-##-- */
2214:      0x6c, /* -##-##-- */
2215:
2216:          /* 0x9e */
2217:      0x00, /* ----- */
2218:      0x00, /* ----- */
2219:      0x00, /* ----- */
2220:      0x00, /* ----- */
2221:      0xff, /* ##### */
2222:      0x00, /* ----- */
2223:      0xef, /* ##-##- */
2224:      0x6c, /* -##-##-- */
2225:      0x6c, /* -##-##-- */
2226:      0x6c, /* -##-##-- */
2227:      0x6c, /* -##-##-- */
2228:      0x6c, /* -##-##-- */
2229:
2230:          /* 0x9f */
2231:      0x6c, /* -##-##-- */
2232:      0x6c, /* -##-##-- */
2233:      0x6c, /* -##-##-- */
2234:      0x6c, /* -##-##-- */
2235:      0xef, /* ##-##- */
2236:      0x00, /* ----- */
2237:      0xef, /* ##-##- */
2238:      0x6c, /* -##-##-- */
2239:      0x6c, /* -##-##-- */
2240:      0x6c, /* -##-##-- */
2241:      0x6c, /* -##-##-- */
2242:      0x6c, /* -##-##-- */
2243:
2244:          /* 0xa0 */
2245:      0x00, /* ----- */
2246:      0x00, /* ----- */
2247:      0x00, /* ----- */
2248:      0x00, /* ----- */
2249:      0x00, /* ----- */
2250:      0x00, /* ----- */
2251:      0x00, /* ----- */
2252:      0x00, /* ----- */
2253:      0x00, /* ----- */
2254:      0x82, /* #-#-#-#- */
2255:      0xfe, /* #####- */
2256:      0x00, /* ----- */
2257:
2258:          /* 0xa1 */
2259:      0x00, /* ----- */
2260:      0x00, /* ----- */
2261:      0x00, /* ----- */
2262:      0x18, /* ---##-#- */
2263:      0x18, /* ---##-#- */
2264:      0x00, /* ----- */
2265:      0x18, /* ---##-#- */
2266:      0x18, /* ---##-#- */
2267:      0x3c, /* --####-# */
2268:      0x3c, /* --####-# */
2269:      0x3c, /* --####-# */
2270:      0x18, /* ---##-#- */
2271:
2272:          /* 0xa2 */
2273:      0x00, /* ----- */
2274:      0x00, /* ----- */
2275:      0x10, /* ---#----- */
2276:      0x7c, /* -#####-# */
2277:      0xd6, /* ##-##-##- */
2278:      0xd0, /* ##-##-#-#- */

```

drivers/video/font-lat9-8x12.c

Page 35/54

```

2279:      0xd0,    /* ##-#---- */
2280:      0xd6,    /* ##-##-##- */
2281:      0x7c,    /* -#####-- */
2282:      0x10,    /* ----#---- */
2283:      0x00,    /* ----- */
2284:      0x00,    /* ----- */
2285:
2286:          /* 0xa3 */
2287:      0x00,    /* ----- */
2288:      0x38,    /* --###---- */
2289:      0x6c,    /* -##-##-  */
2290:      0x60,    /* -##----- */
2291:      0x60,    /* -##----- */
2292:      0xf0,    /* #####---- */
2293:      0x60,    /* -##----- */
2294:      0x66,    /* -##-##-  */
2295:      0xf6,    /* #####-##- */
2296:      0x6c,    /* -##-##-  */
2297:      0x00,    /* ----- */
2298:      0x00,    /* ----- */
2299:
2300:          /* 0xa4 */
2301:      0x00,    /* ----- */
2302:      0x3c,    /* --#####-- */
2303:      0x62,    /* -##-##-  */
2304:      0x60,    /* -##----- */
2305:      0xf8,    /* #####---- */
2306:      0x60,    /* -##----- */
2307:      0xf8,    /* #####---- */
2308:      0x60,    /* -##----- */
2309:      0x62,    /* -##-##-  */
2310:      0x3c,    /* --#####-- */
2311:      0x00,    /* ----- */
2312:      0x00,    /* ----- */
2313:
2314:          /* 0xa5 */
2315:      0x00,    /* ----- */
2316:      0x66,    /* -##-##-  */
2317:      0x66,    /* -##-##-  */
2318:      0x3c,    /* --#####-- */
2319:      0x18,    /* ---##-   */
2320:      0x7e,    /* -#####-  */
2321:      0x18,    /* ----##-  */
2322:      0x3c,    /* --#####-- */
2323:      0x18,    /* ----##-  */
2324:      0x18,    /* ----##-  */
2325:      0x00,    /* ----- */
2326:      0x00,    /* ----- */
2327:
2328:          /* 0xa6 */
2329:      0x6c,    /* -##-##-  */
2330:      0x38,    /* --###---- */
2331:      0x00,    /* ----- */
2332:      0x7c,    /* -#####-- */
2333:      0xc6,    /* ##-##-  */
2334:      0xc0,    /* ##-##-  */
2335:      0x7c,    /* -#####-- */
2336:      0x06,    /* -----##- */
2337:      0xc6,    /* ##-##-  */
2338:      0x7c,    /* -#####-- */
2339:      0x00,    /* ----- */
2340:      0x00,    /* ----- */
2341:
2342:          /* 0xa7 */
2343:      0x7c,    /* -#####-- */
2344:      0xc6,    /* ##-##-  */
2345:      0xc6,    /* ##-##-  */

```

drivers/video/font-lat9-8x12.c

Page 36/54

```

2346:      0x60,    /* -##----- */
2347:      0x7c,    /* -#####-- */
2348:      0xc6,    /* ##---##- */
2349:      0xc6,    /* ##---##- */
2350:      0x7c,    /* -#####-- */
2351:      0x0c,    /* ----##-- */
2352:      0xc6,    /* ##---##- */
2353:      0xc6,    /* ##---##- */
2354:      0x7c,    /* -#####-- */
2355:
2356:          /* 0xa8          */
2357:      0x00,    /* ----- */
2358:      0x6c,    /* -##-##-- */
2359:      0x38,    /* --###--- */
2360:      0x00,    /* ----- */
2361:      0x7c,    /* -#####-- */
2362:      0xc6,    /* ##---##- */
2363:      0x70,    /* -###----- */
2364:      0x1c,    /* ---###-- */
2365:      0xc6,    /* ##---##- */
2366:      0x7c,    /* -#####-- */
2367:      0x00,    /* ----- */
2368:      0x00,    /* ----- */
2369:
2370:          /* 0xa9          */
2371:      0x7e,    /* -#####- */
2372:      0x81,    /* #-----# */
2373:      0x99,    /* #-##---# */
2374:      0xa5,    /* #-#---#-# */
2375:      0xa1,    /* #-#-----# */
2376:      0xa1,    /* #-#-----# */
2377:      0xa5,    /* #-#---#-# */
2378:      0x99,    /* #-##---# */
2379:      0x81,    /* #-----# */
2380:      0x7e,    /* -#####- */
2381:      0x00,    /* ----- */
2382:      0x00,    /* ----- */
2383:
2384:          /* 0xaa          */
2385:      0x3c,    /* --####-- */
2386:      0x6c,    /* -##-##-- */
2387:      0x6c,    /* -##-##-- */
2388:      0x3e,    /* --#####- */
2389:      0x00,    /* ----- */
2390:      0x7e,    /* -#####- */
2391:      0x00,    /* ----- */
2392:      0x00,    /* ----- */
2393:      0x00,    /* ----- */
2394:      0x00,    /* ----- */
2395:      0x00,    /* ----- */
2396:      0x00,    /* ----- */
2397:
2398:          /* 0xab          */
2399:      0x00,    /* ----- */
2400:      0x00,    /* ----- */
2401:      0x00,    /* ----- */
2402:      0x36,    /* --##-##- */
2403:      0x6c,    /* -##-##-- */
2404:      0xd8,    /* ##-##--- */
2405:      0x6c,    /* -##-##-- */
2406:      0x36,    /* --##-##- */
2407:      0x00,    /* ----- */
2408:      0x00,    /* ----- */
2409:      0x00,    /* ----- */
2410:      0x00,    /* ----- */
2411:
2412:          /* 0xac          */

```

drivers/video/font-lat9-8x12.c

Page 37/54

```

2413:      0x00, /* ----- */
2414:      0x00, /* ----- */
2415:      0x00, /* ----- */
2416:      0x00, /* ----- */
2417:      0x00, /* ----- */
2418:      0x7e, /* -#####- */
2419:      0x06, /* -----##- */
2420:      0x06, /* -----##- */
2421:      0x06, /* -----##- */
2422:      0x00, /* ----- */
2423:      0x00, /* ----- */
2424:      0x00, /* ----- */
2425:
2426:          /* 0xad */
2427:      0x00, /* ----- */
2428:      0x00, /* ----- */
2429:      0x00, /* ----- */
2430:      0x00, /* ----- */
2431:      0x00, /* ----- */
2432:      0x7e, /* -#####- */
2433:      0x00, /* ----- */
2434:      0x00, /* ----- */
2435:      0x00, /* ----- */
2436:      0x00, /* ----- */
2437:      0x00, /* ----- */
2438:      0x00, /* ----- */
2439:
2440:          /* 0xae */
2441:      0x7e, /* -#####- */
2442:      0x81, /* #-----# */
2443:      0xb9, /* #-###--# */
2444:      0xa5, /* #-#--#-# */
2445:      0xa5, /* #-#--#-# */
2446:      0xb9, /* #-###--# */
2447:      0xa5, /* #-#--#-# */
2448:      0xa5, /* #-#--#-# */
2449:      0x81, /* #-----# */
2450:      0x7e, /* -#####- */
2451:      0x00, /* ----- */
2452:      0x00, /* ----- */
2453:
2454:          /* 0xaf */
2455:      0xff, /* ##### */
2456:      0x00, /* ----- */
2457:      0x00, /* ----- */
2458:      0x00, /* ----- */
2459:      0x00, /* ----- */
2460:      0x00, /* ----- */
2461:      0x00, /* ----- */
2462:      0x00, /* ----- */
2463:      0x00, /* ----- */
2464:      0x00, /* ----- */
2465:      0x00, /* ----- */
2466:      0x00, /* ----- */
2467:
2468:          /* 0xb0 */
2469:      0x00, /* ----- */
2470:      0x38, /* --###--- */
2471:      0x6c, /* -##-##-- */
2472:      0x38, /* --###--- */
2473:      0x00, /* ----- */
2474:      0x00, /* ----- */
2475:      0x00, /* ----- */
2476:      0x00, /* ----- */
2477:      0x00, /* ----- */
2478:      0x00, /* ----- */
2479:      0x00, /* ----- */

```

drivers/video/font-lat9-8x12.c

Page 38/54

```

2480:      0x00,    /* ----- */
2481:
2482:          /* 0xb1 */
2483:      0x00,    /* ----- */
2484:      0x00,    /* ----- */
2485:      0x00,    /* ----- */
2486:      0x18,    /* ---##--- */
2487:      0x18,    /* ---##--- */
2488:      0x7e,    /* -#####- */
2489:      0x18,    /* ---##--- */
2490:      0x18,    /* ---##--- */
2491:      0x00,    /* ----- */
2492:      0x7e,    /* -#####- */
2493:      0x00,    /* ----- */
2494:      0x00,    /* ----- */
2495:
2496:          /* 0xb2 */
2497:      0x00,    /* ----- */
2498:      0x38,    /* --###--- */
2499:      0x6c,    /* -##-##-- */
2500:      0x18,    /* ---##--- */
2501:      0x30,    /* --##---- */
2502:      0x7c,    /* -#####- */
2503:      0x00,    /* ----- */
2504:      0x00,    /* ----- */
2505:      0x00,    /* ----- */
2506:      0x00,    /* ----- */
2507:      0x00,    /* ----- */
2508:      0x00,    /* ----- */
2509:
2510:          /* 0xb3 */
2511:      0x00,    /* ----- */
2512:      0x38,    /* --###--- */
2513:      0x6c,    /* -##-##-- */
2514:      0x18,    /* ---##--- */
2515:      0x6c,    /* -##-##-- */
2516:      0x38,    /* --###--- */
2517:      0x00,    /* ----- */
2518:      0x00,    /* ----- */
2519:      0x00,    /* ----- */
2520:      0x00,    /* ----- */
2521:      0x00,    /* ----- */
2522:      0x00,    /* ----- */
2523:
2524:          /* 0xb4 */
2525:      0x6c,    /* -##-##-- */
2526:      0x38,    /* --###--- */
2527:      0x00,    /* ----- */
2528:      0xfe,    /* #####-#- */
2529:      0xc6,    /* ##---##- */
2530:      0x8c,    /* #---##-- */
2531:      0x38,    /* --###--- */
2532:      0x62,    /* -##-##-- */
2533:      0xc6,    /* ##---##- */
2534:      0xfe,    /* #####-#- */
2535:      0x00,    /* ----- */
2536:      0x00,    /* ----- */
2537:
2538:          /* 0xb5 */
2539:      0x00,    /* ----- */
2540:      0x00,    /* ----- */
2541:      0x00,    /* ----- */
2542:      0x00,    /* ----- */
2543:      0xcc,    /* ##---##- */
2544:      0xcc,    /* ##---##- */
2545:      0xcc,    /* ##---##- */
2546:      0xcc,    /* ##---##- */

```


drivers/video/font-lat9-8x12.c

Page 39/54

```

2547:      0xcc,      /* ##--##-- */
2548:      0xf6,      /* #####-## */
2549:      0xc0,      /* ##----- */
2550:      0xc0,      /* ##----- */
2551:
2552:          /* 0xb6          */
2553:      0x00,      /* ----- */
2554:      0x7f,      /* -##### */
2555:      0xdb,      /* #-##-## */
2556:      0xdb,      /* #-##-## */
2557:      0xdb,      /* #-##-## */
2558:      0x7b,      /* -###-## */
2559:      0x1b,      /* ---##-## */
2560:      0x1b,      /* ---##-## */
2561:      0x1b,      /* ---##-## */
2562:      0x1b,      /* ---##-## */
2563:      0x00,      /* ----- */
2564:      0x00,      /* ----- */
2565:
2566:          /* 0xb7          */
2567:      0x00,      /* ----- */
2568:      0x00,      /* ----- */
2569:      0x00,      /* ----- */
2570:      0x00,      /* ----- */
2571:      0x00,      /* ----- */
2572:      0x18,      /* ---##--- */
2573:      0x18,      /* ---##--- */
2574:      0x00,      /* ----- */
2575:      0x00,      /* ----- */
2576:      0x00,      /* ----- */
2577:      0x00,      /* ----- */
2578:      0x00,      /* ----- */
2579:
2580:          /* 0xb8          */
2581:      0x00,      /* ----- */
2582:      0x6c,      /* -##-##- */
2583:      0x38,      /* --##---- */
2584:      0x00,      /* ----- */
2585:      0xfe,      /* #####- */
2586:      0x8c,      /* #---##- */
2587:      0x18,      /* ---##--- */
2588:      0x30,      /* --##---- */
2589:      0x62,      /* -##-##- */
2590:      0xfe,      /* #####- */
2591:      0x00,      /* ----- */
2592:      0x00,      /* ----- */
2593:
2594:          /* 0xb9          */
2595:      0x00,      /* ----- */
2596:      0x30,      /* --##---- */
2597:      0x70,      /* -##-##- */
2598:      0x30,      /* --##---- */
2599:      0x30,      /* --##---- */
2600:      0x78,      /* -###---- */
2601:      0x00,      /* ----- */
2602:      0x00,      /* ----- */
2603:      0x00,      /* ----- */
2604:      0x00,      /* ----- */
2605:      0x00,      /* ----- */
2606:      0x00,      /* ----- */
2607:
2608:          /* 0xba          */
2609:      0x38,      /* --##---- */
2610:      0x6c,      /* -##-##- */
2611:      0x6c,      /* -##-##- */
2612:      0x38,      /* --##---- */
2613:      0x00,      /* ----- */

```

drivers/video/font-lat9-8x12.c

Page 40/54

```

2614:      0x7c,    /* #####-- */
2615:      0x00,    /* ----- */
2616:      0x00,    /* ----- */
2617:      0x00,    /* ----- */
2618:      0x00,    /* ----- */
2619:      0x00,    /* ----- */
2620:      0x00,    /* ----- */
2621:
2622:          /* 0xbb */
2623:      0x00,    /* ----- */
2624:      0x00,    /* ----- */
2625:      0x00,    /* ----- */
2626:      0xd8,    /* ##-##- */
2627:      0x6c,    /* -##-##- */
2628:      0x36,    /* --##-##- */
2629:      0x6c,    /* -##-##- */
2630:      0xd8,    /* ##-##- */
2631:      0x00,    /* ----- */
2632:      0x00,    /* ----- */
2633:      0x00,    /* ----- */
2634:      0x00,    /* ----- */
2635:
2636:          /* 0xbc */
2637:      0x00,    /* ----- */
2638:      0x6e,    /* -##-##- */
2639:      0xdf,    /* ##-##### */
2640:      0xd8,    /* ##-##- */
2641:      0xde,    /* ##-##### */
2642:      0xde,    /* ##-##### */
2643:      0xd8,    /* ##-##- */
2644:      0xd8,    /* ##-##- */
2645:      0xdf,    /* ##-##### */
2646:      0x6e,    /* -##-##- */
2647:      0x00,    /* ----- */
2648:      0x00,    /* ----- */
2649:
2650:          /* 0xbd */
2651:      0x00,    /* ----- */
2652:      0x00,    /* ----- */
2653:      0x00,    /* ----- */
2654:      0x00,    /* ----- */
2655:      0x6c,    /* -##-##- */
2656:      0xda,    /* ##-##-##- */
2657:      0xde,    /* ##-##### */
2658:      0xd8,    /* ##-##- */
2659:      0xda,    /* ##-##-##- */
2660:      0x6c,    /* -##-##- */
2661:      0x00,    /* ----- */
2662:      0x00,    /* ----- */
2663:
2664:          /* 0xbe */
2665:      0x66,    /* -##-##- */
2666:      0x66,    /* -##-##- */
2667:      0x00,    /* ----- */
2668:      0x66,    /* -##-##- */
2669:      0x66,    /* -##-##- */
2670:      0x3c,    /* --#####- */
2671:      0x18,    /* ---##- */
2672:      0x18,    /* ---##- */
2673:      0x18,    /* ---##- */
2674:      0x3c,    /* --#####- */
2675:      0x00,    /* ----- */
2676:      0x00,    /* ----- */
2677:
2678:          /* 0xbf */
2679:      0x00,    /* ----- */
2680:      0x00,    /* ----- */

```

drivers/video/font-lat9-8x12.c

Page 41/54

```

2681:      0x00, /* ----- */
2682:      0x30, /* --##---- */
2683:      0x30, /* --##---- */
2684:      0x00, /* ----- */
2685:      0x30, /* --##---- */
2686:      0x30, /* --##---- */
2687:      0x60, /* -##----- */
2688:      0xc6, /* ##---##- */
2689:      0xc6, /* ##---##- */
2690:      0x7c, /* -#####- */
2691:
2692:          /* 0xc0 */
2693:      0x30, /* --##---- */
2694:      0x18, /* ----##---- */
2695:      0x00, /* ----- */
2696:      0x38, /* --###---- */
2697:      0x6c, /* -##-##- */
2698:      0xc6, /* ##---##- */
2699:      0xc6, /* ##---##- */
2700:      0xfe, /* #####- */
2701:      0xc6, /* ##---##- */
2702:      0xc6, /* ##---##- */
2703:      0x00, /* ----- */
2704:      0x00, /* ----- */
2705:
2706:          /* 0xc1 */
2707:      0x18, /* ----##---- */
2708:      0x30, /* --##---- */
2709:      0x00, /* ----- */
2710:      0x38, /* --###---- */
2711:      0x6c, /* -##-##- */
2712:      0xc6, /* ##---##- */
2713:      0xc6, /* ##---##- */
2714:      0xfe, /* #####- */
2715:      0xc6, /* ##---##- */
2716:      0xc6, /* ##---##- */
2717:      0x00, /* ----- */
2718:      0x00, /* ----- */
2719:
2720:          /* 0xc2 */
2721:      0x38, /* --###---- */
2722:      0x6c, /* -##-##- */
2723:      0x38, /* --###---- */
2724:      0x00, /* ----- */
2725:      0x7c, /* -#####- */
2726:      0xc6, /* ##---##- */
2727:      0xc6, /* ##---##- */
2728:      0xfe, /* #####- */
2729:      0xc6, /* ##---##- */
2730:      0xc6, /* ##---##- */
2731:      0x00, /* ----- */
2732:      0x00, /* ----- */
2733:
2734:          /* 0xc3 */
2735:      0x76, /* -###-##- */
2736:      0xdc, /* ##-###- */
2737:      0x00, /* ----- */
2738:      0x38, /* --###---- */
2739:      0x6c, /* -##-##- */
2740:      0xc6, /* ##---##- */
2741:      0xc6, /* ##---##- */
2742:      0xfe, /* #####- */
2743:      0xc6, /* ##---##- */
2744:      0xc6, /* ##---##- */
2745:      0x00, /* ----- */
2746:      0x00, /* ----- */
2747:

```

drivers/video/font-lat9-8x12.c

Page 42/54

```

2748:                /* 0xc4          */
2749:                0x6c, /* --##-##- */
2750:                0x6c, /* --##-##- */
2751:                0x00, /* ----- */
2752:                0x38, /* --###- */
2753:                0x6c, /* --##-##- */
2754:                0xc6, /* ##-##- */
2755:                0xc6, /* ##-##- */
2756:                0xfe, /* #####- */
2757:                0xc6, /* ##-##- */
2758:                0xc6, /* ##-##- */
2759:                0x00, /* ----- */
2760:                0x00, /* ----- */
2761:
2762:                /* 0xc5          */
2763:                0x38, /* --###- */
2764:                0x6c, /* --##-##- */
2765:                0x38, /* --###- */
2766:                0x00, /* ----- */
2767:                0x7c, /* -#####- */
2768:                0xc6, /* ##-##- */
2769:                0xc6, /* ##-##- */
2770:                0xfe, /* #####- */
2771:                0xc6, /* ##-##- */
2772:                0xc6, /* ##-##- */
2773:                0x00, /* ----- */
2774:                0x00, /* ----- */
2775:
2776:                /* 0xc6          */
2777:                0x7e, /* -#####- */
2778:                0xd8, /* ##-##- */
2779:                0xd8, /* ##-##- */
2780:                0xd8, /* ##-##- */
2781:                0xd8, /* ##-##- */
2782:                0xfe, /* #####- */
2783:                0xd8, /* ##-##- */
2784:                0xd8, /* ##-##- */
2785:                0xd8, /* ##-##- */
2786:                0xde, /* ##-####- */
2787:                0x00, /* ----- */
2788:                0x00, /* ----- */
2789:
2790:                /* 0xc7          */
2791:                0x00, /* ----- */
2792:                0x3c, /* --####- */
2793:                0x66, /* --##-##- */
2794:                0xc0, /* ##-##### */
2795:                0xc0, /* ##-##### */
2796:                0xc0, /* ##-##### */
2797:                0xc6, /* ##-##- */
2798:                0x66, /* --##-##- */
2799:                0x3c, /* --####- */
2800:                0x18, /* ---##- */
2801:                0xcc, /* ##-##- */
2802:                0x38, /* --###- */
2803:
2804:                /* 0xc8          */
2805:                0x18, /* ---##- */
2806:                0x0c, /* ----##- */
2807:                0x00, /* ----- */
2808:                0xfe, /* #####- */
2809:                0x66, /* --##-##- */
2810:                0x60, /* -##-##### */
2811:                0x7c, /* -#####- */
2812:                0x60, /* -##-##### */
2813:                0x66, /* --##-##- */
2814:                0xfe, /* #####- */

```

drivers/video/font-lat9-8x12.c

Page 43/54

```

2815:      0x00, /* ----- */
2816:      0x00, /* ----- */
2817:
2818:          /* 0xc9 */
2819:      0x18, /* ---##--- */
2820:      0x30, /* --##---- */
2821:      0x00, /* ----- */
2822:      0xfe, /* #####- */
2823:      0x66, /* -##-##- */
2824:      0x60, /* -##----- */
2825:      0x7c, /* -#####- */
2826:      0x60, /* -##----- */
2827:      0x66, /* -##-##- */
2828:      0xfe, /* #####- */
2829:      0x00, /* ----- */
2830:      0x00, /* ----- */
2831:
2832:          /* 0xca */
2833:      0x38, /* --###--- */
2834:      0x6c, /* -##-##- */
2835:      0x00, /* ----- */
2836:      0xfe, /* #####- */
2837:      0x66, /* -##-##- */
2838:      0x60, /* -##----- */
2839:      0x7c, /* -#####- */
2840:      0x60, /* -##----- */
2841:      0x66, /* -##-##- */
2842:      0xfe, /* #####- */
2843:      0x00, /* ----- */
2844:      0x00, /* ----- */
2845:
2846:          /* 0xcb */
2847:      0x6c, /* -##-##- */
2848:      0x6c, /* -##-##- */
2849:      0x00, /* ----- */
2850:      0xfe, /* #####- */
2851:      0x66, /* -##-##- */
2852:      0x60, /* -##----- */
2853:      0x7c, /* -#####- */
2854:      0x60, /* -##----- */
2855:      0x66, /* -##-##- */
2856:      0xfe, /* #####- */
2857:      0x00, /* ----- */
2858:      0x00, /* ----- */
2859:
2860:          /* 0xcc */
2861:      0x18, /* ---##--- */
2862:      0x0c, /* ----##-- */
2863:      0x00, /* ----- */
2864:      0x3c, /* --####-- */
2865:      0x18, /* ---##--- */
2866:      0x18, /* ---##--- */
2867:      0x18, /* ---##--- */
2868:      0x18, /* ---##--- */
2869:      0x18, /* ---##--- */
2870:      0x3c, /* --####-- */
2871:      0x00, /* ----- */
2872:      0x00, /* ----- */
2873:
2874:          /* 0xcd */
2875:      0x18, /* ---##--- */
2876:      0x30, /* --##---- */
2877:      0x00, /* ----- */
2878:      0x3c, /* --####-- */
2879:      0x18, /* ---##--- */
2880:      0x18, /* ---##--- */
2881:      0x18, /* ---##--- */

```

drivers/video/font-lat9-8x12.c

Page 44/54

```

2882:      0x18,    /* ----##---- */
2883:      0x18,    /* ----##---- */
2884:      0x3c,    /* --####-- */
2885:      0x00,    /* ----- */
2886:      0x00,    /* ----- */
2887:
2888:          /* 0xce */
2889:      0x3c,    /* --####-- */
2890:      0x66,    /* -##--##- */
2891:      0x00,    /* ----- */
2892:      0x3c,    /* --####-- */
2893:      0x18,    /* ----##---- */
2894:      0x18,    /* ----##---- */
2895:      0x18,    /* ----##---- */
2896:      0x18,    /* ----##---- */
2897:      0x18,    /* ----##---- */
2898:      0x3c,    /* --####-- */
2899:      0x00,    /* ----- */
2900:      0x00,    /* ----- */
2901:
2902:          /* 0xcf */
2903:      0x66,    /* -##--##- */
2904:      0x66,    /* -##--##- */
2905:      0x00,    /* ----- */
2906:      0x3c,    /* --####-- */
2907:      0x18,    /* ----##---- */
2908:      0x18,    /* ----##---- */
2909:      0x18,    /* ----##---- */
2910:      0x18,    /* ----##---- */
2911:      0x18,    /* ----##---- */
2912:      0x3c,    /* --####-- */
2913:      0x00,    /* ----- */
2914:      0x00,    /* ----- */
2915:
2916:          /* 0xd0 */
2917:      0x00,    /* ----- */
2918:      0xf8,    /* #####--- */
2919:      0x6c,    /* -##-##- */
2920:      0x66,    /* -##-##- */
2921:      0x66,    /* -##-##- */
2922:      0xf6,    /* #####-##- */
2923:      0x66,    /* -##-##- */
2924:      0x66,    /* -##-##- */
2925:      0x6c,    /* -##-##- */
2926:      0xf8,    /* #####--- */
2927:      0x00,    /* ----- */
2928:      0x00,    /* ----- */
2929:
2930:          /* 0xd1 */
2931:      0x76,    /* -##-##- */
2932:      0xdc,    /* ##-##- */
2933:      0x00,    /* ----- */
2934:      0xc6,    /* ##---##- */
2935:      0xe6,    /* ###-##- */
2936:      0xf6,    /* #####-##- */
2937:      0xde,    /* ##-##### */
2938:      0xce,    /* ##-##-##- */
2939:      0xc6,    /* ##---##- */
2940:      0xc6,    /* ##---##- */
2941:      0x00,    /* ----- */
2942:      0x00,    /* ----- */
2943:
2944:          /* 0xd2 */
2945:      0x30,    /* --##----- */
2946:      0x18,    /* ----##---- */
2947:      0x00,    /* ----- */
2948:      0x7c,    /* -#####-- */

```

drivers/video/font-lat9-8x12.c

Page 45/54

```

2949:      0xc6, /* ##---##- */
2950:      0xc6, /* ##---##- */
2951:      0xc6, /* ##---##- */
2952:      0xc6, /* ##---##- */
2953:      0xc6, /* ##---##- */
2954:      0x7c, /* -#####- */
2955:      0x00, /* ----- */
2956:      0x00, /* ----- */
2957:
2958:          /* 0xd3 */
2959:      0x18, /* ---##--- */
2960:      0x30, /* --##---- */
2961:      0x00, /* ----- */
2962:      0x7c, /* -#####- */
2963:      0xc6, /* ##---##- */
2964:      0xc6, /* ##---##- */
2965:      0xc6, /* ##---##- */
2966:      0xc6, /* ##---##- */
2967:      0xc6, /* ##---##- */
2968:      0x7c, /* -#####- */
2969:      0x00, /* ----- */
2970:      0x00, /* ----- */
2971:
2972:          /* 0xd4 */
2973:      0x38, /* --###--- */
2974:      0x6c, /* -##-##-- */
2975:      0x00, /* ----- */
2976:      0x7c, /* -#####- */
2977:      0xc6, /* ##---##- */
2978:      0xc6, /* ##---##- */
2979:      0xc6, /* ##---##- */
2980:      0xc6, /* ##---##- */
2981:      0xc6, /* ##---##- */
2982:      0x7c, /* -#####- */
2983:      0x00, /* ----- */
2984:      0x00, /* ----- */
2985:
2986:          /* 0xd5 */
2987:      0x76, /* -##-##- */
2988:      0xdc, /* #-###-- */
2989:      0x00, /* ----- */
2990:      0x7c, /* -#####- */
2991:      0xc6, /* ##---##- */
2992:      0xc6, /* ##---##- */
2993:      0xc6, /* ##---##- */
2994:      0xc6, /* ##---##- */
2995:      0xc6, /* ##---##- */
2996:      0x7c, /* -#####- */
2997:      0x00, /* ----- */
2998:      0x00, /* ----- */
2999:
3000:          /* 0xd6 */
3001:      0x6c, /* -##-##-- */
3002:      0x6c, /* -##-##-- */
3003:      0x00, /* ----- */
3004:      0x7c, /* -#####- */
3005:      0xc6, /* ##---##- */
3006:      0xc6, /* ##---##- */
3007:      0xc6, /* ##---##- */
3008:      0xc6, /* ##---##- */
3009:      0xc6, /* ##---##- */
3010:      0x7c, /* -#####- */
3011:      0x00, /* ----- */
3012:      0x00, /* ----- */
3013:
3014:          /* 0xd7 */
3015:      0x00, /* ----- */

```

drivers/video/font-lat9-8x12.c

Page 46/54

```

3016:      0x00, /* ----- */
3017:      0x00, /* ----- */
3018:      0x00, /* ----- */
3019:      0x6c, /* -##-##- */
3020:      0x38, /* ---###--- */
3021:      0x38, /* ---###--- */
3022:      0x6c, /* -##-##- */
3023:      0x00, /* ----- */
3024:      0x00, /* ----- */
3025:      0x00, /* ----- */
3026:      0x00, /* ----- */
3027:
3028:          /* 0xd8 */
3029:      0x00, /* ----- */
3030:      0x7e, /* -#####- */
3031:      0xc6, /* ##---##- */
3032:      0xce, /* ##---##- */
3033:      0xde, /* ##-####- */
3034:      0xd6, /* ##-##-##- */
3035:      0xf6, /* #####-##- */
3036:      0xe6, /* ###---##- */
3037:      0xc6, /* ##---##- */
3038:      0xfc, /* #####--- */
3039:      0x00, /* ----- */
3040:      0x00, /* ----- */
3041:
3042:          /* 0xd9 */
3043:      0x30, /* --##----- */
3044:      0x18, /* ---##----- */
3045:      0x00, /* ----- */
3046:      0xc6, /* ##---##- */
3047:      0xc6, /* ##---##- */
3048:      0xc6, /* ##---##- */
3049:      0xc6, /* ##---##- */
3050:      0xc6, /* ##---##- */
3051:      0xc6, /* ##---##- */
3052:      0x7c, /* -#####- */
3053:      0x00, /* ----- */
3054:      0x00, /* ----- */
3055:
3056:          /* 0xda */
3057:      0x18, /* ---##----- */
3058:      0x30, /* --##----- */
3059:      0x00, /* ----- */
3060:      0xc6, /* ##---##- */
3061:      0xc6, /* ##---##- */
3062:      0xc6, /* ##---##- */
3063:      0xc6, /* ##---##- */
3064:      0xc6, /* ##---##- */
3065:      0xc6, /* ##---##- */
3066:      0x7c, /* -#####- */
3067:      0x00, /* ----- */
3068:      0x00, /* ----- */
3069:
3070:          /* 0xdb */
3071:      0x38, /* ---###--- */
3072:      0x6c, /* -##-##- */
3073:      0x00, /* ----- */
3074:      0xc6, /* ##---##- */
3075:      0xc6, /* ##---##- */
3076:      0xc6, /* ##---##- */
3077:      0xc6, /* ##---##- */
3078:      0xc6, /* ##---##- */
3079:      0xc6, /* ##---##- */
3080:      0x7c, /* -#####- */
3081:      0x00, /* ----- */
3082:      0x00, /* ----- */

```


drivers/video/font-lat9-8x12.c

Page 47/54

```

3083:
3084:          /* 0xdc          */
3085:    0x6c,  /* ---##--- */
3086:    0x6c,  /* ---##--- */
3087:    0x00,  /* ----- */
3088:    0xc6,  /* ##---##- */
3089:    0xc6,  /* ##---##- */
3090:    0xc6,  /* ##---##- */
3091:    0xc6,  /* ##---##- */
3092:    0xc6,  /* ##---##- */
3093:    0xc6,  /* ##---##- */
3094:    0x7c,  /* -#####- */
3095:    0x00,  /* ----- */
3096:    0x00,  /* ----- */
3097:
3098:          /* 0xdd          */
3099:    0x0c,  /* ----##--- */
3100:    0x18,  /* ---##--- */
3101:    0x00,  /* ----- */
3102:    0x66,  /* ---##--- */
3103:    0x66,  /* ---##--- */
3104:    0x66,  /* ---##--- */
3105:    0x3c,  /* --#####- */
3106:    0x18,  /* ---##--- */
3107:    0x18,  /* ---##--- */
3108:    0x3c,  /* --#####- */
3109:    0x00,  /* ----- */
3110:    0x00,  /* ----- */
3111:
3112:          /* 0xde          */
3113:    0x00,  /* ----- */
3114:    0xf0,  /* #####---- */
3115:    0x60,  /* -##----- */
3116:    0x7c,  /* -#####- */
3117:    0x66,  /* -##---##- */
3118:    0x66,  /* -##---##- */
3119:    0x66,  /* -##---##- */
3120:    0x7c,  /* -#####- */
3121:    0x60,  /* -##----- */
3122:    0xf0,  /* #####---- */
3123:    0x00,  /* ----- */
3124:    0x00,  /* ----- */
3125:
3126:          /* 0xdf          */
3127:    0x00,  /* ----- */
3128:    0x7c,  /* -#####- */
3129:    0xc6,  /* ##---##- */
3130:    0xc6,  /* ##---##- */
3131:    0xc6,  /* ##---##- */
3132:    0xcc,  /* ##---##- */
3133:    0xc6,  /* ##---##- */
3134:    0xc6,  /* ##---##- */
3135:    0xd6,  /* ##-##-##- */
3136:    0xdc,  /* ##-###- */
3137:    0x80,  /* #----- */
3138:    0x00,  /* ----- */
3139:
3140:          /* 0xe0          */
3141:    0x60,  /* -##----- */
3142:    0x30,  /* --##----- */
3143:    0x18,  /* ----##--- */
3144:    0x00,  /* ----- */
3145:    0x78,  /* -#####- */
3146:    0x0c,  /* ----##--- */
3147:    0x7c,  /* -#####- */
3148:    0xcc,  /* ##---##- */
3149:    0xdc,  /* ##-###- */

```

drivers/video/font-lat9-8x12.c

Page 48/54

```

3150:      0x76,    /* -###-##- */
3151:      0x00,    /* ----- */
3152:      0x00,    /* ----- */
3153:
3154:          /* 0xe1      */
3155:      0x18,    /* ---##---- */
3156:      0x30,    /* --##----- */
3157:      0x60,    /* -##----- */
3158:      0x00,    /* ----- */
3159:      0x78,    /* -#####--- */
3160:      0x0c,    /* -----##-- */
3161:      0x7c,    /* -#####-- */
3162:      0xcc,    /* ##--##-- */
3163:      0xdc,    /* #-###-- */
3164:      0x76,    /* -###-##- */
3165:      0x00,    /* ----- */
3166:      0x00,    /* ----- */
3167:
3168:          /* 0xe2      */
3169:      0x30,    /* --##----- */
3170:      0x78,    /* -#####--- */
3171:      0xcc,    /* ##--##-- */
3172:      0x00,    /* ----- */
3173:      0x78,    /* -#####--- */
3174:      0x0c,    /* -----##-- */
3175:      0x7c,    /* -#####-- */
3176:      0xcc,    /* ##--##-- */
3177:      0xdc,    /* #-###-- */
3178:      0x76,    /* -###-##- */
3179:      0x00,    /* ----- */
3180:      0x00,    /* ----- */
3181:
3182:          /* 0xe3      */
3183:      0x00,    /* ----- */
3184:      0x76,    /* -###-##- */
3185:      0xdc,    /* #-###-- */
3186:      0x00,    /* ----- */
3187:      0x78,    /* -#####--- */
3188:      0x0c,    /* -----##-- */
3189:      0x7c,    /* -#####-- */
3190:      0xcc,    /* ##--##-- */
3191:      0xdc,    /* #-###-- */
3192:      0x76,    /* -###-##- */
3193:      0x00,    /* ----- */
3194:      0x00,    /* ----- */
3195:
3196:          /* 0xe4      */
3197:      0x00,    /* ----- */
3198:      0x6c,    /* -##-##-- */
3199:      0x6c,    /* -##-##-- */
3200:      0x00,    /* ----- */
3201:      0x78,    /* -#####--- */
3202:      0x0c,    /* -----##-- */
3203:      0x7c,    /* -#####-- */
3204:      0xcc,    /* ##--##-- */
3205:      0xdc,    /* #-###-- */
3206:      0x76,    /* -###-##- */
3207:      0x00,    /* ----- */
3208:      0x00,    /* ----- */
3209:
3210:          /* 0xe5      */
3211:      0x38,    /* ---###---- */
3212:      0x6c,    /* -##-##-- */
3213:      0x38,    /* ---###---- */
3214:      0x00,    /* ----- */
3215:      0x78,    /* -#####--- */
3216:      0x0c,    /* -----##-- */

```

drivers/video/font-lat9-8x12.c

Page 49/54

```

3217:      0x7c,    /* #####-- */
3218:      0xcc,    /* ##---##-- */
3219:      0xdc,    /* ##-###-- */
3220:      0x76,    /* ---###-#- */
3221:      0x00,    /* ----- */
3222:      0x00,    /* ----- */
3223:
3224:          /* 0xe6 */
3225:      0x00,    /* ----- */
3226:      0x00,    /* ----- */
3227:      0x00,    /* ----- */
3228:      0x7e,    /* -#####- */
3229:      0xdb,    /* #-##-## */
3230:      0x1b,    /* ---##-## */
3231:      0x7f,    /* -##### */
3232:      0xd8,    /* ##-##-#- */
3233:      0xdb,    /* #-##-## */
3234:      0x7e,    /* -#####- */
3235:      0x00,    /* ----- */
3236:      0x00,    /* ----- */
3237:
3238:          /* 0xe7 */
3239:      0x00,    /* ----- */
3240:      0x00,    /* ----- */
3241:      0x00,    /* ----- */
3242:      0x7c,    /* -#####-- */
3243:      0xc6,    /* ##---##- */
3244:      0xc0,    /* ##----- */
3245:      0xc0,    /* ##----- */
3246:      0xc6,    /* ##---##- */
3247:      0x7c,    /* -#####-- */
3248:      0x18,    /* ---##-#- */
3249:      0x6c,    /* -##-##-#- */
3250:      0x38,    /* --###-#- */
3251:
3252:          /* 0xe8 */
3253:      0x30,    /* --##----- */
3254:      0x18,    /* ---##-#-#- */
3255:      0x0c,    /* ----##--#- */
3256:      0x00,    /* ----- */
3257:      0x7c,    /* -#####-- */
3258:      0xc6,    /* ##---##- */
3259:      0xfe,    /* ########- */
3260:      0xc0,    /* ##----- */
3261:      0xc6,    /* ##---##- */
3262:      0x7c,    /* -#####-- */
3263:      0x00,    /* ----- */
3264:      0x00,    /* ----- */
3265:
3266:          /* 0xe9 */
3267:      0x0c,    /* ----##--#- */
3268:      0x18,    /* ---##-#-#- */
3269:      0x30,    /* --##----- */
3270:      0x00,    /* ----- */
3271:      0x7c,    /* -#####-- */
3272:      0xc6,    /* ##---##- */
3273:      0xfe,    /* ########- */
3274:      0xc0,    /* ##----- */
3275:      0xc6,    /* ##---##- */
3276:      0x7c,    /* -#####-- */
3277:      0x00,    /* ----- */
3278:      0x00,    /* ----- */
3279:
3280:          /* 0xea */
3281:      0x10,    /* ---#----- */
3282:      0x38,    /* --###----- */
3283:      0x6c,    /* -##-##-#-#- */

```

drivers/video/font-lat9-8x12.c

Page 50/54

```

3284:      0x00, /* ----- */
3285:      0x7c, /* -##### */
3286:      0xc6, /* ##---## */
3287:      0xfe, /* #####- */
3288:      0xc0, /* ##----- */
3289:      0xc6, /* ##---## */
3290:      0x7c, /* -##### */
3291:      0x00, /* ----- */
3292:      0x00, /* ----- */
3293:
3294:          /* 0xeb */
3295:      0x00, /* ----- */
3296:      0x6c, /* -##-##- */
3297:      0x6c, /* -##-##- */
3298:      0x00, /* ----- */
3299:      0x7c, /* -##### */
3300:      0xc6, /* ##---## */
3301:      0xfe, /* #####- */
3302:      0xc0, /* ##----- */
3303:      0xc6, /* ##---## */
3304:      0x7c, /* -##### */
3305:      0x00, /* ----- */
3306:      0x00, /* ----- */
3307:
3308:          /* 0xec */
3309:      0x60, /* -##----- */
3310:      0x30, /* --##----- */
3311:      0x18, /* ---##----- */
3312:      0x00, /* ----- */
3313:      0x38, /* --###----- */
3314:      0x18, /* ---##----- */
3315:      0x18, /* ---##----- */
3316:      0x18, /* ---##----- */
3317:      0x18, /* ---##----- */
3318:      0x3c, /* --####----- */
3319:      0x00, /* ----- */
3320:      0x00, /* ----- */
3321:
3322:          /* 0xed */
3323:      0x0c, /* ----##----- */
3324:      0x18, /* ---##----- */
3325:      0x30, /* --##----- */
3326:      0x00, /* ----- */
3327:      0x38, /* --###----- */
3328:      0x18, /* ---##----- */
3329:      0x18, /* ---##----- */
3330:      0x18, /* ---##----- */
3331:      0x18, /* ---##----- */
3332:      0x3c, /* --####----- */
3333:      0x00, /* ----- */
3334:      0x00, /* ----- */
3335:
3336:          /* 0xee */
3337:      0x18, /* ---##----- */
3338:      0x3c, /* --####----- */
3339:      0x66, /* -##---##- */
3340:      0x00, /* ----- */
3341:      0x38, /* --###----- */
3342:      0x18, /* ---##----- */
3343:      0x18, /* ---##----- */
3344:      0x18, /* ---##----- */
3345:      0x18, /* ---##----- */
3346:      0x3c, /* --####----- */
3347:      0x00, /* ----- */
3348:      0x00, /* ----- */
3349:
3350:          /* 0xef */

```

drivers/video/font-lat9-8x12.c

Page 51/54

```

3351:      0x00, /* ----- */
3352:      0x6c, /* -##-##- */
3353:      0x6c, /* -##-##- */
3354:      0x00, /* ----- */
3355:      0x38, /* ---###- */
3356:      0x18, /* ---##-  */
3357:      0x18, /* ---##-  */
3358:      0x18, /* ---##-  */
3359:      0x18, /* ---##-  */
3360:      0x3c, /* ---###- */
3361:      0x00, /* ----- */
3362:      0x00, /* ----- */
3363:
3364:          /* 0xf0 */
3365:      0x78, /* -####-  */
3366:      0x30, /* --##-   */
3367:      0x78, /* -####-  */
3368:      0x0c, /* ----##- */
3369:      0x7e, /* -#####- */
3370:      0xc6, /* ##---##- */
3371:      0xc6, /* ##---##- */
3372:      0xc6, /* ##---##- */
3373:      0xc6, /* ##---##- */
3374:      0x7c, /* -#####- */
3375:      0x00, /* ----- */
3376:      0x00, /* ----- */
3377:
3378:          /* 0xf1 */
3379:      0x00, /* ----- */
3380:      0x76, /* -###-##- */
3381:      0xdc, /* ##-##-  */
3382:      0x00, /* ----- */
3383:      0xdc, /* ##-##-  */
3384:      0x66, /* -##-##- */
3385:      0x66, /* -##-##- */
3386:      0x66, /* -##-##- */
3387:      0x66, /* -##-##- */
3388:      0x66, /* -##-##- */
3389:      0x00, /* ----- */
3390:      0x00, /* ----- */
3391:
3392:          /* 0xf2 */
3393:      0x60, /* -##----- */
3394:      0x30, /* --##----- */
3395:      0x18, /* ---##----- */
3396:      0x00, /* ----- */
3397:      0x7c, /* -#####- */
3398:      0xc6, /* ##---##- */
3399:      0xc6, /* ##---##- */
3400:      0xc6, /* ##---##- */
3401:      0xc6, /* ##---##- */
3402:      0x7c, /* -#####- */
3403:      0x00, /* ----- */
3404:      0x00, /* ----- */
3405:
3406:          /* 0xf3 */
3407:      0x0c, /* ----##-  */
3408:      0x18, /* ---##-   */
3409:      0x30, /* --##-   */
3410:      0x00, /* ----- */
3411:      0x7c, /* -#####- */
3412:      0xc6, /* ##---##- */
3413:      0xc6, /* ##---##- */
3414:      0xc6, /* ##---##- */
3415:      0xc6, /* ##---##- */
3416:      0x7c, /* -#####- */
3417:      0x00, /* ----- */

```

drivers/video/font-lat9-8x12.c

Page 52/54

```

3418:      0x00, /* ----- */
3419:
3420:      /* 0xf4 */
3421:      0x10, /* ---#---- */
3422:      0x38, /* --###--- */
3423:      0x6c, /* -##-##-- */
3424:      0x00, /* ----- */
3425:      0x7c, /* -#####-- */
3426:      0xc6, /* ##---##- */
3427:      0xc6, /* ##---##- */
3428:      0xc6, /* ##---##- */
3429:      0xc6, /* ##---##- */
3430:      0x7c, /* -#####-- */
3431:      0x00, /* ----- */
3432:      0x00, /* ----- */
3433:
3434:      /* 0xf5 */
3435:      0x00, /* ----- */
3436:      0x76, /* -###-##- */
3437:      0xdc, /* ##-###-- */
3438:      0x00, /* ----- */
3439:      0x7c, /* -#####-- */
3440:      0xc6, /* ##---##- */
3441:      0xc6, /* ##---##- */
3442:      0xc6, /* ##---##- */
3443:      0xc6, /* ##---##- */
3444:      0x7c, /* -#####-- */
3445:      0x00, /* ----- */
3446:      0x00, /* ----- */
3447:
3448:      /* 0xf6 */
3449:      0x00, /* ----- */
3450:      0x6c, /* -##-##-- */
3451:      0x6c, /* -##-##-- */
3452:      0x00, /* ----- */
3453:      0x7c, /* -#####-- */
3454:      0xc6, /* ##---##- */
3455:      0xc6, /* ##---##- */
3456:      0xc6, /* ##---##- */
3457:      0xc6, /* ##---##- */
3458:      0x7c, /* -#####-- */
3459:      0x00, /* ----- */
3460:      0x00, /* ----- */
3461:
3462:      /* 0xf7 */
3463:      0x00, /* ----- */
3464:      0x00, /* ----- */
3465:      0x18, /* ---##--- */
3466:      0x18, /* ---##--- */
3467:      0x00, /* ----- */
3468:      0x7e, /* -#####-- */
3469:      0x00, /* ----- */
3470:      0x18, /* ---##--- */
3471:      0x18, /* ---##--- */
3472:      0x00, /* ----- */
3473:      0x00, /* ----- */
3474:      0x00, /* ----- */
3475:
3476:      /* 0xf8 */
3477:      0x00, /* ----- */
3478:      0x00, /* ----- */
3479:      0x00, /* ----- */
3480:      0x00, /* ----- */
3481:      0x7e, /* -#####-- */
3482:      0xce, /* ##-###-- */
3483:      0xde, /* ##-###-- */
3484:      0xf6, /* #####-##- */

```

drivers/video/font-lat9-8x12.c

Page 53/54

```

3485:      0xe6,      /* ###--##- */
3486:      0xfc,      /* #####-- */
3487:      0x00,      /* ----- */
3488:      0x00,      /* ----- */
3489:
3490:          /* 0xf9          */
3491:      0xc0,      /* ##----- */
3492:      0x60,      /* -##----- */
3493:      0x30,      /* --##----- */
3494:      0x00,      /* ----- */
3495:      0xcc,      /* ##--##-- */
3496:      0xcc,      /* ##--##-- */
3497:      0xcc,      /* ##--##-- */
3498:      0xcc,      /* ##--##-- */
3499:      0xcc,      /* ##--##-- */
3500:      0x76,      /* -###-##- */
3501:      0x00,      /* ----- */
3502:      0x00,      /* ----- */
3503:
3504:          /* 0xfa          */
3505:      0x0c,      /* ----##-- */
3506:      0x18,      /* ---##--- */
3507:      0x30,      /* --##----- */
3508:      0x00,      /* ----- */
3509:      0xcc,      /* ##--##-- */
3510:      0xcc,      /* ##--##-- */
3511:      0xcc,      /* ##--##-- */
3512:      0xcc,      /* ##--##-- */
3513:      0xcc,      /* ##--##-- */
3514:      0x76,      /* -###-##- */
3515:      0x00,      /* ----- */
3516:      0x00,      /* ----- */
3517:
3518:          /* 0xfb          */
3519:      0x30,      /* --##----- */
3520:      0x78,      /* -#####- */
3521:      0xcc,      /* ##--##-- */
3522:      0x00,      /* ----- */
3523:      0xcc,      /* ##--##-- */
3524:      0xcc,      /* ##--##-- */
3525:      0xcc,      /* ##--##-- */
3526:      0xcc,      /* ##--##-- */
3527:      0xcc,      /* ##--##-- */
3528:      0x76,      /* -###-##- */
3529:      0x00,      /* ----- */
3530:      0x00,      /* ----- */
3531:
3532:          /* 0xfc          */
3533:      0x00,      /* ----- */
3534:      0xcc,      /* ##--##-- */
3535:      0xcc,      /* ##--##-- */
3536:      0x00,      /* ----- */
3537:      0xcc,      /* ##--##-- */
3538:      0xcc,      /* ##--##-- */
3539:      0xcc,      /* ##--##-- */
3540:      0xcc,      /* ##--##-- */
3541:      0xcc,      /* ##--##-- */
3542:      0x76,      /* -###-##- */
3543:      0x00,      /* ----- */
3544:      0x00,      /* ----- */
3545:
3546:          /* 0xfd          */
3547:      0x0c,      /* ----##-- */
3548:      0x18,      /* ---##--- */
3549:      0x30,      /* --##----- */
3550:      0x00,      /* ----- */
3551:      0xc6,      /* ##--##-- */

```

drivers/video/font-lat9-8x12.c

Page 54/54

```

3552:      0xc6,    /* ##---##- */
3553:      0xc6,    /* ##---##- */
3554:      0xce,    /* ##---##- */
3555:      0x76,    /* -###-##- */
3556:      0x06,    /* -----##- */
3557:      0xc6,    /* ##---##- */
3558:      0x7c,    /* -#####- */
3559:
3560:          /* 0xfe          */
3561:      0x00,    /* ----- */
3562:      0xf0,    /* ####---- */
3563:      0x60,    /* -##----- */
3564:      0x60,    /* -##----- */
3565:      0x78,    /* -####---- */
3566:      0x6c,    /* -##-##-- */
3567:      0x6c,    /* -##-##-- */
3568:      0x6c,    /* -##-##-- */
3569:      0x78,    /* -####---- */
3570:      0x60,    /* -##----- */
3571:      0x60,    /* -##----- */
3572:      0xf0,    /* ####---- */
3573:
3574:          /* 0xff          */
3575:      0x00,    /* ----- */
3576:      0xc6,    /* ##---##- */
3577:      0xc6,    /* ##---##- */
3578:      0x00,    /* ----- */
3579:      0xc6,    /* ##---##- */
3580:      0xc6,    /* ##---##- */
3581:      0xc6,    /* ##---##- */
3582:      0xce,    /* ##---##- */
3583:      0x76,    /* -###-##- */
3584:      0x06,    /* -----##- */
3585:      0xc6,    /* ##---##- */
3586:      0x7c,    /* -#####- */
3587:
3588: };
3589:
3590: static unsigned char cursorshape_8x12[] = {
3591:      0x00,    /* ----- */
3592:      0x00,    /* ----- */
3593:      0x00,    /* ----- */
3594:      0x00,    /* ----- */
3595:      0x00,    /* ----- */
3596:      0x00,    /* ----- */
3597:      0x00,    /* ----- */
3598:      0x00,    /* ----- */
3599:      0x00,    /* ----- */
3600:      0x00,    /* ----- */
3601:      0xFF,    /* ##### */
3602:      0xFF,    /* ##### */
3603: };
3604:
3605: struct fbcon_font_desc font_lat9_8x12 = {
3606:      "VGA8x12",
3607:      8,
3608:      12,
3609:      fontdata_8x12,
3610:      cursorshape_8x12
3611: };

```


drivers/video/font-lat9-8x14.c

Page 1/62

```

1: #include <fiwix/font.h>
2:
3: static unsigned char fontdata_8x14[] = {
4:     /* 0x00          */
5:     0x00, /* ----- */
6:     0x7e, /* -#####- */
7:     0xc3, /* ##----## */
8:     0x99, /* #--##--# */
9:     0x99, /* #--##--# */
10:    0xf3, /* #####--## */
11:    0xe7, /* ###--### */
12:    0xe7, /* ###--### */
13:    0xf7, /* ######## */
14:    0xe7, /* ###--### */
15:    0xe7, /* ###--### */
16:    0x7e, /* -#####- */
17:    0x00, /* ----- */
18:    0x00, /* ----- */
19:
20:     /* 0x01          */
21:    0x00, /* ----- */
22:    0x00, /* ----- */
23:    0x00, /* ----- */
24:    0x00, /* ----- */
25:    0x76, /* -###-##- */
26:    0xdc, /* #-###-- */
27:    0x00, /* ----- */
28:    0x76, /* -###-##- */
29:    0xdc, /* #-###-- */
30:    0x00, /* ----- */
31:    0x00, /* ----- */
32:    0x00, /* ----- */
33:    0x00, /* ----- */
34:    0x00, /* ----- */
35:
36:     /* 0x02          */
37:    0x00, /* ----- */
38:    0x6e, /* -##-###- */
39:    0xd8, /* #-##--- */
40:    0xd8, /* #-##--- */
41:    0xd8, /* #-##--- */
42:    0xd8, /* #-##--- */
43:    0xde, /* #-#####- */
44:    0xd8, /* #-##--- */
45:    0xd8, /* #-##--- */
46:    0xd8, /* #-##--- */
47:    0x6e, /* -##-###- */
48:    0x00, /* ----- */
49:    0x00, /* ----- */
50:    0x00, /* ----- */
51:
52:     /* 0x03          */
53:    0x00, /* ----- */
54:    0x00, /* ----- */
55:    0x00, /* ----- */
56:    0x00, /* ----- */
57:    0x6e, /* -##-###- */
58:    0xdb, /* #-##-## */
59:    0xdb, /* #-##-## */
60:    0xdf, /* #-##### */
61:    0xd8, /* #-##--- */
62:    0xdb, /* #-##-## */
63:    0x6e, /* -##-###- */
64:    0x00, /* ----- */
65:    0x00, /* ----- */
66:    0x00, /* ----- */
67:

```

drivers/video/font-lat9-8x14.c

Page 2/62

```

68:                /* 0x04          */
69:                0x00, /* ----- */
70:                0x00, /* ----- */
71:                0x00, /* ----- */
72:                0x10, /* ----#---- */
73:                0x38, /* --###--- */
74:                0x7c, /* -#####- */
75:                0xfe, /* #####- */
76:                0x7c, /* -#####- */
77:                0x38, /* --###--- */
78:                0x10, /* ----#---- */
79:                0x00, /* ----- */
80:                0x00, /* ----- */
81:                0x00, /* ----- */
82:                0x00, /* ----- */
83:
84:                /* 0x05          */
85:                0x00, /* ----- */
86:                0x88, /* #---#--- */
87:                0x88, /* #---#--- */
88:                0xf8, /* #####--- */
89:                0x88, /* #---#--- */
90:                0x88, /* #---#--- */
91:                0x00, /* ----- */
92:                0x3e, /* --#####- */
93:                0x08, /* ----#---- */
94:                0x08, /* ----#---- */
95:                0x08, /* ----#---- */
96:                0x08, /* ----#---- */
97:                0x00, /* ----- */
98:                0x00, /* ----- */
99:
100:               /* 0x06          */
101:               0x00, /* ----- */
102:               0xf8, /* #####--- */
103:               0x80, /* #----- */
104:               0xe0, /* ###----- */
105:               0x80, /* #----- */
106:               0x80, /* #----- */
107:               0x00, /* ----- */
108:               0x3e, /* --#####- */
109:               0x20, /* --#----- */
110:               0x38, /* --###--- */
111:               0x20, /* --#----- */
112:               0x20, /* --#----- */
113:               0x00, /* ----- */
114:               0x00, /* ----- */
115:
116:               /* 0x07          */
117:               0x00, /* ----- */
118:               0x78, /* -#####- */
119:               0x80, /* #----- */
120:               0x80, /* #----- */
121:               0x80, /* #----- */
122:               0x78, /* -#####- */
123:               0x00, /* ----- */
124:               0x3c, /* --#####- */
125:               0x22, /* --#---#- */
126:               0x3e, /* --#####- */
127:               0x24, /* --#---#- */
128:               0x22, /* --#---#- */
129:               0x00, /* ----- */
130:               0x00, /* ----- */
131:
132:               /* 0x08          */
133:               0x00, /* ----- */
134:               0x80, /* #----- */

```

drivers/video/font-lat9-8x14.c

Page 3/62

```

135:      0x80,    /* #----- */
136:      0x80,    /* #----- */
137:      0x80,    /* #----- */
138:      0xf8,    /* #####--- */
139:      0x00,    /* ----- */
140:      0x3e,    /* --#####- */
141:      0x20,    /* --#----- */
142:      0x38,    /* --###---- */
143:      0x20,    /* --#----- */
144:      0x20,    /* --#----- */
145:      0x00,    /* ----- */
146:      0x00,    /* ----- */
147:
148:          /* 0x09          */
149:      0x88,    /* #---#--- */
150:      0x22,    /* --#---#- */
151:      0x88,    /* #---#--- */
152:      0x22,    /* --#---#- */
153:      0x88,    /* #---#--- */
154:      0x22,    /* --#---#- */
155:      0x88,    /* #---#--- */
156:      0x22,    /* --#---#- */
157:      0x88,    /* #---#--- */
158:      0x22,    /* --#---#- */
159:      0x88,    /* #---#--- */
160:      0x22,    /* --#---#- */
161:      0x88,    /* #---#--- */
162:      0x22,    /* --#---#- */
163:
164:          /* 0x0a          */
165:      0xaa,    /* #-#-#-#- */
166:      0x55,    /* -#-#-#-# */
167:      0xaa,    /* #-#-#-#- */
168:      0x55,    /* -#-#-#-# */
169:      0xaa,    /* #-#-#-#- */
170:      0x55,    /* -#-#-#-# */
171:      0xaa,    /* #-#-#-#- */
172:      0x55,    /* -#-#-#-# */
173:      0xaa,    /* #-#-#-#- */
174:      0x55,    /* -#-#-#-# */
175:      0xaa,    /* #-#-#-#- */
176:      0x55,    /* -#-#-#-# */
177:      0xaa,    /* #-#-#-#- */
178:      0x55,    /* -#-#-#-# */
179:
180:          /* 0x0b          */
181:      0xbb,    /* #-###-## */
182:      0xee,    /* ###-###- */
183:      0xbb,    /* #-###-## */
184:      0xee,    /* ###-###- */
185:      0xbb,    /* #-###-## */
186:      0xee,    /* ###-###- */
187:      0xbb,    /* #-###-## */
188:      0xee,    /* ###-###- */
189:      0xbb,    /* #-###-## */
190:      0xee,    /* ###-###- */
191:      0xbb,    /* #-###-## */
192:      0xee,    /* ###-###- */
193:      0xbb,    /* #-###-## */
194:      0xee,    /* ###-###- */
195:
196:          /* 0x0c          */
197:      0xff,    /* ##### */
198:      0xff,    /* ##### */
199:      0xff,    /* ##### */
200:      0xff,    /* ##### */
201:      0xff,    /* ##### */

```

drivers/video/font-lat9-8x14.c

Page 4/62

```

202:      0xff, /* ##### */
203:      0xff, /* ##### */
204:      0xff, /* ##### */
205:      0xff, /* ##### */
206:      0xff, /* ##### */
207:      0xff, /* ##### */
208:      0xff, /* ##### */
209:      0xff, /* ##### */
210:      0xff, /* ##### */
211:
212:          /* 0x0d */
213:      0x00, /* ----- */
214:      0x00, /* ----- */
215:      0x00, /* ----- */
216:      0x00, /* ----- */
217:      0x00, /* ----- */
218:      0x00, /* ----- */
219:      0x00, /* ----- */
220:      0xff, /* ##### */
221:      0xff, /* ##### */
222:      0xff, /* ##### */
223:      0xff, /* ##### */
224:      0xff, /* ##### */
225:      0xff, /* ##### */
226:      0xff, /* ##### */
227:
228:          /* 0x0e */
229:      0xff, /* ##### */
230:      0xff, /* ##### */
231:      0xff, /* ##### */
232:      0xff, /* ##### */
233:      0xff, /* ##### */
234:      0xff, /* ##### */
235:      0xff, /* ##### */
236:      0x00, /* ----- */
237:      0x00, /* ----- */
238:      0x00, /* ----- */
239:      0x00, /* ----- */
240:      0x00, /* ----- */
241:      0x00, /* ----- */
242:      0x00, /* ----- */
243:
244:          /* 0x0f */
245:      0xf0, /* ####---- */
246:      0xf0, /* ####---- */
247:      0xf0, /* ####---- */
248:      0xf0, /* ####---- */
249:      0xf0, /* ####---- */
250:      0xf0, /* ####---- */
251:      0xf0, /* ####---- */
252:      0xf0, /* ####---- */
253:      0xf0, /* ####---- */
254:      0xf0, /* ####---- */
255:      0xf0, /* ####---- */
256:      0xf0, /* ####---- */
257:      0xf0, /* ####---- */
258:      0xf0, /* ####---- */
259:
260:          /* 0x10 */
261:      0x0f, /* ----#### */
262:      0x0f, /* ----#### */
263:      0x0f, /* ----#### */
264:      0x0f, /* ----#### */
265:      0x0f, /* ----#### */
266:      0x0f, /* ----#### */
267:      0x0f, /* ----#### */
268:      0x0f, /* ----#### */

```

drivers/video/font-lat9-8x14.c

```

269:      0x0f, /* ----#### */
270:      0x0f, /* ----#### */
271:      0x0f, /* ----#### */
272:      0x0f, /* ----#### */
273:      0x0f, /* ----#### */
274:      0x0f, /* ----#### */
275:
276:          /* 0x11 */
277:      0x00, /* ----- */
278:      0x88, /* #---#--- */
279:      0xc8, /* ##--#--- */
280:      0xa8, /* #-#-#--- */
281:      0x98, /* #--#-#--- */
282:      0x88, /* #---#--- */
283:      0x00, /* ----- */
284:      0x20, /* --#----- */
285:      0x20, /* --#----- */
286:      0x20, /* --#----- */
287:      0x20, /* --#----- */
288:      0x3e, /* --#####- */
289:      0x00, /* ----- */
290:      0x00, /* ----- */
291:
292:          /* 0x12 */
293:      0x00, /* ----- */
294:      0x88, /* #---#--- */
295:      0x88, /* #---#--- */
296:      0x50, /* -#-#-#--- */
297:      0x50, /* -#-#-#--- */
298:      0x20, /* --#----- */
299:      0x00, /* ----- */
300:      0x3e, /* --#####- */
301:      0x08, /* ----#---- */
302:      0x08, /* ----#---- */
303:      0x08, /* ----#---- */
304:      0x08, /* ----#---- */
305:      0x00, /* ----- */
306:      0x00, /* ----- */
307:
308:          /* 0x13 */
309:      0x00, /* ----- */
310:      0x00, /* ----- */
311:      0x00, /* ----- */
312:      0x06, /* ----##- */
313:      0x0c, /* ----##- */
314:      0x18, /* ---##-#- */
315:      0x30, /* --##-#-#- */
316:      0x7e, /* -#####- */
317:      0x00, /* ----- */
318:      0x7e, /* -#####- */
319:      0x00, /* ----- */
320:      0x00, /* ----- */
321:      0x00, /* ----- */
322:      0x00, /* ----- */
323:
324:          /* 0x14 */
325:      0x00, /* ----- */
326:      0x00, /* ----- */
327:      0x00, /* ----- */
328:      0x60, /* -##-#-#-#- */
329:      0x30, /* ---##-#-#- */
330:      0x18, /* ----##-#-#- */
331:      0x0c, /* ----##-#-#- */
332:      0x7e, /* -#####- */
333:      0x00, /* ----- */
334:      0x7e, /* -#####- */
335:      0x00, /* ----- */

```

drivers/video/font-lat9-8x14.c

Page 6/62

```

336:      0x00, /* ----- */
337:      0x00, /* ----- */
338:      0x00, /* ----- */
339:
340:          /* 0x15 */
341:      0x00, /* ----- */
342:      0x00, /* ----- */
343:      0x00, /* ----- */
344:      0x06, /* -----##- */
345:      0x0c, /* -----##- */
346:      0xfe, /* #####- */
347:      0x38, /* --###- */
348:      0xfe, /* #####- */
349:      0x60, /* -##----- */
350:      0xc0, /* ##----- */
351:      0x00, /* ----- */
352:      0x00, /* ----- */
353:      0x00, /* ----- */
354:      0x00, /* ----- */
355:
356:          /* 0x16 */
357:      0x00, /* ----- */
358:      0x00, /* ----- */
359:      0x02, /* -----#- */
360:      0x0e, /* ----###- */
361:      0x3e, /* --#####- */
362:      0x7e, /* -#####- */
363:      0xfe, /* #####- */
364:      0x7e, /* -#####- */
365:      0x3e, /* --#####- */
366:      0x0e, /* ----###- */
367:      0x02, /* -----#- */
368:      0x00, /* ----- */
369:      0x00, /* ----- */
370:      0x00, /* ----- */
371:
372:          /* 0x17 */
373:      0x00, /* ----- */
374:      0x00, /* ----- */
375:      0x80, /* #----- */
376:      0xe0, /* ###----- */
377:      0xf0, /* ####----- */
378:      0xfc, /* #####-- */
379:      0xfe, /* #####- */
380:      0xfc, /* #####-- */
381:      0xf0, /* ####----- */
382:      0xe0, /* ###----- */
383:      0x80, /* #----- */
384:      0x00, /* ----- */
385:      0x00, /* ----- */
386:      0x00, /* ----- */
387:
388:          /* 0x18 */
389:      0x00, /* ----- */
390:      0x00, /* ----- */
391:      0x18, /* ---##--- */
392:      0x3c, /* --####- */
393:      0x7e, /* -#####- */
394:      0x18, /* ---##--- */
395:      0x18, /* ----#--- */
396:      0x18, /* ----#--- */
397:      0x18, /* ----#--- */
398:      0x18, /* ----#--- */
399:      0x18, /* ----#--- */
400:      0x00, /* ----- */
401:      0x00, /* ----- */
402:      0x00, /* ----- */

```

drivers/video/font-lat9-8x14.c

Page 7/62

```

403:
404:          /* 0x19          */
405:          0x00,          /* ----- */
406:          0x00,          /* ----- */
407:          0x18,          /* ---##--- */
408:          0x18,          /* ---##--- */
409:          0x18,          /* ---##--- */
410:          0x18,          /* ---##--- */
411:          0x18,          /* ---##--- */
412:          0x18,          /* ---##--- */
413:          0x7e,          /* -#####- */
414:          0x3c,          /* --#####- */
415:          0x18,          /* ---##--- */
416:          0x00,          /* ----- */
417:          0x00,          /* ----- */
418:          0x00,          /* ----- */
419:
420:          /* 0x1a          */
421:          0x00,          /* ----- */
422:          0x00,          /* ----- */
423:          0x00,          /* ----- */
424:          0x00,          /* ----- */
425:          0x18,          /* ---##--- */
426:          0x0c,          /* ----##-- */
427:          0xfe,          /* #####- */
428:          0x0c,          /* ----##-- */
429:          0x18,          /* ---##--- */
430:          0x00,          /* ----- */
431:          0x00,          /* ----- */
432:          0x00,          /* ----- */
433:          0x00,          /* ----- */
434:          0x00,          /* ----- */
435:
436:          /* 0x1b          */
437:          0x00,          /* ----- */
438:          0x00,          /* ----- */
439:          0x00,          /* ----- */
440:          0x00,          /* ----- */
441:          0x30,          /* --##---- */
442:          0x60,          /* -##----- */
443:          0xfe,          /* #####- */
444:          0x60,          /* -##----- */
445:          0x30,          /* --##---- */
446:          0x00,          /* ----- */
447:          0x00,          /* ----- */
448:          0x00,          /* ----- */
449:          0x00,          /* ----- */
450:          0x00,          /* ----- */
451:
452:          /* 0x1c          */
453:          0x00,          /* ----- */
454:          0x00,          /* ----- */
455:          0x18,          /* ---##--- */
456:          0x3c,          /* --#####- */
457:          0x7e,          /* -#####- */
458:          0x18,          /* ---##--- */
459:          0x18,          /* ---##--- */
460:          0x18,          /* ---##--- */
461:          0x7e,          /* -#####- */
462:          0x3c,          /* --#####- */
463:          0x18,          /* ---##--- */
464:          0x00,          /* ----- */
465:          0x00,          /* ----- */
466:          0x00,          /* ----- */
467:
468:          /* 0x1d          */
469:          0x00,          /* ----- */

```

drivers/video/font-lat9-8x14.c

Page 8/62

```

470:      0x00, /* ----- */
471:      0x00, /* ----- */
472:      0x00, /* ----- */
473:      0x28, /* --#-#--- */
474:      0x6c, /* -##-##-- */
475:      0xfe, /* #####- */
476:      0x6c, /* -##-##-- */
477:      0x28, /* --#-#--- */
478:      0x00, /* ----- */
479:      0x00, /* ----- */
480:      0x00, /* ----- */
481:      0x00, /* ----- */
482:      0x00, /* ----- */
483:
484:          /* 0x1e */
485:      0x00, /* ----- */
486:      0x00, /* ----- */
487:      0x06, /* -----##- */
488:      0x06, /* -----##- */
489:      0x36, /* --##-##- */
490:      0x66, /* -##-##- */
491:      0xfe, /* #####- */
492:      0x60, /* -##------ */
493:      0x30, /* --##------ */
494:      0x00, /* ----- */
495:      0x00, /* ----- */
496:      0x00, /* ----- */
497:      0x00, /* ----- */
498:      0x00, /* ----- */
499:
500:          /* 0x1f */
501:      0x00, /* ----- */
502:      0x00, /* ----- */
503:      0x00, /* ----- */
504:      0x00, /* ----- */
505:      0xc0, /* ##----- */
506:      0x7c, /* -#####- */
507:      0x6e, /* -##-##-#- */
508:      0x6c, /* -##-##-#- */
509:      0x6c, /* -##-##-#- */
510:      0x6c, /* -##-##-#- */
511:      0x6c, /* -##-##-#- */
512:      0x00, /* ----- */
513:      0x00, /* ----- */
514:      0x00, /* ----- */
515:
516:          /* 0x20 (' ') */
517:      0x00, /* ----- */
518:      0x00, /* ----- */
519:      0x00, /* ----- */
520:      0x00, /* ----- */
521:      0x00, /* ----- */
522:      0x00, /* ----- */
523:      0x00, /* ----- */
524:      0x00, /* ----- */
525:      0x00, /* ----- */
526:      0x00, /* ----- */
527:      0x00, /* ----- */
528:      0x00, /* ----- */
529:      0x00, /* ----- */
530:      0x00, /* ----- */
531:
532:          /* 0x21 ('!') */
533:      0x00, /* ----- */
534:      0x00, /* ----- */
535:      0x18, /* ----##- */
536:      0x3c, /* --####- */

```


drivers/video/font-lat9-8x14.c

Page 9/62

```

537:      0x3c, /* ---###-- */
538:      0x3c, /* ---###-- */
539:      0x18, /* ----##---- */
540:      0x18, /* ----##---- */
541:      0x00, /* ----- */
542:      0x18, /* ----##---- */
543:      0x18, /* ----##---- */
544:      0x00, /* ----- */
545:      0x00, /* ----- */
546:      0x00, /* ----- */
547:
548:      /* 0x22 ('"') */
549:      0x00, /* ----- */
550:      0x00, /* ----- */
551:      0x36, /* --##-##- */
552:      0x36, /* --##-##- */
553:      0x14, /* ----#-#- */
554:      0x00, /* ----- */
555:      0x00, /* ----- */
556:      0x00, /* ----- */
557:      0x00, /* ----- */
558:      0x00, /* ----- */
559:      0x00, /* ----- */
560:      0x00, /* ----- */
561:      0x00, /* ----- */
562:      0x00, /* ----- */
563:
564:      /* 0x23 ('#') */
565:      0x00, /* ----- */
566:      0x00, /* ----- */
567:      0x6c, /* -##-##- */
568:      0xfe, /* #####- */
569:      0x6c, /* -##-##- */
570:      0x6c, /* -##-##- */
571:      0xfe, /* #####- */
572:      0x6c, /* -##-##- */
573:      0x00, /* ----- */
574:      0x00, /* ----- */
575:      0x00, /* ----- */
576:      0x00, /* ----- */
577:      0x00, /* ----- */
578:      0x00, /* ----- */
579:
580:      /* 0x24 ('$') */
581:      0x00, /* ----- */
582:      0x00, /* ----- */
583:      0x10, /* ---#---- */
584:      0x7c, /* -#####- */
585:      0xd6, /* ##-##-##- */
586:      0x70, /* -###----- */
587:      0x38, /* --###---- */
588:      0x1c, /* ---###---- */
589:      0xd6, /* ##-##-##- */
590:      0x7c, /* -#####- */
591:      0x10, /* ---#---- */
592:      0x00, /* ----- */
593:      0x00, /* ----- */
594:      0x00, /* ----- */
595:
596:      /* 0x25 ('%') */
597:      0x00, /* ----- */
598:      0x00, /* ----- */
599:      0x00, /* ----- */
600:      0x00, /* ----- */
601:      0x62, /* -##---#- */
602:      0x66, /* -##-##- */
603:      0x0c, /* ----##- */

```

drivers/video/font-lat9-8x14.c

Page 10/62

```

604:      0x18,    /* ----##---- */
605:      0x30,    /* --##----- */
606:      0x66,    /* -##--##- */
607:      0xc6,    /* ##---##- */
608:      0x00,    /* ----- */
609:      0x00,    /* ----- */
610:      0x00,    /* ----- */
611:
612:          /* 0x26 ('&') */
613:      0x00,    /* ----- */
614:      0x00,    /* ----- */
615:      0x38,    /* --###--- */
616:      0x6c,    /* -##-##- */
617:      0x38,    /* --###--- */
618:      0x38,    /* --###--- */
619:      0x76,    /* -###-##- */
620:      0xf6,    /* ####-##- */
621:      0xce,    /* ##--###- */
622:      0xcc,    /* ##--##- */
623:      0x76,    /* -###-##- */
624:      0x00,    /* ----- */
625:      0x00,    /* ----- */
626:      0x00,    /* ----- */
627:
628:          /* 0x27 ('''') */
629:      0x00,    /* ----- */
630:      0x1c,    /* ---###-- */
631:      0x1c,    /* ---###-- */
632:      0x0c,    /* ----##-- */
633:      0x18,    /* ---##--- */
634:      0x00,    /* ----- */
635:      0x00,    /* ----- */
636:      0x00,    /* ----- */
637:      0x00,    /* ----- */
638:      0x00,    /* ----- */
639:      0x00,    /* ----- */
640:      0x00,    /* ----- */
641:      0x00,    /* ----- */
642:      0x00,    /* ----- */
643:
644:          /* 0x28 ('(') */
645:      0x00,    /* ----- */
646:      0x00,    /* ----- */
647:      0x0c,    /* ----##-- */
648:      0x18,    /* ---##--- */
649:      0x30,    /* --##----- */
650:      0x30,    /* --##----- */
651:      0x30,    /* --##----- */
652:      0x30,    /* --##----- */
653:      0x30,    /* --##----- */
654:      0x18,    /* ---##--- */
655:      0x0c,    /* ----##-- */
656:      0x00,    /* ----- */
657:      0x00,    /* ----- */
658:      0x00,    /* ----- */
659:
660:          /* 0x29 (')') */
661:      0x00,    /* ----- */
662:      0x00,    /* ----- */
663:      0x30,    /* --##----- */
664:      0x18,    /* ---##--- */
665:      0x0c,    /* ----##-- */
666:      0x0c,    /* ----##-- */
667:      0x0c,    /* ----##-- */
668:      0x0c,    /* ----##-- */
669:      0x0c,    /* ----##-- */
670:      0x18,    /* ---##--- */

```

drivers/video/font-lat9-8x14.c

Page 11/62

```

671:      0x30, /* --##----- */
672:      0x00, /* ----- */
673:      0x00, /* ----- */
674:      0x00, /* ----- */
675:
676:          /* 0x2a ('**') */
677:      0x00, /* ----- */
678:      0x00, /* ----- */
679:      0x00, /* ----- */
680:      0x00, /* ----- */
681:      0x6c, /* -##-##-- */
682:      0x38, /* --###--- */
683:      0xfe, /* #####- */
684:      0x38, /* --###--- */
685:      0x6c, /* -##-##-- */
686:      0x00, /* ----- */
687:      0x00, /* ----- */
688:      0x00, /* ----- */
689:      0x00, /* ----- */
690:      0x00, /* ----- */
691:
692:          /* 0x2b ('+') */
693:      0x00, /* ----- */
694:      0x00, /* ----- */
695:      0x00, /* ----- */
696:      0x00, /* ----- */
697:      0x18, /* ---##--- */
698:      0x18, /* ---##--- */
699:      0x7e, /* -#####- */
700:      0x18, /* ---##--- */
701:      0x18, /* ---##--- */
702:      0x00, /* ----- */
703:      0x00, /* ----- */
704:      0x00, /* ----- */
705:      0x00, /* ----- */
706:      0x00, /* ----- */
707:
708:          /* 0x2c (',') */
709:      0x00, /* ----- */
710:      0x00, /* ----- */
711:      0x00, /* ----- */
712:      0x00, /* ----- */
713:      0x00, /* ----- */
714:      0x00, /* ----- */
715:      0x00, /* ----- */
716:      0x00, /* ----- */
717:      0x0c, /* ----##-- */
718:      0x0c, /* ----##-- */
719:      0x0c, /* ----##-- */
720:      0x18, /* ---##--- */
721:      0x00, /* ----- */
722:      0x00, /* ----- */
723:
724:          /* 0x2d ('-') */
725:      0x00, /* ----- */
726:      0x00, /* ----- */
727:      0x00, /* ----- */
728:      0x00, /* ----- */
729:      0x00, /* ----- */
730:      0x00, /* ----- */
731:      0xfe, /* #####- */
732:      0x00, /* ----- */
733:      0x00, /* ----- */
734:      0x00, /* ----- */
735:      0x00, /* ----- */
736:      0x00, /* ----- */
737:      0x00, /* ----- */

```

drivers/video/font-lat9-8x14.c

Page 12/62

```

738:      0x00,    /* ----- */
739:
740:      /* 0x2e ('.') */
741:      0x00,    /* ----- */
742:      0x00,    /* ----- */
743:      0x00,    /* ----- */
744:      0x00,    /* ----- */
745:      0x00,    /* ----- */
746:      0x00,    /* ----- */
747:      0x00,    /* ----- */
748:      0x00,    /* ----- */
749:      0x00,    /* ----- */
750:      0x18,    /* ---##--- */
751:      0x18,    /* ---##--- */
752:      0x00,    /* ----- */
753:      0x00,    /* ----- */
754:      0x00,    /* ----- */
755:
756:      /* 0x2f ('/') */
757:      0x00,    /* ----- */
758:      0x00,    /* ----- */
759:      0x00,    /* ----- */
760:      0x00,    /* ----- */
761:      0x06,    /* -----##- */
762:      0x0c,    /* -----##- */
763:      0x18,    /* ---##--- */
764:      0x30,    /* --##----- */
765:      0x60,    /* -##----- */
766:      0xc0,    /* ##----- */
767:      0x00,    /* ----- */
768:      0x00,    /* ----- */
769:      0x00,    /* ----- */
770:      0x00,    /* ----- */
771:
772:      /* 0x30 ('0') */
773:      0x00,    /* ----- */
774:      0x00,    /* ----- */
775:      0x7c,    /* -#####- */
776:      0xc6,    /* ##---##- */
777:      0xc6,    /* ##---##- */
778:      0xc6,    /* ##---##- */
779:      0xd6,    /* ##-##-##- */
780:      0xc6,    /* ##---##- */
781:      0xc6,    /* ##---##- */
782:      0xc6,    /* ##---##- */
783:      0x7c,    /* -#####- */
784:      0x00,    /* ----- */
785:      0x00,    /* ----- */
786:      0x00,    /* ----- */
787:
788:      /* 0x31 ('1') */
789:      0x00,    /* ----- */
790:      0x00,    /* ----- */
791:      0x18,    /* ---##--- */
792:      0x78,    /* -#####- */
793:      0x18,    /* ---##--- */
794:      0x18,    /* ---##--- */
795:      0x18,    /* ---##--- */
796:      0x18,    /* ---##--- */
797:      0x18,    /* ---##--- */
798:      0x18,    /* ---##--- */
799:      0x7e,    /* -#####- */
800:      0x00,    /* ----- */
801:      0x00,    /* ----- */
802:      0x00,    /* ----- */
803:
804:      /* 0x32 ('2') */

```

drivers/video/font-lat9-8x14.c

Page 13/62

```

805:      0x00, /* ----- */
806:      0x00, /* ----- */
807:      0x7c, /* -#####- */
808:      0xc6, /* ##----##- */
809:      0xc6, /* ##----##- */
810:      0x0c, /* ----##-- */
811:      0x18, /* ---##--- */
812:      0x30, /* --##---- */
813:      0x60, /* -##----- */
814:      0xc6, /* ##----##- */
815:      0xfe, /* #####-- */
816:      0x00, /* ----- */
817:      0x00, /* ----- */
818:      0x00, /* ----- */
819:
820:          /* 0x33 ('3') */
821:      0x00, /* ----- */
822:      0x00, /* ----- */
823:      0x7c, /* -#####- */
824:      0xc6, /* ##----##- */
825:      0x06, /* -----##- */
826:      0x06, /* -----##- */
827:      0x3c, /* --#####- */
828:      0x06, /* -----##- */
829:      0x06, /* -----##- */
830:      0xc6, /* ##----##- */
831:      0x7c, /* -#####- */
832:      0x00, /* ----- */
833:      0x00, /* ----- */
834:      0x00, /* ----- */
835:
836:          /* 0x34 ('4') */
837:      0x00, /* ----- */
838:      0x00, /* ----- */
839:      0x0c, /* ----##-- */
840:      0x1c, /* ---###-- */
841:      0x3c, /* --#####- */
842:      0x6c, /* -##-##-- */
843:      0xc6, /* ##----##- */
844:      0xfe, /* #####-- */
845:      0x0c, /* ----##-- */
846:      0x0c, /* ----##-- */
847:      0x0c, /* ----##-- */
848:      0x00, /* ----- */
849:      0x00, /* ----- */
850:      0x00, /* ----- */
851:
852:          /* 0x35 ('5') */
853:      0x00, /* ----- */
854:      0x00, /* ----- */
855:      0xfe, /* #####-- */
856:      0xc0, /* ##----- */
857:      0xc0, /* ##----- */
858:      0xc0, /* ##----- */
859:      0xfc, /* #####-- */
860:      0x06, /* -----##- */
861:      0x06, /* -----##- */
862:      0xc6, /* ##----##- */
863:      0x7c, /* -#####- */
864:      0x00, /* ----- */
865:      0x00, /* ----- */
866:      0x00, /* ----- */
867:
868:          /* 0x36 ('6') */
869:      0x00, /* ----- */
870:      0x00, /* ----- */
871:      0x7c, /* -#####- */

```

drivers/video/font-lat9-8x14.c

Page 14/62

```

872:      0xc6, /* ##---##- */
873:      0xc0, /* ##----- */
874:      0xc0, /* ##----- */
875:      0xfc, /* #####--- */
876:      0xc6, /* ##---##- */
877:      0xc6, /* ##---##- */
878:      0xc6, /* ##---##- */
879:      0x7c, /* -#####-- */
880:      0x00, /* ----- */
881:      0x00, /* ----- */
882:      0x00, /* ----- */
883:
884:          /* 0x37 ('7') */
885:      0x00, /* ----- */
886:      0x00, /* ----- */
887:      0xfe, /* #####--- */
888:      0xc6, /* ##---##- */
889:      0x0c, /* ----##-- */
890:      0x18, /* ----##-- */
891:      0x30, /* --##---- */
892:      0x30, /* --##---- */
893:      0x30, /* --##---- */
894:      0x30, /* --##---- */
895:      0x30, /* --##---- */
896:      0x00, /* ----- */
897:      0x00, /* ----- */
898:      0x00, /* ----- */
899:
900:          /* 0x38 ('8') */
901:      0x00, /* ----- */
902:      0x00, /* ----- */
903:      0x7c, /* -#####-- */
904:      0xc6, /* ##---##- */
905:      0xc6, /* ##---##- */
906:      0xc6, /* ##---##- */
907:      0x7c, /* -#####-- */
908:      0xc6, /* ##---##- */
909:      0xc6, /* ##---##- */
910:      0xc6, /* ##---##- */
911:      0x7c, /* -#####-- */
912:      0x00, /* ----- */
913:      0x00, /* ----- */
914:      0x00, /* ----- */
915:
916:          /* 0x39 ('9') */
917:      0x00, /* ----- */
918:      0x00, /* ----- */
919:      0x7c, /* -#####-- */
920:      0xc6, /* ##---##- */
921:      0xc6, /* ##---##- */
922:      0xc6, /* ##---##- */
923:      0x7e, /* -#####-- */
924:      0x06, /* ----##-- */
925:      0x06, /* ----##-- */
926:      0xc6, /* ##---##- */
927:      0x7c, /* -#####-- */
928:      0x00, /* ----- */
929:      0x00, /* ----- */
930:      0x00, /* ----- */
931:
932:          /* 0x3a (':') */
933:      0x00, /* ----- */
934:      0x00, /* ----- */
935:      0x00, /* ----- */
936:      0x00, /* ----- */
937:      0x0c, /* ----##-- */
938:      0x0c, /* ----##-- */

```

drivers/video/font-lat9-8x14.c Page 15/62

```

939:      0x00, /* ----- */
940:      0x00, /* ----- */
941:      0x0c, /* ----##-- */
942:      0x0c, /* ----##-- */
943:      0x00, /* ----- */
944:      0x00, /* ----- */
945:      0x00, /* ----- */
946:      0x00, /* ----- */
947:
948:          /* 0x3b (';') */
949:      0x00, /* ----- */
950:      0x00, /* ----- */
951:      0x00, /* ----- */
952:      0x00, /* ----- */
953:      0x0c, /* ----##-- */
954:      0x0c, /* ----##-- */
955:      0x00, /* ----- */
956:      0x00, /* ----- */
957:      0x0c, /* ----##-- */
958:      0x0c, /* ----##-- */
959:      0x0c, /* ----##-- */
960:      0x18, /* ---##--- */
961:      0x00, /* ----- */
962:      0x00, /* ----- */
963:
964:          /* 0x3c ('<') */
965:      0x00, /* ----- */
966:      0x00, /* ----- */
967:      0x0c, /* ----##-- */
968:      0x18, /* ---##--- */
969:      0x30, /* --##---- */
970:      0x60, /* -##----- */
971:      0xc0, /* ##----- */
972:      0x60, /* -##----- */
973:      0x30, /* --##---- */
974:      0x18, /* ---##--- */
975:      0x0c, /* ----##-- */
976:      0x00, /* ----- */
977:      0x00, /* ----- */
978:      0x00, /* ----- */
979:
980:          /* 0x3d ('=') */
981:      0x00, /* ----- */
982:      0x00, /* ----- */
983:      0x00, /* ----- */
984:      0x00, /* ----- */
985:      0x00, /* ----- */
986:      0xfe, /* #####--- */
987:      0x00, /* ----- */
988:      0xfe, /* #####--- */
989:      0x00, /* ----- */
990:      0x00, /* ----- */
991:      0x00, /* ----- */
992:      0x00, /* ----- */
993:      0x00, /* ----- */
994:      0x00, /* ----- */
995:
996:          /* 0x3e ('>') */
997:      0x00, /* ----- */
998:      0x00, /* ----- */
999:      0x60, /* -##----- */
1000:      0x30, /* --##---- */
1001:      0x18, /* ---##--- */
1002:      0x0c, /* ----##-- */
1003:      0x06, /* -----##- */
1004:      0x0c, /* ----##-- */
1005:      0x18, /* ---##--- */

```

drivers/video/font-lat9-8x14.c

Page 16/62

```

1006:      0x30,    /* --##----- */
1007:      0x60,    /* -##----- */
1008:      0x00,    /* ----- */
1009:      0x00,    /* ----- */
1010:      0x00,    /* ----- */
1011:
1012:          /* 0x3f ('?') */
1013:      0x00,    /* ----- */
1014:      0x00,    /* ----- */
1015:      0x7c,    /* -#####-- */
1016:      0xc6,    /* ##---##- */
1017:      0xc6,    /* ##---##- */
1018:      0x0c,    /* ----##-- */
1019:      0x18,    /* ----##-- */
1020:      0x18,    /* ----##-- */
1021:      0x00,    /* ----- */
1022:      0x18,    /* ----##-- */
1023:      0x18,    /* ----##-- */
1024:      0x00,    /* ----- */
1025:      0x00,    /* ----- */
1026:      0x00,    /* ----- */
1027:
1028:          /* 0x40 ('@') */
1029:      0x00,    /* ----- */
1030:      0x00,    /* ----- */
1031:      0x7c,    /* -#####-- */
1032:      0xc6,    /* ##---##- */
1033:      0xc6,    /* ##---##- */
1034:      0xde,    /* ##-####- */
1035:      0xde,    /* ##-####- */
1036:      0xde,    /* ##-####- */
1037:      0xdc,    /* ##-####- */
1038:      0xc0,    /* ##----- */
1039:      0x7e,    /* -#####- */
1040:      0x00,    /* ----- */
1041:      0x00,    /* ----- */
1042:      0x00,    /* ----- */
1043:
1044:          /* 0x41 ('A') */
1045:      0x00,    /* ----- */
1046:      0x00,    /* ----- */
1047:      0x38,    /* --###--- */
1048:      0x6c,    /* -##-##-- */
1049:      0xc6,    /* ##---##- */
1050:      0xc6,    /* ##---##- */
1051:      0xc6,    /* ##---##- */
1052:      0xfe,    /* #####--- */
1053:      0xc6,    /* ##---##- */
1054:      0xc6,    /* ##---##- */
1055:      0xc6,    /* ##---##- */
1056:      0x00,    /* ----- */
1057:      0x00,    /* ----- */
1058:      0x00,    /* ----- */
1059:
1060:          /* 0x42 ('B') */
1061:      0x00,    /* ----- */
1062:      0x00,    /* ----- */
1063:      0xfc,    /* #####--- */
1064:      0x66,    /* -##-##-- */
1065:      0x66,    /* -##-##-- */
1066:      0x66,    /* -##-##-- */
1067:      0x7c,    /* -#####-- */
1068:      0x66,    /* -##-##-- */
1069:      0x66,    /* -##-##-- */
1070:      0x66,    /* -##-##-- */
1071:      0xfc,    /* #####--- */
1072:      0x00,    /* ----- */

```


drivers/video/font-lat9-8x14.c

Page 17/62

```

1073:      0x00, /* ----- */
1074:      0x00, /* ----- */
1075:
1076:      /* 0x43 ('C') */
1077:      0x00, /* ----- */
1078:      0x00, /* ----- */
1079:      0x3c, /* ---###-- */
1080:      0x66, /* -##---##- */
1081:      0xc0, /* ##----- */
1082:      0xc0, /* ##----- */
1083:      0xc0, /* ##----- */
1084:      0xc0, /* ##----- */
1085:      0xc0, /* ##----- */
1086:      0x66, /* -##---##- */
1087:      0x3c, /* ---###-- */
1088:      0x00, /* ----- */
1089:      0x00, /* ----- */
1090:      0x00, /* ----- */
1091:
1092:      /* 0x44 ('D') */
1093:      0x00, /* ----- */
1094:      0x00, /* ----- */
1095:      0xf8, /* #####--- */
1096:      0x6c, /* -##-##-- */
1097:      0x66, /* -##---##- */
1098:      0x66, /* -##---##- */
1099:      0x66, /* -##---##- */
1100:      0x66, /* -##---##- */
1101:      0x66, /* -##---##- */
1102:      0x6c, /* -##-##-- */
1103:      0xf8, /* #####--- */
1104:      0x00, /* ----- */
1105:      0x00, /* ----- */
1106:      0x00, /* ----- */
1107:
1108:      /* 0x45 ('E') */
1109:      0x00, /* ----- */
1110:      0x00, /* ----- */
1111:      0xfe, /* #####--- */
1112:      0x66, /* -##---##- */
1113:      0x60, /* -##----- */
1114:      0x60, /* -##----- */
1115:      0x7c, /* -#####-- */
1116:      0x60, /* -##----- */
1117:      0x60, /* -##----- */
1118:      0x66, /* -##---##- */
1119:      0xfe, /* #####--- */
1120:      0x00, /* ----- */
1121:      0x00, /* ----- */
1122:      0x00, /* ----- */
1123:
1124:      /* 0x46 ('F') */
1125:      0x00, /* ----- */
1126:      0x00, /* ----- */
1127:      0xfe, /* #####--- */
1128:      0x66, /* -##---##- */
1129:      0x60, /* -##----- */
1130:      0x60, /* -##----- */
1131:      0x7c, /* -#####-- */
1132:      0x60, /* -##----- */
1133:      0x60, /* -##----- */
1134:      0x60, /* -##----- */
1135:      0xf0, /* #####--- */
1136:      0x00, /* ----- */
1137:      0x00, /* ----- */
1138:      0x00, /* ----- */
1139:

```

drivers/video/font-lat9-8x14.c

Page 18/62

```

1140:                /* 0x47 ('G') */
1141:    0x00,          /* ----- */
1142:    0x00,          /* ----- */
1143:    0x7c,          /* -#####- */
1144:    0xc6,          /* ##----##- */
1145:    0xc6,          /* ##----##- */
1146:    0xc0,          /* ##----- */
1147:    0xc0,          /* ##----- */
1148:    0xce,          /* ##--##-- */
1149:    0xc6,          /* ##----##- */
1150:    0xc6,          /* ##----##- */
1151:    0x7c,          /* -#####- */
1152:    0x00,          /* ----- */
1153:    0x00,          /* ----- */
1154:    0x00,          /* ----- */
1155:
1156:                /* 0x48 ('H') */
1157:    0x00,          /* ----- */
1158:    0x00,          /* ----- */
1159:    0xc6,          /* ##----##- */
1160:    0xc6,          /* ##----##- */
1161:    0xc6,          /* ##----##- */
1162:    0xc6,          /* ##----##- */
1163:    0xfe,          /* #####-- */
1164:    0xc6,          /* ##----##- */
1165:    0xc6,          /* ##----##- */
1166:    0xc6,          /* ##----##- */
1167:    0xc6,          /* ##----##- */
1168:    0x00,          /* ----- */
1169:    0x00,          /* ----- */
1170:    0x00,          /* ----- */
1171:
1172:                /* 0x49 ('I') */
1173:    0x00,          /* ----- */
1174:    0x00,          /* ----- */
1175:    0x3c,          /* --#####- */
1176:    0x18,          /* ---##--- */
1177:    0x18,          /* ---##--- */
1178:    0x18,          /* ---##--- */
1179:    0x18,          /* ---##--- */
1180:    0x18,          /* ---##--- */
1181:    0x18,          /* ---##--- */
1182:    0x18,          /* ---##--- */
1183:    0x3c,          /* --#####- */
1184:    0x00,          /* ----- */
1185:    0x00,          /* ----- */
1186:    0x00,          /* ----- */
1187:
1188:                /* 0x4a ('J') */
1189:    0x00,          /* ----- */
1190:    0x00,          /* ----- */
1191:    0x3c,          /* --#####- */
1192:    0x18,          /* ---##--- */
1193:    0x18,          /* ---##--- */
1194:    0x18,          /* ---##--- */
1195:    0x18,          /* ---##--- */
1196:    0x18,          /* ---##--- */
1197:    0xd8,          /* ##-##--- */
1198:    0xd8,          /* ##-##--- */
1199:    0x70,          /* -###----- */
1200:    0x00,          /* ----- */
1201:    0x00,          /* ----- */
1202:    0x00,          /* ----- */
1203:
1204:                /* 0x4b ('K') */
1205:    0x00,          /* ----- */
1206:    0x00,          /* ----- */

```

drivers/video/font-lat9-8x14.c

Page 19/62

```

1207:      0xc6,    /* ##---##- */
1208:      0xcc,    /* ##---##- */
1209:      0xd8,    /* ##-##--- */
1210:      0xf0,    /* ####---- */
1211:      0xf0,    /* ####---- */
1212:      0xd8,    /* ##-##--- */
1213:      0xcc,    /* ##---##- */
1214:      0xc6,    /* ##---##- */
1215:      0xc6,    /* ##---##- */
1216:      0x00,    /* ----- */
1217:      0x00,    /* ----- */
1218:      0x00,    /* ----- */
1219:
1220:          /* 0x4c ('L') */
1221:      0x00,    /* ----- */
1222:      0x00,    /* ----- */
1223:      0xf0,    /* ####---- */
1224:      0x60,    /* -##----- */
1225:      0x60,    /* -##----- */
1226:      0x60,    /* -##----- */
1227:      0x60,    /* -##----- */
1228:      0x60,    /* -##----- */
1229:      0x62,    /* -##---#- */
1230:      0x66,    /* -##-##- */
1231:      0xfe,    /* #####--- */
1232:      0x00,    /* ----- */
1233:      0x00,    /* ----- */
1234:      0x00,    /* ----- */
1235:
1236:          /* 0x4d ('M') */
1237:      0x00,    /* ----- */
1238:      0x00,    /* ----- */
1239:      0xc6,    /* ##---##- */
1240:      0xc6,    /* ##---##- */
1241:      0xee,    /* ###-##-#- */
1242:      0xfe,    /* #####--- */
1243:      0xd6,    /* ##-##-##- */
1244:      0xd6,    /* ##-##-##- */
1245:      0xd6,    /* ##-##-##- */
1246:      0xc6,    /* ##---##- */
1247:      0xc6,    /* ##---##- */
1248:      0x00,    /* ----- */
1249:      0x00,    /* ----- */
1250:      0x00,    /* ----- */
1251:
1252:          /* 0x4e ('N') */
1253:      0x00,    /* ----- */
1254:      0x00,    /* ----- */
1255:      0xc6,    /* ##---##- */
1256:      0xc6,    /* ##---##- */
1257:      0xe6,    /* ###-##-#- */
1258:      0xe6,    /* ###-##-#- */
1259:      0xf6,    /* ####-##- */
1260:      0xde,    /* ##-####- */
1261:      0xce,    /* ##---##- */
1262:      0xce,    /* ##---##- */
1263:      0xc6,    /* ##---##- */
1264:      0x00,    /* ----- */
1265:      0x00,    /* ----- */
1266:      0x00,    /* ----- */
1267:
1268:          /* 0x4f ('O') */
1269:      0x00,    /* ----- */
1270:      0x00,    /* ----- */
1271:      0x7c,    /* -#####- */
1272:      0xc6,    /* ##---##- */
1273:      0xc6,    /* ##---##- */

```

drivers/video/font-lat9-8x14.c

Page 20/62

```

1274:      0xc6, /* ##---##- */
1275:      0xc6, /* ##---##- */
1276:      0xc6, /* ##---##- */
1277:      0xc6, /* ##---##- */
1278:      0xc6, /* ##---##- */
1279:      0x7c, /* -#####- */
1280:      0x00, /* ----- */
1281:      0x00, /* ----- */
1282:      0x00, /* ----- */
1283:
1284:          /* 0x50 ('P') */
1285:      0x00, /* ----- */
1286:      0x00, /* ----- */
1287:      0xfc, /* #####-- */
1288:      0x66, /* -##--##- */
1289:      0x66, /* -##--##- */
1290:      0x66, /* -##--##- */
1291:      0x7c, /* -#####- */
1292:      0x60, /* -##----- */
1293:      0x60, /* -##----- */
1294:      0x60, /* -##----- */
1295:      0xf0, /* #####---- */
1296:      0x00, /* ----- */
1297:      0x00, /* ----- */
1298:      0x00, /* ----- */
1299:
1300:          /* 0x51 ('Q') */
1301:      0x00, /* ----- */
1302:      0x00, /* ----- */
1303:      0x7c, /* -#####- */
1304:      0xc6, /* ##---##- */
1305:      0xc6, /* ##---##- */
1306:      0xc6, /* ##---##- */
1307:      0xc6, /* ##---##- */
1308:      0xc6, /* ##---##- */
1309:      0xc6, /* ##---##- */
1310:      0xd6, /* ##-##-##- */
1311:      0x7c, /* -#####- */
1312:      0x06, /* -----##- */
1313:      0x00, /* ----- */
1314:      0x00, /* ----- */
1315:
1316:          /* 0x52 ('R') */
1317:      0x00, /* ----- */
1318:      0x00, /* ----- */
1319:      0xfc, /* #####-- */
1320:      0x66, /* -##--##- */
1321:      0x66, /* -##--##- */
1322:      0x66, /* -##--##- */
1323:      0x7c, /* -#####- */
1324:      0x78, /* -#####- */
1325:      0x6c, /* -##-##-- */
1326:      0x66, /* -##--##- */
1327:      0xe6, /* #####--##- */
1328:      0x00, /* ----- */
1329:      0x00, /* ----- */
1330:      0x00, /* ----- */
1331:
1332:          /* 0x53 ('S') */
1333:      0x00, /* ----- */
1334:      0x00, /* ----- */
1335:      0x7c, /* -#####- */
1336:      0xc6, /* ##---##- */
1337:      0xc0, /* ##----- */
1338:      0x60, /* -##----- */
1339:      0x38, /* --###---- */
1340:      0x0c, /* ----##-- */

```

drivers/video/font-lat9-8x14.c

Page 21/62

```

1341:      0x06, /* -----##- */
1342:      0xc6, /* ##----##- */
1343:      0x7c, /* -#####- */
1344:      0x00, /* ----- */
1345:      0x00, /* ----- */
1346:      0x00, /* ----- */
1347:
1348:          /* 0x54 ('T') */
1349:      0x00, /* ----- */
1350:      0x00, /* ----- */
1351:      0x7e, /* -#####- */
1352:      0x5a, /* -#-##-##- */
1353:      0x18, /* ---##--- */
1354:      0x18, /* ---##--- */
1355:      0x18, /* ---##--- */
1356:      0x18, /* ---##--- */
1357:      0x18, /* ---##--- */
1358:      0x18, /* ---##--- */
1359:      0x3c, /* --####- */
1360:      0x00, /* ----- */
1361:      0x00, /* ----- */
1362:      0x00, /* ----- */
1363:
1364:          /* 0x55 ('U') */
1365:      0x00, /* ----- */
1366:      0x00, /* ----- */
1367:      0xc6, /* ##----##- */
1368:      0xc6, /* ##----##- */
1369:      0xc6, /* ##----##- */
1370:      0xc6, /* ##----##- */
1371:      0xc6, /* ##----##- */
1372:      0xc6, /* ##----##- */
1373:      0xc6, /* ##----##- */
1374:      0xc6, /* ##----##- */
1375:      0x7c, /* -#####- */
1376:      0x00, /* ----- */
1377:      0x00, /* ----- */
1378:      0x00, /* ----- */
1379:
1380:          /* 0x56 ('V') */
1381:      0x00, /* ----- */
1382:      0x00, /* ----- */
1383:      0xc6, /* ##----##- */
1384:      0xc6, /* ##----##- */
1385:      0xc6, /* ##----##- */
1386:      0xc6, /* ##----##- */
1387:      0xc6, /* ##----##- */
1388:      0xc6, /* ##----##- */
1389:      0x6c, /* -##-##- */
1390:      0x38, /* --###--- */
1391:      0x10, /* ---#---- */
1392:      0x00, /* ----- */
1393:      0x00, /* ----- */
1394:      0x00, /* ----- */
1395:
1396:          /* 0x57 ('W') */
1397:      0x00, /* ----- */
1398:      0x00, /* ----- */
1399:      0xc6, /* ##----##- */
1400:      0xc6, /* ##----##- */
1401:      0xd6, /* ##-##-##- */
1402:      0xd6, /* ##-##-##- */
1403:      0xd6, /* ##-##-##- */
1404:      0xfe, /* #####-#- */
1405:      0xee, /* ###-###- */
1406:      0xc6, /* ##----##- */
1407:      0xc6, /* ##----##- */

```

drivers/video/font-lat9-8x14.c

Page 22/62

```

1408:      0x00, /* ----- */
1409:      0x00, /* ----- */
1410:      0x00, /* ----- */
1411:
1412:          /* 0x58 ('X') */
1413:      0x00, /* ----- */
1414:      0x00, /* ----- */
1415:      0xc6, /* ##---##- */
1416:      0xc6, /* ##---##- */
1417:      0x6c, /* -##-##-  */
1418:      0x38, /* --###---  */
1419:      0x38, /* --###---  */
1420:      0x38, /* --###---  */
1421:      0x6c, /* -##-##-  */
1422:      0xc6, /* ##---##- */
1423:      0xc6, /* ##---##- */
1424:      0x00, /* ----- */
1425:      0x00, /* ----- */
1426:      0x00, /* ----- */
1427:
1428:          /* 0x59 ('Y') */
1429:      0x00, /* ----- */
1430:      0x00, /* ----- */
1431:      0x66, /* -##--##- */
1432:      0x66, /* -##--##- */
1433:      0x66, /* -##--##- */
1434:      0x66, /* -##--##- */
1435:      0x3c, /* ---####-- */
1436:      0x18, /* ---##---  */
1437:      0x18, /* ---##---  */
1438:      0x18, /* ---##---  */
1439:      0x3c, /* ---####-- */
1440:      0x00, /* ----- */
1441:      0x00, /* ----- */
1442:      0x00, /* ----- */
1443:
1444:          /* 0x5a ('Z') */
1445:      0x00, /* ----- */
1446:      0x00, /* ----- */
1447:      0xfe, /* #####--- */
1448:      0xc6, /* ##---##- */
1449:      0x8c, /* #---##-  */
1450:      0x18, /* ---##---  */
1451:      0x30, /* --##----- */
1452:      0x60, /* -##----- */
1453:      0xc2, /* ##-----# */
1454:      0xc6, /* ##---##- */
1455:      0xfe, /* #####--- */
1456:      0x00, /* ----- */
1457:      0x00, /* ----- */
1458:      0x00, /* ----- */
1459:
1460:          /* 0x5b ('[') */
1461:      0x00, /* ----- */
1462:      0x00, /* ----- */
1463:      0x7c, /* -#####-- */
1464:      0x60, /* -##----- */
1465:      0x60, /* -##----- */
1466:      0x60, /* -##----- */
1467:      0x60, /* -##----- */
1468:      0x60, /* -##----- */
1469:      0x60, /* -##----- */
1470:      0x60, /* -##----- */
1471:      0x7c, /* -#####-- */
1472:      0x00, /* ----- */
1473:      0x00, /* ----- */
1474:      0x00, /* ----- */

```

drivers/video/font-lat9-8x14.c

Page 23/62

```

1475:
1476:          /* 0x5c ('\') */
1477:    0x00,  /* ----- */
1478:    0x00,  /* ----- */
1479:    0x00,  /* ----- */
1480:    0x00,  /* ----- */
1481:    0xc0,  /* ##----- */
1482:    0x60,  /* -##----- */
1483:    0x30,  /* --##----- */
1484:    0x18,  /* ---##----- */
1485:    0x0c,  /* ----##----- */
1486:    0x06,  /* -----##- */
1487:    0x00,  /* ----- */
1488:    0x00,  /* ----- */
1489:    0x00,  /* ----- */
1490:    0x00,  /* ----- */
1491:
1492:          /* 0x5d (']') */
1493:    0x00,  /* ----- */
1494:    0x00,  /* ----- */
1495:    0x7c,  /* -#####-- */
1496:    0x0c,  /* ----##----- */
1497:    0x0c,  /* ----##----- */
1498:    0x0c,  /* ----##----- */
1499:    0x0c,  /* ----##----- */
1500:    0x0c,  /* ----##----- */
1501:    0x0c,  /* ----##----- */
1502:    0x0c,  /* ----##----- */
1503:    0x7c,  /* -#####-- */
1504:    0x00,  /* ----- */
1505:    0x00,  /* ----- */
1506:    0x00,  /* ----- */
1507:
1508:          /* 0x5e ('^') */
1509:    0x00,  /* ----- */
1510:    0x00,  /* ----- */
1511:    0x18,  /* ---##----- */
1512:    0x3c,  /* --#####-- */
1513:    0x66,  /* -##--##- */
1514:    0x00,  /* ----- */
1515:    0x00,  /* ----- */
1516:    0x00,  /* ----- */
1517:    0x00,  /* ----- */
1518:    0x00,  /* ----- */
1519:    0x00,  /* ----- */
1520:    0x00,  /* ----- */
1521:    0x00,  /* ----- */
1522:    0x00,  /* ----- */
1523:
1524:          /* 0x5f ('_') */
1525:    0x00,  /* ----- */
1526:    0x00,  /* ----- */
1527:    0x00,  /* ----- */
1528:    0x00,  /* ----- */
1529:    0x00,  /* ----- */
1530:    0x00,  /* ----- */
1531:    0x00,  /* ----- */
1532:    0x00,  /* ----- */
1533:    0x00,  /* ----- */
1534:    0x00,  /* ----- */
1535:    0x00,  /* ----- */
1536:    0x00,  /* ----- */
1537:    0x00,  /* ----- */
1538:    0xff,  /* ##### */
1539:
1540:          /* 0x60 ('`') */
1541:    0x00,  /* ----- */

```

drivers/video/font-lat9-8x14.c

Page 24/62

```

1542:      0x1c, /* ----###-- */
1543:      0x1c, /* ----###-- */
1544:      0x18, /* ----##--- */
1545:      0x0c, /* -----##-- */
1546:      0x00, /* -----* */
1547:      0x00, /* -----* */
1548:      0x00, /* -----* */
1549:      0x00, /* -----* */
1550:      0x00, /* -----* */
1551:      0x00, /* -----* */
1552:      0x00, /* -----* */
1553:      0x00, /* -----* */
1554:      0x00, /* -----* */
1555:
1556:          /* 0x61 ('a') */
1557:      0x00, /* -----* */
1558:      0x00, /* -----* */
1559:      0x00, /* -----* */
1560:      0x00, /* -----* */
1561:      0x00, /* -----* */
1562:      0x78, /* -####--- */
1563:      0x0c, /* ----##-- */
1564:      0x7c, /* -#####-- */
1565:      0xcc, /* ##---##-- */
1566:      0xdc, /* #-###--- */
1567:      0x76, /* -###-##- */
1568:      0x00, /* -----* */
1569:      0x00, /* -----* */
1570:      0x00, /* -----* */
1571:
1572:          /* 0x62 ('b') */
1573:      0x00, /* -----* */
1574:      0x00, /* -----* */
1575:      0xe0, /* ###----- */
1576:      0x60, /* -##----- */
1577:      0x60, /* -##----- */
1578:      0x7c, /* -#####-- */
1579:      0x66, /* -##-##-#- */
1580:      0x66, /* -##-##-#- */
1581:      0x66, /* -##-##-#- */
1582:      0x66, /* -##-##-#- */
1583:      0xfc, /* #####--- */
1584:      0x00, /* -----* */
1585:      0x00, /* -----* */
1586:      0x00, /* -----* */
1587:
1588:          /* 0x63 ('c') */
1589:      0x00, /* -----* */
1590:      0x00, /* -----* */
1591:      0x00, /* -----* */
1592:      0x00, /* -----* */
1593:      0x00, /* -----* */
1594:      0x7c, /* -#####-- */
1595:      0xc6, /* ##---##- */
1596:      0xc0, /* ##----- */
1597:      0xc0, /* ##----- */
1598:      0xc6, /* ##---##- */
1599:      0x7c, /* -#####-- */
1600:      0x00, /* -----* */
1601:      0x00, /* -----* */
1602:      0x00, /* -----* */
1603:
1604:          /* 0x64 ('d') */
1605:      0x00, /* -----* */
1606:      0x00, /* -----* */
1607:      0x1c, /* ----###-- */
1608:      0x0c, /* ----##-- */

```


drivers/video/font-lat9-8x14.c

Page 25/62

```

1609:      0x0c, /* ----##-- */
1610:      0x7c, /* -#####-- */
1611:      0xcc, /* ##--##-- */
1612:      0xcc, /* ##--##-- */
1613:      0xcc, /* ##--##-- */
1614:      0xcc, /* ##--##-- */
1615:      0x7e, /* -#####-- */
1616:      0x00, /* ----- */
1617:      0x00, /* ----- */
1618:      0x00, /* ----- */
1619:
1620:          /* 0x65 ('e') */
1621:      0x00, /* ----- */
1622:      0x00, /* ----- */
1623:      0x00, /* ----- */
1624:      0x00, /* ----- */
1625:      0x00, /* ----- */
1626:      0x7c, /* -#####-- */
1627:      0xc6, /* ##--##-- */
1628:      0xfe, /* #####--- */
1629:      0xc0, /* ##----- */
1630:      0xc6, /* ##--##-- */
1631:      0x7c, /* -#####-- */
1632:      0x00, /* ----- */
1633:      0x00, /* ----- */
1634:      0x00, /* ----- */
1635:
1636:          /* 0x66 ('f') */
1637:      0x00, /* ----- */
1638:      0x00, /* ----- */
1639:      0x1c, /* ---###-- */
1640:      0x36, /* --##-##- */
1641:      0x30, /* --##----- */
1642:      0x30, /* --##----- */
1643:      0xfc, /* #####--- */
1644:      0x30, /* --##----- */
1645:      0x30, /* --##----- */
1646:      0x30, /* --##----- */
1647:      0x78, /* -#####-- */
1648:      0x00, /* ----- */
1649:      0x00, /* ----- */
1650:      0x00, /* ----- */
1651:
1652:          /* 0x67 ('g') */
1653:      0x00, /* ----- */
1654:      0x00, /* ----- */
1655:      0x00, /* ----- */
1656:      0x00, /* ----- */
1657:      0x00, /* ----- */
1658:      0x76, /* -###-##- */
1659:      0xce, /* ##--##-- */
1660:      0xc6, /* ##--##-- */
1661:      0xc6, /* ##--##-- */
1662:      0x7e, /* -#####-- */
1663:      0x06, /* -----##- */
1664:      0xc6, /* ##--##-- */
1665:      0x7c, /* -#####-- */
1666:      0x00, /* ----- */
1667:
1668:          /* 0x68 ('h') */
1669:      0x00, /* ----- */
1670:      0x00, /* ----- */
1671:      0xe0, /* ###----- */
1672:      0x60, /* -##----- */
1673:      0x60, /* -##----- */
1674:      0x6c, /* -##-##-- */
1675:      0x76, /* -###-##- */

```

drivers/video/font-lat9-8x14.c

Page 26/62

```

1676:      0x66,    /* ---##---##- */
1677:      0x66,    /* ---##---##- */
1678:      0x66,    /* ---##---##- */
1679:      0xe6,    /* ###---##- */
1680:      0x00,    /* ----- */
1681:      0x00,    /* ----- */
1682:      0x00,    /* ----- */
1683:
1684:          /* 0x69 ('i') */
1685:      0x00,    /* ----- */
1686:      0x00,    /* ----- */
1687:      0x18,    /* ---##--- */
1688:      0x18,    /* ---##--- */
1689:      0x00,    /* ----- */
1690:      0x38,    /* ---###--- */
1691:      0x18,    /* ---##--- */
1692:      0x18,    /* ---##--- */
1693:      0x18,    /* ---##--- */
1694:      0x18,    /* ---##--- */
1695:      0x3c,    /* ---####--- */
1696:      0x00,    /* ----- */
1697:      0x00,    /* ----- */
1698:      0x00,    /* ----- */
1699:
1700:          /* 0x6a ('j') */
1701:      0x00,    /* ----- */
1702:      0x00,    /* ----- */
1703:      0x0c,    /* ---##--- */
1704:      0x0c,    /* ---##--- */
1705:      0x00,    /* ----- */
1706:      0x1c,    /* ---###--- */
1707:      0x0c,    /* ---##--- */
1708:      0x0c,    /* ---##--- */
1709:      0x0c,    /* ---##--- */
1710:      0x0c,    /* ---##--- */
1711:      0xcc,    /* ##---##--- */
1712:      0xcc,    /* ##---##--- */
1713:      0x78,    /* -####--- */
1714:      0x00,    /* ----- */
1715:
1716:          /* 0x6b ('k') */
1717:      0x00,    /* ----- */
1718:      0x00,    /* ----- */
1719:      0xe0,    /* ###----- */
1720:      0x60,    /* -##----- */
1721:      0x60,    /* -##----- */
1722:      0x66,    /* -##---##- */
1723:      0x6c,    /* -##---##- */
1724:      0x78,    /* -####--- */
1725:      0x6c,    /* -##---##- */
1726:      0x66,    /* -##---##- */
1727:      0xe6,    /* ###---##- */
1728:      0x00,    /* ----- */
1729:      0x00,    /* ----- */
1730:      0x00,    /* ----- */
1731:
1732:          /* 0x6c ('l') */
1733:      0x00,    /* ----- */
1734:      0x00,    /* ----- */
1735:      0x38,    /* ---###--- */
1736:      0x18,    /* ---##--- */
1737:      0x18,    /* ---##--- */
1738:      0x18,    /* ---##--- */
1739:      0x18,    /* ---##--- */
1740:      0x18,    /* ---##--- */
1741:      0x18,    /* ---##--- */
1742:      0x18,    /* ---##--- */

```

drivers/video/font-lat9-8x14.c

Page 27/62

```

1743:      0x3c, /* ---###-- */
1744:      0x00, /* ----- */
1745:      0x00, /* ----- */
1746:      0x00, /* ----- */
1747:
1748:          /* 0x6d ('m') */
1749:      0x00, /* ----- */
1750:      0x00, /* ----- */
1751:      0x00, /* ----- */
1752:      0x00, /* ----- */
1753:      0x00, /* ----- */
1754:      0x6c, /* -##-##-- */
1755:      0xfe, /* #####- */
1756:      0xd6, /* ##-##-#- */
1757:      0xd6, /* ##-##-#- */
1758:      0xc6, /* ##---##- */
1759:      0xc6, /* ##---##- */
1760:      0x00, /* ----- */
1761:      0x00, /* ----- */
1762:      0x00, /* ----- */
1763:
1764:          /* 0x6e ('n') */
1765:      0x00, /* ----- */
1766:      0x00, /* ----- */
1767:      0x00, /* ----- */
1768:      0x00, /* ----- */
1769:      0x00, /* ----- */
1770:      0xdc, /* ##-###-- */
1771:      0x66, /* -##-##-#- */
1772:      0x66, /* -##-##-#- */
1773:      0x66, /* -##-##-#- */
1774:      0x66, /* -##-##-#- */
1775:      0x66, /* -##-##-#- */
1776:      0x00, /* ----- */
1777:      0x00, /* ----- */
1778:      0x00, /* ----- */
1779:
1780:          /* 0x6f ('o') */
1781:      0x00, /* ----- */
1782:      0x00, /* ----- */
1783:      0x00, /* ----- */
1784:      0x00, /* ----- */
1785:      0x00, /* ----- */
1786:      0x7c, /* -#####- */
1787:      0xc6, /* ##---##- */
1788:      0xc6, /* ##---##- */
1789:      0xc6, /* ##---##- */
1790:      0xc6, /* ##---##- */
1791:      0x7c, /* -#####- */
1792:      0x00, /* ----- */
1793:      0x00, /* ----- */
1794:      0x00, /* ----- */
1795:
1796:          /* 0x70 ('p') */
1797:      0x00, /* ----- */
1798:      0x00, /* ----- */
1799:      0x00, /* ----- */
1800:      0x00, /* ----- */
1801:      0x00, /* ----- */
1802:      0xdc, /* ##-###-- */
1803:      0x66, /* -##-##-#- */
1804:      0x66, /* -##-##-#- */
1805:      0x66, /* -##-##-#- */
1806:      0x7c, /* -#####- */
1807:      0x60, /* -##----- */
1808:      0x60, /* -##----- */
1809:      0xf0, /* #####----- */

```

drivers/video/font-lat9-8x14.c

Page 28/62

```

1810:      0x00, /* ----- */
1811:
1812:      /* 0x71 ('q') */
1813:      0x00, /* ----- */
1814:      0x00, /* ----- */
1815:      0x00, /* ----- */
1816:      0x00, /* ----- */
1817:      0x00, /* ----- */
1818:      0x76, /* -###-##- */
1819:      0xcc, /* ##--##-- */
1820:      0xcc, /* ##--##-- */
1821:      0xcc, /* ##--##-- */
1822:      0x7c, /* -#####-- */
1823:      0x0c, /* ----##-- */
1824:      0x0c, /* ----##-- */
1825:      0x1e, /* ---####- */
1826:      0x00, /* ----- */
1827:
1828:      /* 0x72 ('r') */
1829:      0x00, /* ----- */
1830:      0x00, /* ----- */
1831:      0x00, /* ----- */
1832:      0x00, /* ----- */
1833:      0x00, /* ----- */
1834:      0xdc, /* #-###-- */
1835:      0x66, /* -##--##- */
1836:      0x60, /* -##----- */
1837:      0x60, /* -##----- */
1838:      0x60, /* -##----- */
1839:      0xf0, /* #####----- */
1840:      0x00, /* ----- */
1841:      0x00, /* ----- */
1842:      0x00, /* ----- */
1843:
1844:      /* 0x73 ('s') */
1845:      0x00, /* ----- */
1846:      0x00, /* ----- */
1847:      0x00, /* ----- */
1848:      0x00, /* ----- */
1849:      0x00, /* ----- */
1850:      0x7c, /* -#####-- */
1851:      0xc6, /* ##---##- */
1852:      0x70, /* -###----- */
1853:      0x1c, /* ---###-- */
1854:      0xc6, /* ##---##- */
1855:      0x7c, /* -#####-- */
1856:      0x00, /* ----- */
1857:      0x00, /* ----- */
1858:      0x00, /* ----- */
1859:
1860:      /* 0x74 ('t') */
1861:      0x00, /* ----- */
1862:      0x00, /* ----- */
1863:      0x30, /* --##----- */
1864:      0x30, /* --##----- */
1865:      0x30, /* --##----- */
1866:      0xfc, /* #####----- */
1867:      0x30, /* --##----- */
1868:      0x30, /* --##----- */
1869:      0x30, /* --##----- */
1870:      0x36, /* ---##-##- */
1871:      0x1c, /* ---###-- */
1872:      0x00, /* ----- */
1873:      0x00, /* ----- */
1874:      0x00, /* ----- */
1875:
1876:      /* 0x75 ('u') */

```

drivers/video/font-lat9-8x14.c

Page 29/62

```

1877:      0x00, /* ----- */
1878:      0x00, /* ----- */
1879:      0x00, /* ----- */
1880:      0x00, /* ----- */
1881:      0x00, /* ----- */
1882:      0xcc, /* ##---##- */
1883:      0xcc, /* ##---##- */
1884:      0xcc, /* ##---##- */
1885:      0xcc, /* ##---##- */
1886:      0xcc, /* ##---##- */
1887:      0x76, /* -###-##- */
1888:      0x00, /* ----- */
1889:      0x00, /* ----- */
1890:      0x00, /* ----- */
1891:
1892:          /* 0x76 ('v') */
1893:      0x00, /* ----- */
1894:      0x00, /* ----- */
1895:      0x00, /* ----- */
1896:      0x00, /* ----- */
1897:      0x00, /* ----- */
1898:      0xc6, /* ##---##- */
1899:      0xc6, /* ##---##- */
1900:      0xc6, /* ##---##- */
1901:      0x6c, /* -###-##- */
1902:      0x38, /* --###--- */
1903:      0x10, /* ---#---- */
1904:      0x00, /* ----- */
1905:      0x00, /* ----- */
1906:      0x00, /* ----- */
1907:
1908:          /* 0x77 ('w') */
1909:      0x00, /* ----- */
1910:      0x00, /* ----- */
1911:      0x00, /* ----- */
1912:      0x00, /* ----- */
1913:      0x00, /* ----- */
1914:      0xc6, /* ##---##- */
1915:      0xc6, /* ##---##- */
1916:      0xd6, /* ##-##-##- */
1917:      0xd6, /* ##-##-##- */
1918:      0xfe, /* #####-#- */
1919:      0x6c, /* -###-##- */
1920:      0x00, /* ----- */
1921:      0x00, /* ----- */
1922:      0x00, /* ----- */
1923:
1924:          /* 0x78 ('x') */
1925:      0x00, /* ----- */
1926:      0x00, /* ----- */
1927:      0x00, /* ----- */
1928:      0x00, /* ----- */
1929:      0x00, /* ----- */
1930:      0xc6, /* ##---##- */
1931:      0x6c, /* -###-##- */
1932:      0x38, /* --###--- */
1933:      0x38, /* --###--- */
1934:      0x6c, /* -###-##- */
1935:      0xc6, /* ##---##- */
1936:      0x00, /* ----- */
1937:      0x00, /* ----- */
1938:      0x00, /* ----- */
1939:
1940:          /* 0x79 ('y') */
1941:      0x00, /* ----- */
1942:      0x00, /* ----- */
1943:      0x00, /* ----- */

```

drivers/video/font-lat9-8x14.c

Page 30/62

```

1944:      0x00, /* ----- */
1945:      0x00, /* ----- */
1946:      0xc6, /* ##---##- */
1947:      0xc6, /* ##---##- */
1948:      0xc6, /* ##---##- */
1949:      0xce, /* ##---##- */
1950:      0x76, /* -###-##- */
1951:      0x06, /* -----##- */
1952:      0xc6, /* ##---##- */
1953:      0x7c, /* -#####- */
1954:      0x00, /* ----- */
1955:
1956:          /* 0x7a ('z') */
1957:      0x00, /* ----- */
1958:      0x00, /* ----- */
1959:      0x00, /* ----- */
1960:      0x00, /* ----- */
1961:      0x00, /* ----- */
1962:      0xfe, /* #####- */
1963:      0x8c, /* #---##- */
1964:      0x18, /* ---##- */
1965:      0x30, /* --##- */
1966:      0x62, /* -##-#- */
1967:      0xfe, /* #####- */
1968:      0x00, /* ----- */
1969:      0x00, /* ----- */
1970:      0x00, /* ----- */
1971:
1972:          /* 0x7b ('{') */
1973:      0x00, /* ----- */
1974:      0x00, /* ----- */
1975:      0x0e, /* ----###- */
1976:      0x18, /* ---##- */
1977:      0x18, /* ---##- */
1978:      0x18, /* ---##- */
1979:      0x70, /* -###- */
1980:      0x18, /* ---##- */
1981:      0x18, /* ---##- */
1982:      0x18, /* ---##- */
1983:      0x0e, /* ----###- */
1984:      0x00, /* ----- */
1985:      0x00, /* ----- */
1986:      0x00, /* ----- */
1987:
1988:          /* 0x7c ('|') */
1989:      0x00, /* ----- */
1990:      0x00, /* ----- */
1991:      0x18, /* ---##- */
1992:      0x18, /* ---##- */
1993:      0x18, /* ---##- */
1994:      0x18, /* ---##- */
1995:      0x18, /* ---##- */
1996:      0x18, /* ---##- */
1997:      0x18, /* ---##- */
1998:      0x18, /* ---##- */
1999:      0x18, /* ---##- */
2000:      0x00, /* ----- */
2001:      0x00, /* ----- */
2002:      0x00, /* ----- */
2003:
2004:          /* 0x7d ('}') */
2005:      0x00, /* ----- */
2006:      0x00, /* ----- */
2007:      0x70, /* -###- */
2008:      0x18, /* ---##- */
2009:      0x18, /* ---##- */
2010:      0x18, /* ---##- */

```

drivers/video/font-lat9-8x14.c

Page 31/62

```

2011:      0x0e,    /* ----###- */
2012:      0x18,    /* ---##---- */
2013:      0x18,    /* ---##---- */
2014:      0x18,    /* ---##---- */
2015:      0x70,    /* -###----- */
2016:      0x00,    /* ----- */
2017:      0x00,    /* ----- */
2018:      0x00,    /* ----- */
2019:
2020:      /* 0x7e ('~') */
2021:      0x00,    /* ----- */
2022:      0x00,    /* ----- */
2023:      0x76,    /* -###-##- */
2024:      0xdc,    /* ##-###-- */
2025:      0x00,    /* ----- */
2026:      0x00,    /* ----- */
2027:      0x00,    /* ----- */
2028:      0x00,    /* ----- */
2029:      0x00,    /* ----- */
2030:      0x00,    /* ----- */
2031:      0x00,    /* ----- */
2032:      0x00,    /* ----- */
2033:      0x00,    /* ----- */
2034:      0x00,    /* ----- */
2035:
2036:      /* 0x7f */
2037:      0x00,    /* ----- */
2038:      0x66,    /* -##--##- */
2039:      0x66,    /* -##--##- */
2040:      0x00,    /* ----- */
2041:      0x66,    /* -##--##- */
2042:      0x66,    /* -##--##- */
2043:      0x66,    /* -##--##- */
2044:      0x3c,    /* --#####- */
2045:      0x18,    /* ---##---- */
2046:      0x18,    /* ---##---- */
2047:      0x3c,    /* --#####- */
2048:      0x00,    /* ----- */
2049:      0x00,    /* ----- */
2050:      0x00,    /* ----- */
2051:
2052:      /* 0x80 */
2053:      0x00,    /* ----- */
2054:      0x00,    /* ----- */
2055:      0x00,    /* ----- */
2056:      0xff,    /* ######## */
2057:      0x00,    /* ----- */
2058:      0x00,    /* ----- */
2059:      0x00,    /* ----- */
2060:      0x00,    /* ----- */
2061:      0x00,    /* ----- */
2062:      0x00,    /* ----- */
2063:      0x00,    /* ----- */
2064:      0x00,    /* ----- */
2065:      0x00,    /* ----- */
2066:      0x00,    /* ----- */
2067:
2068:      /* 0x81 */
2069:      0x18,    /* ---##---- */
2070:      0x18,    /* ---##---- */
2071:      0x18,    /* ---##---- */
2072:      0x18,    /* ---##---- */
2073:      0x18,    /* ---##---- */
2074:      0x18,    /* ---##---- */
2075:      0x18,    /* ---##---- */
2076:      0x00,    /* ----- */
2077:      0x00,    /* ----- */

```

drivers/video/font-lat9-8x14.c

Page 32/62

```

2078:      0x00, /* ----- */
2079:      0x00, /* ----- */
2080:      0x00, /* ----- */
2081:      0x00, /* ----- */
2082:      0x00, /* ----- */
2083:
2084:          /* 0x82 */
2085:      0x00, /* ----- */
2086:      0x00, /* ----- */
2087:      0x00, /* ----- */
2088:      0x00, /* ----- */
2089:      0x00, /* ----- */
2090:      0x00, /* ----- */
2091:      0x1f, /* ----##### */
2092:      0x00, /* ----- */
2093:      0x00, /* ----- */
2094:      0x00, /* ----- */
2095:      0x00, /* ----- */
2096:      0x00, /* ----- */
2097:      0x00, /* ----- */
2098:      0x00, /* ----- */
2099:
2100:          /* 0x83 */
2101:      0x18, /* ---##--- */
2102:      0x18, /* ---##--- */
2103:      0x18, /* ---##--- */
2104:      0x18, /* ---##--- */
2105:      0x18, /* ---##--- */
2106:      0x18, /* ---##--- */
2107:      0x1f, /* ----##### */
2108:      0x00, /* ----- */
2109:      0x00, /* ----- */
2110:      0x00, /* ----- */
2111:      0x00, /* ----- */
2112:      0x00, /* ----- */
2113:      0x00, /* ----- */
2114:      0x00, /* ----- */
2115:
2116:          /* 0x84 */
2117:      0x00, /* ----- */
2118:      0x00, /* ----- */
2119:      0x00, /* ----- */
2120:      0x00, /* ----- */
2121:      0x00, /* ----- */
2122:      0x00, /* ----- */
2123:      0x18, /* ---##--- */
2124:      0x18, /* ---##--- */
2125:      0x18, /* ---##--- */
2126:      0x18, /* ---##--- */
2127:      0x18, /* ---##--- */
2128:      0x18, /* ---##--- */
2129:      0x18, /* ---##--- */
2130:      0x18, /* ---##--- */
2131:
2132:          /* 0x85 */
2133:      0x18, /* ---##--- */
2134:      0x18, /* ---##--- */
2135:      0x18, /* ---##--- */
2136:      0x18, /* ---##--- */
2137:      0x18, /* ---##--- */
2138:      0x18, /* ---##--- */
2139:      0x18, /* ---##--- */
2140:      0x18, /* ---##--- */
2141:      0x18, /* ---##--- */
2142:      0x18, /* ---##--- */
2143:      0x18, /* ---##--- */
2144:      0x18, /* ---##--- */

```


drivers/video/font-lat9-8x14.c

Page 33/62

```

2145:      0x18, /* ---##--- */
2146:      0x18, /* ---##--- */
2147:
2148:          /* 0x86 */
2149:      0x00, /* ----- */
2150:      0x00, /* ----- */
2151:      0x00, /* ----- */
2152:      0x00, /* ----- */
2153:      0x00, /* ----- */
2154:      0x00, /* ----- */
2155:      0x1f, /* ----##### */
2156:      0x18, /* ---##--- */
2157:      0x18, /* ---##--- */
2158:      0x18, /* ---##--- */
2159:      0x18, /* ---##--- */
2160:      0x18, /* ---##--- */
2161:      0x18, /* ---##--- */
2162:      0x18, /* ---##--- */
2163:
2164:          /* 0x87 */
2165:      0x18, /* ---##--- */
2166:      0x18, /* ---##--- */
2167:      0x18, /* ---##--- */
2168:      0x18, /* ---##--- */
2169:      0x18, /* ---##--- */
2170:      0x18, /* ---##--- */
2171:      0x1f, /* ----##### */
2172:      0x18, /* ---##--- */
2173:      0x18, /* ---##--- */
2174:      0x18, /* ---##--- */
2175:      0x18, /* ---##--- */
2176:      0x18, /* ---##--- */
2177:      0x18, /* ---##--- */
2178:      0x18, /* ---##--- */
2179:
2180:          /* 0x88 */
2181:      0x00, /* ----- */
2182:      0x00, /* ----- */
2183:      0x00, /* ----- */
2184:      0x00, /* ----- */
2185:      0x00, /* ----- */
2186:      0x00, /* ----- */
2187:      0xf8, /* #####--- */
2188:      0x00, /* ----- */
2189:      0x00, /* ----- */
2190:      0x00, /* ----- */
2191:      0x00, /* ----- */
2192:      0x00, /* ----- */
2193:      0x00, /* ----- */
2194:      0x00, /* ----- */
2195:
2196:          /* 0x89 */
2197:      0x18, /* ---##--- */
2198:      0x18, /* ---##--- */
2199:      0x18, /* ---##--- */
2200:      0x18, /* ---##--- */
2201:      0x18, /* ---##--- */
2202:      0x18, /* ---##--- */
2203:      0xf8, /* #####--- */
2204:      0x00, /* ----- */
2205:      0x00, /* ----- */
2206:      0x00, /* ----- */
2207:      0x00, /* ----- */
2208:      0x00, /* ----- */
2209:      0x00, /* ----- */
2210:      0x00, /* ----- */
2211:

```

drivers/video/font-lat9-8x14.c

Page 34/62

```

2212:                /* 0x8a          */
2213:                0x00, /* ----- */
2214:                0x00, /* ----- */
2215:                0x00, /* ----- */
2216:                0x00, /* ----- */
2217:                0x00, /* ----- */
2218:                0x00, /* ----- */
2219:                0xff, /* ##### */
2220:                0x00, /* ----- */
2221:                0x00, /* ----- */
2222:                0x00, /* ----- */
2223:                0x00, /* ----- */
2224:                0x00, /* ----- */
2225:                0x00, /* ----- */
2226:                0x00, /* ----- */
2227:
2228:                /* 0x8b          */
2229:                0x18, /* ---##--- */
2230:                0x18, /* ---##--- */
2231:                0x18, /* ---##--- */
2232:                0x18, /* ---##--- */
2233:                0x18, /* ---##--- */
2234:                0x18, /* ---##--- */
2235:                0xff, /* ##### */
2236:                0x00, /* ----- */
2237:                0x00, /* ----- */
2238:                0x00, /* ----- */
2239:                0x00, /* ----- */
2240:                0x00, /* ----- */
2241:                0x00, /* ----- */
2242:                0x00, /* ----- */
2243:
2244:                /* 0x8c          */
2245:                0x00, /* ----- */
2246:                0x00, /* ----- */
2247:                0x00, /* ----- */
2248:                0x00, /* ----- */
2249:                0x00, /* ----- */
2250:                0x00, /* ----- */
2251:                0xf8, /* #####--- */
2252:                0x18, /* ---##--- */
2253:                0x18, /* ---##--- */
2254:                0x18, /* ---##--- */
2255:                0x18, /* ---##--- */
2256:                0x18, /* ---##--- */
2257:                0x18, /* ---##--- */
2258:                0x18, /* ---##--- */
2259:
2260:                /* 0x8d          */
2261:                0x18, /* ---##--- */
2262:                0x18, /* ---##--- */
2263:                0x18, /* ---##--- */
2264:                0x18, /* ---##--- */
2265:                0x18, /* ---##--- */
2266:                0x18, /* ---##--- */
2267:                0xf8, /* #####--- */
2268:                0x18, /* ---##--- */
2269:                0x18, /* ---##--- */
2270:                0x18, /* ---##--- */
2271:                0x18, /* ---##--- */
2272:                0x18, /* ---##--- */
2273:                0x18, /* ---##--- */
2274:                0x18, /* ---##--- */
2275:
2276:                /* 0x8e          */
2277:                0x00, /* ----- */
2278:                0x00, /* ----- */

```

drivers/video/font-lat9-8x14.c

Page 35/62

```

2279:      0x00, /* ----- */
2280:      0x00, /* ----- */
2281:      0x00, /* ----- */
2282:      0x00, /* ----- */
2283:      0xff, /* ##### */
2284:      0x18, /* ---#--- */
2285:      0x18, /* ---#--- */
2286:      0x18, /* ---#--- */
2287:      0x18, /* ---#--- */
2288:      0x18, /* ---#--- */
2289:      0x18, /* ---#--- */
2290:      0x18, /* ---#--- */
2291:
2292:          /* 0x8f */
2293:      0x18, /* ---#--- */
2294:      0x18, /* ---#--- */
2295:      0x18, /* ---#--- */
2296:      0x18, /* ---#--- */
2297:      0x18, /* ---#--- */
2298:      0x18, /* ---#--- */
2299:      0xff, /* ##### */
2300:      0x18, /* ---#--- */
2301:      0x18, /* ---#--- */
2302:      0x18, /* ---#--- */
2303:      0x18, /* ---#--- */
2304:      0x18, /* ---#--- */
2305:      0x18, /* ---#--- */
2306:      0x18, /* ---#--- */
2307:
2308:          /* 0x90 */
2309:      0x00, /* ----- */
2310:      0x00, /* ----- */
2311:      0x00, /* ----- */
2312:      0x00, /* ----- */
2313:      0x00, /* ----- */
2314:      0x00, /* ----- */
2315:      0x00, /* ----- */
2316:      0x00, /* ----- */
2317:      0x00, /* ----- */
2318:      0xff, /* ##### */
2319:      0x00, /* ----- */
2320:      0x00, /* ----- */
2321:      0x00, /* ----- */
2322:      0x00, /* ----- */
2323:
2324:          /* 0x91 */
2325:      0x6c, /* -##-##- */
2326:      0x6c, /* -##-##- */
2327:      0x6c, /* -##-##- */
2328:      0x6c, /* -##-##- */
2329:      0x6c, /* -##-##- */
2330:      0x6c, /* -##-##- */
2331:      0x6c, /* -##-##- */
2332:      0x7c, /* -#####- */
2333:      0x00, /* ----- */
2334:      0x00, /* ----- */
2335:      0x00, /* ----- */
2336:      0x00, /* ----- */
2337:      0x00, /* ----- */
2338:      0x00, /* ----- */
2339:
2340:          /* 0x92 */
2341:      0x00, /* ----- */
2342:      0x00, /* ----- */
2343:      0x00, /* ----- */
2344:      0x00, /* ----- */
2345:      0x00, /* ----- */

```

drivers/video/font-lat9-8x14.c

Page 36/62

```

2346:      0x3f, /* ----### * /
2347:      0x30, /* ---##----- * /
2348:      0x3f, /* ----### * /
2349:      0x00, /* ----- * /
2350:      0x00, /* ----- * /
2351:      0x00, /* ----- * /
2352:      0x00, /* ----- * /
2353:      0x00, /* ----- * /
2354:      0x00, /* ----- * /
2355:
2356:          /* 0x93 * /
2357:      0x6c, /* -##-##-- * /
2358:      0x6c, /* -##-##-- * /
2359:      0x6c, /* -##-##-- * /
2360:      0x6c, /* -##-##-- * /
2361:      0x6c, /* -##-##-- * /
2362:      0x6f, /* -##-#### * /
2363:      0x60, /* -##----- * /
2364:      0x7f, /* -##### * /
2365:      0x00, /* ----- * /
2366:      0x00, /* ----- * /
2367:      0x00, /* ----- * /
2368:      0x00, /* ----- * /
2369:      0x00, /* ----- * /
2370:      0x00, /* ----- * /
2371:
2372:          /* 0x94 * /
2373:      0x00, /* ----- * /
2374:      0x00, /* ----- * /
2375:      0x00, /* ----- * /
2376:      0x00, /* ----- * /
2377:      0x00, /* ----- * /
2378:      0x7c, /* -#####-- * /
2379:      0x6c, /* -##-##-- * /
2380:      0x6c, /* -##-##-- * /
2381:      0x6c, /* -##-##-- * /
2382:      0x6c, /* -##-##-- * /
2383:      0x6c, /* -##-##-- * /
2384:      0x6c, /* -##-##-- * /
2385:      0x6c, /* -##-##-- * /
2386:      0x6c, /* -##-##-- * /
2387:
2388:          /* 0x95 * /
2389:      0x6c, /* -##-##-- * /
2390:      0x6c, /* -##-##-- * /
2391:      0x6c, /* -##-##-- * /
2392:      0x6c, /* -##-##-- * /
2393:      0x6c, /* -##-##-- * /
2394:      0x6c, /* -##-##-- * /
2395:      0x6c, /* -##-##-- * /
2396:      0x6c, /* -##-##-- * /
2397:      0x6c, /* -##-##-- * /
2398:      0x6c, /* -##-##-- * /
2399:      0x6c, /* -##-##-- * /
2400:      0x6c, /* -##-##-- * /
2401:      0x6c, /* -##-##-- * /
2402:      0x6c, /* -##-##-- * /
2403:
2404:          /* 0x96 * /
2405:      0x00, /* ----- * /
2406:      0x00, /* ----- * /
2407:      0x00, /* ----- * /
2408:      0x00, /* ----- * /
2409:      0x00, /* ----- * /
2410:      0x7f, /* -##### * /
2411:      0x60, /* -##----- * /
2412:      0x6f, /* -##-#### * /

```

```

drivers/video/font-lat9-8x14.c Page 37/62
2413:      0x6c, /* -##-##-- */
2414:      0x6c, /* -##-##-- */
2415:      0x6c, /* -##-##-- */
2416:      0x6c, /* -##-##-- */
2417:      0x6c, /* -##-##-- */
2418:      0x6c, /* -##-##-- */
2419:
2420:          /* 0x97 */
2421:      0x6c, /* -##-##-- */
2422:      0x6c, /* -##-##-- */
2423:      0x6c, /* -##-##-- */
2424:      0x6c, /* -##-##-- */
2425:      0x6c, /* -##-##-- */
2426:      0x6f, /* -##-#### */
2427:      0x60, /* -##----- */
2428:      0x6f, /* -##-#### */
2429:      0x6c, /* -##-##-- */
2430:      0x6c, /* -##-##-- */
2431:      0x6c, /* -##-##-- */
2432:      0x6c, /* -##-##-- */
2433:      0x6c, /* -##-##-- */
2434:      0x6c, /* -##-##-- */
2435:
2436:          /* 0x98 */
2437:      0x00, /* ----- */
2438:      0x00, /* ----- */
2439:      0x00, /* ----- */
2440:      0x00, /* ----- */
2441:      0x00, /* ----- */
2442:      0xfc, /* #####-- */
2443:      0x0c, /* ----##-- */
2444:      0xfc, /* #####-- */
2445:      0x00, /* ----- */
2446:      0x00, /* ----- */
2447:      0x00, /* ----- */
2448:      0x00, /* ----- */
2449:      0x00, /* ----- */
2450:      0x00, /* ----- */
2451:
2452:          /* 0x99 */
2453:      0x6c, /* -##-##-- */
2454:      0x6c, /* -##-##-- */
2455:      0x6c, /* -##-##-- */
2456:      0x6c, /* -##-##-- */
2457:      0x6c, /* -##-##-- */
2458:      0xec, /* -##-##-- */
2459:      0x0c, /* ----##-- */
2460:      0xfc, /* #####-- */
2461:      0x00, /* ----- */
2462:      0x00, /* ----- */
2463:      0x00, /* ----- */
2464:      0x00, /* ----- */
2465:      0x00, /* ----- */
2466:      0x00, /* ----- */
2467:
2468:          /* 0x9a */
2469:      0x00, /* ----- */
2470:      0x00, /* ----- */
2471:      0x00, /* ----- */
2472:      0x00, /* ----- */
2473:      0x00, /* ----- */
2474:      0xff, /* ##### */
2475:      0x00, /* ----- */
2476:      0xff, /* ##### */
2477:      0x00, /* ----- */
2478:      0x00, /* ----- */
2479:      0x00, /* ----- */

```

drivers/video/font-lat9-8x14.c

Page 38/62

```

2480:      0x00, /* ----- */
2481:      0x00, /* ----- */
2482:      0x00, /* ----- */
2483:
2484:          /* 0x9b */
2485:      0x6c, /* -##-##-- */
2486:      0x6c, /* -##-##-- */
2487:      0x6c, /* -##-##-- */
2488:      0x6c, /* -##-##-- */
2489:      0x6c, /* -##-##-- */
2490:      0xef, /* ###-#### */
2491:      0x00, /* ----- */
2492:      0xff, /* ##### */
2493:      0x00, /* ----- */
2494:      0x00, /* ----- */
2495:      0x00, /* ----- */
2496:      0x00, /* ----- */
2497:      0x00, /* ----- */
2498:      0x00, /* ----- */
2499:
2500:          /* 0x9c */
2501:      0x00, /* ----- */
2502:      0x00, /* ----- */
2503:      0x00, /* ----- */
2504:      0x00, /* ----- */
2505:      0x00, /* ----- */
2506:      0xfc, /* #####-- */
2507:      0x0c, /* ----##-- */
2508:      0xec, /* ###-##-- */
2509:      0x6c, /* -##-##-- */
2510:      0x6c, /* -##-##-- */
2511:      0x6c, /* -##-##-- */
2512:      0x6c, /* -##-##-- */
2513:      0x6c, /* -##-##-- */
2514:      0x6c, /* -##-##-- */
2515:
2516:          /* 0x9d */
2517:      0x6c, /* -##-##-- */
2518:      0x6c, /* -##-##-- */
2519:      0x6c, /* -##-##-- */
2520:      0x6c, /* -##-##-- */
2521:      0x6c, /* -##-##-- */
2522:      0xec, /* ###-##-- */
2523:      0x0c, /* ----##-- */
2524:      0xec, /* ###-##-- */
2525:      0x6c, /* -##-##-- */
2526:      0x6c, /* -##-##-- */
2527:      0x6c, /* -##-##-- */
2528:      0x6c, /* -##-##-- */
2529:      0x6c, /* -##-##-- */
2530:      0x6c, /* -##-##-- */
2531:
2532:          /* 0x9e */
2533:      0x00, /* ----- */
2534:      0x00, /* ----- */
2535:      0x00, /* ----- */
2536:      0x00, /* ----- */
2537:      0x00, /* ----- */
2538:      0xff, /* ##### */
2539:      0x00, /* ----- */
2540:      0xef, /* ###-#### */
2541:      0x6c, /* -##-##-- */
2542:      0x6c, /* -##-##-- */
2543:      0x6c, /* -##-##-- */
2544:      0x6c, /* -##-##-- */
2545:      0x6c, /* -##-##-- */
2546:      0x6c, /* -##-##-- */

```

drivers/video/font-lat9-8x14.c

Page 39/62

```

2547:
2548:          /* 0x9f          */
2549:          0x6c, /* -##-##-- */
2550:          0x6c, /* -##-##-- */
2551:          0x6c, /* -##-##-- */
2552:          0x6c, /* -##-##-- */
2553:          0x6c, /* -##-##-- */
2554:          0xef, /* ###-#### */
2555:          0x00, /* ----- */
2556:          0xef, /* ###-#### */
2557:          0x6c, /* -##-##-- */
2558:          0x6c, /* -##-##-- */
2559:          0x6c, /* -##-##-- */
2560:          0x6c, /* -##-##-- */
2561:          0x6c, /* -##-##-- */
2562:          0x6c, /* -##-##-- */
2563:
2564:          /* 0xa0          */
2565:          0x00, /* ----- */
2566:          0x00, /* ----- */
2567:          0x00, /* ----- */
2568:          0x00, /* ----- */
2569:          0x00, /* ----- */
2570:          0x00, /* ----- */
2571:          0x00, /* ----- */
2572:          0x00, /* ----- */
2573:          0x00, /* ----- */
2574:          0x00, /* ----- */
2575:          0x82, /* #-----#- */
2576:          0xfe, /* #####-#- */
2577:          0x00, /* ----- */
2578:          0x00, /* ----- */
2579:
2580:          /* 0xa1          */
2581:          0x00, /* ----- */
2582:          0x00, /* ----- */
2583:          0x00, /* ----- */
2584:          0x00, /* ----- */
2585:          0x18, /* ---##--- */
2586:          0x18, /* ---##--- */
2587:          0x00, /* ----- */
2588:          0x18, /* ---##--- */
2589:          0x18, /* ---##--- */
2590:          0x3c, /* --####-- */
2591:          0x3c, /* --####-- */
2592:          0x3c, /* --####-- */
2593:          0x18, /* ---##--- */
2594:          0x00, /* ----- */
2595:
2596:          /* 0xa2          */
2597:          0x00, /* ----- */
2598:          0x00, /* ----- */
2599:          0x00, /* ----- */
2600:          0x10, /* ---#---- */
2601:          0x7c, /* -#####-- */
2602:          0xd6, /* ##-##-##- */
2603:          0xd0, /* ##-#---- */
2604:          0xd0, /* ##-#---- */
2605:          0xd6, /* ##-##-##- */
2606:          0x7c, /* -#####-- */
2607:          0x10, /* ---#---- */
2608:          0x00, /* ----- */
2609:          0x00, /* ----- */
2610:          0x00, /* ----- */
2611:
2612:          /* 0xa3          */
2613:          0x00, /* ----- */

```

drivers/video/font-lat9-8x14.c

Page 40/62

```

2614:      0x00,    /* ----- */
2615:      0x38,    /* ---###--- */
2616:      0x6c,    /* -##-##-- */
2617:      0x60,    /* -##----- */
2618:      0x60,    /* -##----- */
2619:      0xf0,    /* #####---- */
2620:      0x60,    /* -##----- */
2621:      0x66,    /* -##--##- */
2622:      0xf6,    /* #####-##- */
2623:      0x6c,    /* -##-##-- */
2624:      0x00,    /* ----- */
2625:      0x00,    /* ----- */
2626:      0x00,    /* ----- */
2627:
2628:          /* 0xa4      */
2629:      0x00,    /* ----- */
2630:      0x00,    /* ----- */
2631:      0x3c,    /* ---####-- */
2632:      0x62,    /* -##---#- */
2633:      0x60,    /* -##----- */
2634:      0xf8,    /* #####---- */
2635:      0x60,    /* -##----- */
2636:      0xf8,    /* #####---- */
2637:      0x60,    /* -##----- */
2638:      0x62,    /* -##---#- */
2639:      0x3c,    /* ---####-- */
2640:      0x00,    /* ----- */
2641:      0x00,    /* ----- */
2642:      0x00,    /* ----- */
2643:
2644:          /* 0xa5      */
2645:      0x00,    /* ----- */
2646:      0x00,    /* ----- */
2647:      0x66,    /* -##--##- */
2648:      0x66,    /* -##--##- */
2649:      0x3c,    /* ---####-- */
2650:      0x18,    /* ---##---- */
2651:      0x7e,    /* -#####- */
2652:      0x18,    /* ---##---- */
2653:      0x3c,    /* ---####-- */
2654:      0x18,    /* ---##---- */
2655:      0x18,    /* ---##---- */
2656:      0x00,    /* ----- */
2657:      0x00,    /* ----- */
2658:      0x00,    /* ----- */
2659:
2660:          /* 0xa6      */
2661:      0x6c,    /* -##-##-- */
2662:      0x38,    /* ---###--- */
2663:      0x00,    /* ----- */
2664:      0x7c,    /* -#####-- */
2665:      0xc6,    /* ##---##- */
2666:      0xc6,    /* ##---##- */
2667:      0x60,    /* -##----- */
2668:      0x1c,    /* ----###-- */
2669:      0x06,    /* -----##- */
2670:      0xc6,    /* ##---##- */
2671:      0x7c,    /* -#####-- */
2672:      0x00,    /* ----- */
2673:      0x00,    /* ----- */
2674:      0x00,    /* ----- */
2675:
2676:          /* 0xa7      */
2677:      0x00,    /* ----- */
2678:      0x7c,    /* -#####-- */
2679:      0xc6,    /* ##---##- */
2680:      0xc6,    /* ##---##- */

```


drivers/video/font-lat9-8x14.c

Page 41/62

```

2681:      0x60,    /* -##----- */
2682:      0x7c,    /* -#####-- */
2683:      0xc6,    /* ##---##- */
2684:      0xc6,    /* ##---##- */
2685:      0x7c,    /* -#####-- */
2686:      0x0c,    /* ----##-- */
2687:      0xc6,    /* ##---##- */
2688:      0xc6,    /* ##---##- */
2689:      0x7c,    /* -#####-- */
2690:      0x00,    /* ----- */
2691:
2692:          /* 0xa8 */
2693:      0x00,    /* ----- */
2694:      0x00,    /* ----- */
2695:      0x00,    /* ----- */
2696:      0x00,    /* ----- */
2697:      0x00,    /* ----- */
2698:      0x00,    /* ----- */
2699:      0x00,    /* ----- */
2700:      0x00,    /* ----- */
2701:      0x00,    /* ----- */
2702:      0x00,    /* ----- */
2703:      0x00,    /* ----- */
2704:      0x00,    /* ----- */
2705:      0x00,    /* ----- */
2706:      0x00,    /* ----- */
2707:
2708:          /* 0xa9 */
2709:      0x00,    /* ----- */
2710:      0x7e,    /* -#####- */
2711:      0x81,    /* #-----# */
2712:      0x99,    /* #-##-##-# */
2713:      0xa5,    /* #-#-##-#-# */
2714:      0xa1,    /* #-#-##-##-# */
2715:      0xa5,    /* #-#-##-##-# */
2716:      0x99,    /* #-#-##-##-# */
2717:      0x81,    /* #-----# */
2718:      0x7e,    /* -#####- */
2719:      0x00,    /* ----- */
2720:      0x00,    /* ----- */
2721:      0x00,    /* ----- */
2722:      0x00,    /* ----- */
2723:
2724:          /* 0xaa */
2725:      0x00,    /* ----- */
2726:      0x3c,    /* --#####- */
2727:      0x6c,    /* -##-##-##- */
2728:      0x6c,    /* -##-##-##- */
2729:      0x3e,    /* --#####- */
2730:      0x00,    /* ----- */
2731:      0x7e,    /* -#####- */
2732:      0x00,    /* ----- */
2733:      0x00,    /* ----- */
2734:      0x00,    /* ----- */
2735:      0x00,    /* ----- */
2736:      0x00,    /* ----- */
2737:      0x00,    /* ----- */
2738:      0x00,    /* ----- */
2739:
2740:          /* 0xab */
2741:      0x00,    /* ----- */
2742:      0x00,    /* ----- */
2743:      0x00,    /* ----- */
2744:      0x00,    /* ----- */
2745:      0x36,    /* --##-##-# */
2746:      0x6c,    /* -##-##-##- */
2747:      0xd8,    /* ##-##-##-# */

```

drivers/video/font-lat9-8x14.c

Page 42/62

```

2748:      0x6c,    /* --##--##-- */
2749:      0x36,    /* ---##-##- */
2750:      0x00,    /* ----- */
2751:      0x00,    /* ----- */
2752:      0x00,    /* ----- */
2753:      0x00,    /* ----- */
2754:      0x00,    /* ----- */
2755:
2756:          /* 0xac */
2757:      0x00,    /* ----- */
2758:      0x00,    /* ----- */
2759:      0x00,    /* ----- */
2760:      0x00,    /* ----- */
2761:      0x00,    /* ----- */
2762:      0x00,    /* ----- */
2763:      0x7e,    /* -#####- */
2764:      0x06,    /* -----##- */
2765:      0x06,    /* -----##- */
2766:      0x06,    /* -----##- */
2767:      0x00,    /* ----- */
2768:      0x00,    /* ----- */
2769:      0x00,    /* ----- */
2770:      0x00,    /* ----- */
2771:
2772:          /* 0xad */
2773:      0x00,    /* ----- */
2774:      0x00,    /* ----- */
2775:      0x00,    /* ----- */
2776:      0x00,    /* ----- */
2777:      0x00,    /* ----- */
2778:      0x00,    /* ----- */
2779:      0x7e,    /* -#####- */
2780:      0x00,    /* ----- */
2781:      0x00,    /* ----- */
2782:      0x00,    /* ----- */
2783:      0x00,    /* ----- */
2784:      0x00,    /* ----- */
2785:      0x00,    /* ----- */
2786:      0x00,    /* ----- */
2787:
2788:          /* 0xae */
2789:      0x00,    /* ----- */
2790:      0x7e,    /* -#####- */
2791:      0x81,    /* #-----# */
2792:      0xb9,    /* #-###-#-# */
2793:      0xa5,    /* #-#--#-#-# */
2794:      0xb9,    /* #-###-#-# */
2795:      0xa5,    /* #-#--#-#-# */
2796:      0xa5,    /* #-#--#-#-# */
2797:      0x81,    /* #-----# */
2798:      0x7e,    /* -#####- */
2799:      0x00,    /* ----- */
2800:      0x00,    /* ----- */
2801:      0x00,    /* ----- */
2802:      0x00,    /* ----- */
2803:
2804:          /* 0xaf */
2805:      0xff,    /* ##### */
2806:      0x00,    /* ----- */
2807:      0x00,    /* ----- */
2808:      0x00,    /* ----- */
2809:      0x00,    /* ----- */
2810:      0x00,    /* ----- */
2811:      0x00,    /* ----- */
2812:      0x00,    /* ----- */
2813:      0x00,    /* ----- */
2814:      0x00,    /* ----- */

```

drivers/video/font-lat9-8x14.c

Page 43/62

```

2815:      0x00, /* ----- */
2816:      0x00, /* ----- */
2817:      0x00, /* ----- */
2818:      0x00, /* ----- */
2819:
2820:          /* 0xb0 */
2821:      0x00, /* ----- */
2822:      0x38, /* ---###--- */
2823:      0x6c, /* -##-##- */
2824:      0x38, /* ---###--- */
2825:      0x00, /* ----- */
2826:      0x00, /* ----- */
2827:      0x00, /* ----- */
2828:      0x00, /* ----- */
2829:      0x00, /* ----- */
2830:      0x00, /* ----- */
2831:      0x00, /* ----- */
2832:      0x00, /* ----- */
2833:      0x00, /* ----- */
2834:      0x00, /* ----- */
2835:
2836:          /* 0xb1 */
2837:      0x00, /* ----- */
2838:      0x00, /* ----- */
2839:      0x00, /* ----- */
2840:      0x00, /* ----- */
2841:      0x18, /* ----##--- */
2842:      0x18, /* ----##--- */
2843:      0x7e, /* -#####- */
2844:      0x18, /* ----##--- */
2845:      0x18, /* ----##--- */
2846:      0x00, /* ----- */
2847:      0x7e, /* -#####- */
2848:      0x00, /* ----- */
2849:      0x00, /* ----- */
2850:      0x00, /* ----- */
2851:
2852:          /* 0xb2 */
2853:      0x00, /* ----- */
2854:      0x38, /* ---###--- */
2855:      0x6c, /* -##-##- */
2856:      0x18, /* ----##--- */
2857:      0x30, /* --##----- */
2858:      0x7c, /* -#####- */
2859:      0x00, /* ----- */
2860:      0x00, /* ----- */
2861:      0x00, /* ----- */
2862:      0x00, /* ----- */
2863:      0x00, /* ----- */
2864:      0x00, /* ----- */
2865:      0x00, /* ----- */
2866:      0x00, /* ----- */
2867:
2868:          /* 0xb3 */
2869:      0x00, /* ----- */
2870:      0x38, /* ---###--- */
2871:      0x6c, /* -##-##- */
2872:      0x18, /* ----##--- */
2873:      0x6c, /* -##-##- */
2874:      0x38, /* ---###--- */
2875:      0x00, /* ----- */
2876:      0x00, /* ----- */
2877:      0x00, /* ----- */
2878:      0x00, /* ----- */
2879:      0x00, /* ----- */
2880:      0x00, /* ----- */
2881:      0x00, /* ----- */

```

drivers/video/font-lat9-8x14.c

Page 44/62

```

2882:      0x00,    /* ----- */
2883:
2884:      /* 0xb4 */
2885:      0x6c,    /* -##-##- */
2886:      0x38,    /* --###-   */
2887:      0x00,    /* ----- */
2888:      0xfe,    /* #####-  */
2889:      0xc6,    /* ##-##-  */
2890:      0x8c,    /* #---#-   */
2891:      0x18,    /* ---##-   */
2892:      0x30,    /* --##-    */
2893:      0x62,    /* -##-##-  */
2894:      0xc6,    /* ##-##-  */
2895:      0xfe,    /* #####-  */
2896:      0x00,    /* ----- */
2897:      0x00,    /* ----- */
2898:      0x00,    /* ----- */
2899:
2900:      /* 0xb5 */
2901:      0x00,    /* ----- */
2902:      0x00,    /* ----- */
2903:      0x00,    /* ----- */
2904:      0x00,    /* ----- */
2905:      0x00,    /* ----- */
2906:      0xcc,    /* ##-##-  */
2907:      0xcc,    /* ##-##-  */
2908:      0xcc,    /* ##-##-  */
2909:      0xcc,    /* ##-##-  */
2910:      0xcc,    /* ##-##-  */
2911:      0xf6,    /* ####-##- */
2912:      0xc0,    /* ##-     */
2913:      0xc0,    /* ##-     */
2914:      0xc0,    /* ##-     */
2915:
2916:      /* 0xb6 */
2917:      0x00,    /* ----- */
2918:      0x00,    /* ----- */
2919:      0x7f,    /* -##### */
2920:      0xdb,    /* ##-##-## */
2921:      0xdb,    /* ##-##-## */
2922:      0xdb,    /* ##-##-## */
2923:      0x7b,    /* -##-##-## */
2924:      0x1b,    /* ---##-## */
2925:      0x1b,    /* ---##-## */
2926:      0x1b,    /* ---##-## */
2927:      0x1b,    /* ---##-## */
2928:      0x00,    /* ----- */
2929:      0x00,    /* ----- */
2930:      0x00,    /* ----- */
2931:
2932:      /* 0xb7 */
2933:      0x00,    /* ----- */
2934:      0x00,    /* ----- */
2935:      0x00,    /* ----- */
2936:      0x00,    /* ----- */
2937:      0x00,    /* ----- */
2938:      0x00,    /* ----- */
2939:      0x18,    /* ---##-   */
2940:      0x18,    /* ---##-   */
2941:      0x00,    /* ----- */
2942:      0x00,    /* ----- */
2943:      0x00,    /* ----- */
2944:      0x00,    /* ----- */
2945:      0x00,    /* ----- */
2946:      0x00,    /* ----- */
2947:
2948:      /* 0xb8 */

```

drivers/video/font-lat9-8x14.c

Page 45/62

```

2949:      0x00, /* ----- */
2950:      0x00, /* ----- */
2951:      0x6c, /* -##-##- */
2952:      0x38, /* ---###--- */
2953:      0x00, /* ----- */
2954:      0xfe, /* #####- */
2955:      0x8c, /* #---##- */
2956:      0x18, /* ----##- */
2957:      0x30, /* --##---- */
2958:      0x62, /* -##---#- */
2959:      0xfe, /* #####- */
2960:      0x00, /* ----- */
2961:      0x00, /* ----- */
2962:      0x00, /* ----- */
2963:
2964:          /* 0xb9 */
2965:      0x00, /* ----- */
2966:      0x30, /* --##---- */
2967:      0x70, /* -##-#-#- */
2968:      0x30, /* --##---- */
2969:      0x30, /* --##---- */
2970:      0x78, /* -#####- */
2971:      0x00, /* ----- */
2972:      0x00, /* ----- */
2973:      0x00, /* ----- */
2974:      0x00, /* ----- */
2975:      0x00, /* ----- */
2976:      0x00, /* ----- */
2977:      0x00, /* ----- */
2978:      0x00, /* ----- */
2979:
2980:          /* 0xba */
2981:      0x00, /* ----- */
2982:      0x38, /* ---###--- */
2983:      0x6c, /* -##-##- */
2984:      0x6c, /* -##-##- */
2985:      0x38, /* ---###--- */
2986:      0x00, /* ----- */
2987:      0x7c, /* -#####- */
2988:      0x00, /* ----- */
2989:      0x00, /* ----- */
2990:      0x00, /* ----- */
2991:      0x00, /* ----- */
2992:      0x00, /* ----- */
2993:      0x00, /* ----- */
2994:      0x00, /* ----- */
2995:
2996:          /* 0xbb */
2997:      0x00, /* ----- */
2998:      0x00, /* ----- */
2999:      0x00, /* ----- */
3000:      0x00, /* ----- */
3001:      0xd8, /* ##-##-#- */
3002:      0x6c, /* -##-##- */
3003:      0x36, /* --##-##- */
3004:      0x6c, /* -##-##- */
3005:      0xd8, /* ##-##-#- */
3006:      0x00, /* ----- */
3007:      0x00, /* ----- */
3008:      0x00, /* ----- */
3009:      0x00, /* ----- */
3010:      0x00, /* ----- */
3011:
3012:          /* 0xbc */
3013:      0x00, /* ----- */
3014:      0x00, /* ----- */
3015:      0x77, /* -###-### */

```

drivers/video/font-lat9-8x14.c

Page 46/62

```

3016:      0xdf, /* ##-#### */
3017:      0xd8, /* ##-##--- */
3018:      0xde, /* ##-####- */
3019:      0xde, /* ##-####- */
3020:      0xd8, /* ##-##--- */
3021:      0xd8, /* ##-##--- */
3022:      0xdf, /* ##-#### */
3023:      0x77, /* -##-### */
3024:      0x00, /* ----- */
3025:      0x00, /* ----- */
3026:      0x00, /* ----- */
3027:
3028:          /* 0xbd */
3029:      0x00, /* ----- */
3030:      0x00, /* ----- */
3031:      0x00, /* ----- */
3032:      0x00, /* ----- */
3033:      0x00, /* ----- */
3034:      0x6e, /* -##-##- */
3035:      0xdb, /* ##-##-## */
3036:      0xdf, /* ##-#### */
3037:      0xd8, /* ##-##--- */
3038:      0xdb, /* ##-##-## */
3039:      0x6e, /* -##-##- */
3040:      0x00, /* ----- */
3041:      0x00, /* ----- */
3042:      0x00, /* ----- */
3043:
3044:          /* 0xbe */
3045:      0x00, /* ----- */
3046:      0x66, /* -##-##- */
3047:      0x66, /* -##-##- */
3048:      0x00, /* ----- */
3049:      0x66, /* -##-##- */
3050:      0x66, /* -##-##- */
3051:      0x3c, /* --###- */
3052:      0x18, /* ---##- */
3053:      0x18, /* ---##- */
3054:      0x18, /* ---##- */
3055:      0x3c, /* --###- */
3056:      0x00, /* ----- */
3057:      0x00, /* ----- */
3058:      0x00, /* ----- */
3059:
3060:          /* 0xbf */
3061:      0x00, /* ----- */
3062:      0x00, /* ----- */
3063:      0x00, /* ----- */
3064:      0x00, /* ----- */
3065:      0x30, /* --##- */
3066:      0x30, /* --##- */
3067:      0x00, /* ----- */
3068:      0x30, /* --##- */
3069:      0x30, /* --##- */
3070:      0x60, /* -##- */
3071:      0xc6, /* ##-##- */
3072:      0xc6, /* ##-##- */
3073:      0x7c, /* -#####- */
3074:      0x00, /* ----- */
3075:
3076:          /* 0xc0 */
3077:      0x60, /* -##- */
3078:      0x30, /* --##- */
3079:      0x18, /* ---##- */
3080:      0x00, /* ----- */
3081:      0x38, /* --###- */
3082:      0x6c, /* -##-##- */

```

drivers/video/font-lat9-8x14.c Page 47/62

```

3083:      0xc6, /* ##---##- */
3084:      0xc6, /* ##---##- */
3085:      0xfe, /* #####- */
3086:      0xc6, /* ##---##- */
3087:      0xc6, /* ##---##- */
3088:      0x00, /* ----- */
3089:      0x00, /* ----- */
3090:      0x00, /* ----- */
3091:
3092:          /* 0xc1 */
3093:      0x0c, /* ----##- */
3094:      0x18, /* ---##--- */
3095:      0x30, /* --##---- */
3096:      0x00, /* ----- */
3097:      0x38, /* --###--- */
3098:      0x6c, /* -##-##- */
3099:      0xc6, /* ##---##- */
3100:      0xc6, /* ##---##- */
3101:      0xfe, /* #####- */
3102:      0xc6, /* ##---##- */
3103:      0xc6, /* ##---##- */
3104:      0x00, /* ----- */
3105:      0x00, /* ----- */
3106:      0x00, /* ----- */
3107:
3108:          /* 0xc2 */
3109:      0x10, /* ---#---- */
3110:      0x38, /* --###--- */
3111:      0x6c, /* -##-##- */
3112:      0x00, /* ----- */
3113:      0x38, /* --##--- */
3114:      0x6c, /* -##-##- */
3115:      0xc6, /* ##---##- */
3116:      0xc6, /* ##---##- */
3117:      0xfe, /* #####- */
3118:      0xc6, /* ##---##- */
3119:      0xc6, /* ##---##- */
3120:      0x00, /* ----- */
3121:      0x00, /* ----- */
3122:      0x00, /* ----- */
3123:
3124:          /* 0xc3 */
3125:      0x00, /* ----- */
3126:      0x76, /* -###-##- */
3127:      0xdc, /* ##-###- */
3128:      0x00, /* ----- */
3129:      0x38, /* --##--- */
3130:      0x6c, /* -##-##- */
3131:      0xc6, /* ##---##- */
3132:      0xc6, /* ##---##- */
3133:      0xfe, /* #####- */
3134:      0xc6, /* ##---##- */
3135:      0xc6, /* ##---##- */
3136:      0x00, /* ----- */
3137:      0x00, /* ----- */
3138:      0x00, /* ----- */
3139:
3140:          /* 0xc4 */
3141:      0x00, /* ----- */
3142:      0x6c, /* -##-##- */
3143:      0x6c, /* -##-##- */
3144:      0x00, /* ----- */
3145:      0x38, /* --##--- */
3146:      0x6c, /* -##-##- */
3147:      0xc6, /* ##---##- */
3148:      0xc6, /* ##---##- */
3149:      0xfe, /* #####- */

```

drivers/video/font-lat9-8x14.c

Page 48/62

```

3150:      0xc6,      /* ##---##- */
3151:      0xc6,      /* ##---##- */
3152:      0x00,      /* ----- */
3153:      0x00,      /* ----- */
3154:      0x00,      /* ----- */
3155:
3156:          /* 0xc5          */
3157:      0x38,      /* ---###--- */
3158:      0x6c,      /* -##-##-- */
3159:      0x38,      /* ---###--- */
3160:      0x00,      /* ----- */
3161:      0x38,      /* ---###--- */
3162:      0x6c,      /* -##-##-- */
3163:      0xc6,      /* ##---##- */
3164:      0xc6,      /* ##---##- */
3165:      0xfe,      /* #####- */
3166:      0xc6,      /* ##---##- */
3167:      0xc6,      /* ##---##- */
3168:      0x00,      /* ----- */
3169:      0x00,      /* ----- */
3170:      0x00,      /* ----- */
3171:
3172:          /* 0xc6          */
3173:      0x00,      /* ----- */
3174:      0x7e,      /* -#####- */
3175:      0xd8,      /* ##-##--- */
3176:      0xd8,      /* ##-##--- */
3177:      0xd8,      /* ##-##--- */
3178:      0xd8,      /* ##-##--- */
3179:      0xfe,      /* #####- */
3180:      0xd8,      /* ##-##--- */
3181:      0xd8,      /* ##-##--- */
3182:      0xd8,      /* ##-##--- */
3183:      0xde,      /* ##-####- */
3184:      0x00,      /* ----- */
3185:      0x00,      /* ----- */
3186:      0x00,      /* ----- */
3187:
3188:          /* 0xc7          */
3189:      0x00,      /* ----- */
3190:      0x00,      /* ----- */
3191:      0x3c,      /* ---###--- */
3192:      0x66,      /* -##-##-- */
3193:      0xc0,      /* ##----- */
3194:      0xc0,      /* ##----- */
3195:      0xc0,      /* ##----- */
3196:      0xc0,      /* ##----- */
3197:      0xc0,      /* ##----- */
3198:      0x66,      /* -##-##-- */
3199:      0x3c,      /* ---###--- */
3200:      0x18,      /* ---##--- */
3201:      0x6c,      /* -##-##-- */
3202:      0x38,      /* ---###--- */
3203:
3204:          /* 0xc8          */
3205:      0x30,      /* ---##--- */
3206:      0x18,      /* ---##--- */
3207:      0x0c,      /* ----##-- */
3208:      0x00,      /* ----- */
3209:      0xfe,      /* #####- */
3210:      0x66,      /* -##-##-- */
3211:      0x60,      /* -##----- */
3212:      0x7c,      /* -#####- */
3213:      0x60,      /* -##----- */
3214:      0x66,      /* -##-##-- */
3215:      0xfe,      /* #####- */
3216:      0x00,      /* ----- */

```


drivers/video/font-lat9-8x14.c

Page 49/62

```

3217:      0x00, /* ----- */
3218:      0x00, /* ----- */
3219:
3220:          /* 0xc9 */
3221:      0x0c, /* ----##-- */
3222:      0x18, /* ---##--- */
3223:      0x30, /* --##---- */
3224:      0x00, /* ----- */
3225:      0xfe, /* #####- */
3226:      0x66, /* -##-##- */
3227:      0x60, /* -##----- */
3228:      0x7c, /* -#####- */
3229:      0x60, /* -##----- */
3230:      0x66, /* -##-##- */
3231:      0xfe, /* #####- */
3232:      0x00, /* ----- */
3233:      0x00, /* ----- */
3234:      0x00, /* ----- */
3235:
3236:          /* 0xca */
3237:      0x10, /* ---#---- */
3238:      0x38, /* --###--- */
3239:      0x6c, /* -##-##- */
3240:      0x00, /* ----- */
3241:      0xfe, /* #####- */
3242:      0x66, /* -##-##- */
3243:      0x60, /* -##----- */
3244:      0x7c, /* -#####- */
3245:      0x60, /* -##----- */
3246:      0x66, /* -##-##- */
3247:      0xfe, /* #####- */
3248:      0x00, /* ----- */
3249:      0x00, /* ----- */
3250:      0x00, /* ----- */
3251:
3252:          /* 0xcb */
3253:      0x00, /* ----- */
3254:      0x6c, /* -##-##- */
3255:      0x6c, /* -##-##- */
3256:      0x00, /* ----- */
3257:      0xfe, /* #####- */
3258:      0x66, /* -##-##- */
3259:      0x60, /* -##----- */
3260:      0x7c, /* -#####- */
3261:      0x60, /* -##----- */
3262:      0x66, /* -##-##- */
3263:      0xfe, /* #####- */
3264:      0x00, /* ----- */
3265:      0x00, /* ----- */
3266:      0x00, /* ----- */
3267:
3268:          /* 0xcc */
3269:      0x30, /* --##---- */
3270:      0x18, /* ---##--- */
3271:      0x0c, /* ----##-- */
3272:      0x00, /* ----- */
3273:      0x3c, /* --####- */
3274:      0x18, /* ---##--- */
3275:      0x18, /* ---##--- */
3276:      0x18, /* ---##--- */
3277:      0x18, /* ---##--- */
3278:      0x18, /* ---##--- */
3279:      0x3c, /* --####- */
3280:      0x00, /* ----- */
3281:      0x00, /* ----- */
3282:      0x00, /* ----- */
3283:

```

drivers/video/font-lat9-8x14.c

Page 50/62

```

3284:                /* 0xcd          */
3285:    0x0c,          /* ----##-- */
3286:    0x18,          /* ---##--- */
3287:    0x30,          /* --##---- */
3288:    0x00,          /* ----- */
3289:    0x3c,          /* --####-- */
3290:    0x18,          /* ---##--- */
3291:    0x18,          /* ---##--- */
3292:    0x18,          /* ---##--- */
3293:    0x18,          /* ---##--- */
3294:    0x18,          /* ---##--- */
3295:    0x3c,          /* --####-- */
3296:    0x00,          /* ----- */
3297:    0x00,          /* ----- */
3298:    0x00,          /* ----- */
3299:
3300:                /* 0xce          */
3301:    0x18,          /* ---##--- */
3302:    0x3c,          /* --####-- */
3303:    0x66,          /* -##-##- */
3304:    0x00,          /* ----- */
3305:    0x3c,          /* --####-- */
3306:    0x18,          /* ---##--- */
3307:    0x18,          /* ---##--- */
3308:    0x18,          /* ---##--- */
3309:    0x18,          /* ---##--- */
3310:    0x18,          /* ---##--- */
3311:    0x3c,          /* --####-- */
3312:    0x00,          /* ----- */
3313:    0x00,          /* ----- */
3314:    0x00,          /* ----- */
3315:
3316:                /* 0xcf          */
3317:    0x00,          /* ----- */
3318:    0x66,          /* -##-##- */
3319:    0x66,          /* -##-##- */
3320:    0x00,          /* ----- */
3321:    0x3c,          /* --####-- */
3322:    0x18,          /* ---##--- */
3323:    0x18,          /* ---##--- */
3324:    0x18,          /* ---##--- */
3325:    0x18,          /* ---##--- */
3326:    0x18,          /* ---##--- */
3327:    0x3c,          /* --####-- */
3328:    0x00,          /* ----- */
3329:    0x00,          /* ----- */
3330:    0x00,          /* ----- */
3331:
3332:                /* 0xd0          */
3333:    0x00,          /* ----- */
3334:    0x00,          /* ----- */
3335:    0xf8,          /* #####--- */
3336:    0x6c,          /* -##-##- */
3337:    0x66,          /* -##-##- */
3338:    0x66,          /* -##-##- */
3339:    0xf6,          /* #####-##- */
3340:    0x66,          /* -##-##- */
3341:    0x66,          /* -##-##- */
3342:    0x6c,          /* -##-##- */
3343:    0xf8,          /* #####--- */
3344:    0x00,          /* ----- */
3345:    0x00,          /* ----- */
3346:    0x00,          /* ----- */
3347:
3348:                /* 0xd1          */
3349:    0x00,          /* ----- */
3350:    0x76,          /* -##-##- */

```

drivers/video/font-lat9-8x14.c

Page 51/62

```

3351:      0xdc,    /* ##-##-##-- */
3352:      0x00,    /* ----- */
3353:      0xc6,    /* ##---##- */
3354:      0xe6,    /* ###--##- */
3355:      0xf6,    /* ####-##- */
3356:      0xde,    /* ##-####- */
3357:      0xce,    /* ##--###- */
3358:      0xc6,    /* ##---##- */
3359:      0xc6,    /* ##---##- */
3360:      0x00,    /* ----- */
3361:      0x00,    /* ----- */
3362:      0x00,    /* ----- */
3363:
3364:          /* 0xd2          */
3365:      0x60,    /* -##----- */
3366:      0x30,    /* --##----- */
3367:      0x18,    /* ---##----- */
3368:      0x00,    /* ----- */
3369:      0x7c,    /* -#####-- */
3370:      0xc6,    /* ##---##- */
3371:      0xc6,    /* ##---##- */
3372:      0xc6,    /* ##---##- */
3373:      0xc6,    /* ##---##- */
3374:      0xc6,    /* ##---##- */
3375:      0x7c,    /* -#####-- */
3376:      0x00,    /* ----- */
3377:      0x00,    /* ----- */
3378:      0x00,    /* ----- */
3379:
3380:          /* 0xd3          */
3381:      0x0c,    /* -----##-- */
3382:      0x18,    /* ---##----- */
3383:      0x30,    /* --##----- */
3384:      0x00,    /* ----- */
3385:      0x7c,    /* -#####-- */
3386:      0xc6,    /* ##---##- */
3387:      0xc6,    /* ##---##- */
3388:      0xc6,    /* ##---##- */
3389:      0xc6,    /* ##---##- */
3390:      0xc6,    /* ##---##- */
3391:      0x7c,    /* -#####-- */
3392:      0x00,    /* ----- */
3393:      0x00,    /* ----- */
3394:      0x00,    /* ----- */
3395:
3396:          /* 0xd4          */
3397:      0x10,    /* ---#----- */
3398:      0x38,    /* --###----- */
3399:      0x6c,    /* -##-##-#-- */
3400:      0x00,    /* ----- */
3401:      0x7c,    /* -#####-- */
3402:      0xc6,    /* ##---##- */
3403:      0xc6,    /* ##---##- */
3404:      0xc6,    /* ##---##- */
3405:      0xc6,    /* ##---##- */
3406:      0xc6,    /* ##---##- */
3407:      0x7c,    /* -#####-- */
3408:      0x00,    /* ----- */
3409:      0x00,    /* ----- */
3410:      0x00,    /* ----- */
3411:
3412:          /* 0xd5          */
3413:      0x00,    /* ----- */
3414:      0x76,    /* -###-##-#- */
3415:      0xdc,    /* ##-##-##-- */
3416:      0x00,    /* ----- */
3417:      0x7c,    /* -#####-- */

```

drivers/video/font-lat9-8x14.c

Page 52/62

```

3418:      0xc6, /* ##---##- */
3419:      0xc6, /* ##---##- */
3420:      0xc6, /* ##---##- */
3421:      0xc6, /* ##---##- */
3422:      0xc6, /* ##---##- */
3423:      0x7c, /* -#####- */
3424:      0x00, /* ----- */
3425:      0x00, /* ----- */
3426:      0x00, /* ----- */
3427:
3428:          /* 0xd6 */
3429:      0x00, /* ----- */
3430:      0x6c, /* -##-##- */
3431:      0x6c, /* -##-##- */
3432:      0x00, /* ----- */
3433:      0x7c, /* -#####- */
3434:      0xc6, /* ##---##- */
3435:      0xc6, /* ##---##- */
3436:      0xc6, /* ##---##- */
3437:      0xc6, /* ##---##- */
3438:      0xc6, /* ##---##- */
3439:      0x7c, /* -#####- */
3440:      0x00, /* ----- */
3441:      0x00, /* ----- */
3442:      0x00, /* ----- */
3443:
3444:          /* 0xd7 */
3445:      0x00, /* ----- */
3446:      0x00, /* ----- */
3447:      0x00, /* ----- */
3448:      0x00, /* ----- */
3449:      0x00, /* ----- */
3450:      0x6c, /* -##-##- */
3451:      0x38, /* --###--- */
3452:      0x38, /* --###--- */
3453:      0x6c, /* -##-##- */
3454:      0x00, /* ----- */
3455:      0x00, /* ----- */
3456:      0x00, /* ----- */
3457:      0x00, /* ----- */
3458:      0x00, /* ----- */
3459:
3460:          /* 0xd8 */
3461:      0x00, /* ----- */
3462:      0x00, /* ----- */
3463:      0x7e, /* -#####- */
3464:      0xc6, /* ##---##- */
3465:      0xce, /* ##-###- */
3466:      0xde, /* ##-####- */
3467:      0xd6, /* ##-##-##- */
3468:      0xf6, /* #####-##- */
3469:      0xe6, /* ###-##- */
3470:      0xc6, /* ##---##- */
3471:      0xfc, /* #####-##- */
3472:      0x00, /* ----- */
3473:      0x00, /* ----- */
3474:      0x00, /* ----- */
3475:
3476:          /* 0xd9 */
3477:      0x60, /* -##----- */
3478:      0x30, /* ---##----- */
3479:      0x18, /* ----##---- */
3480:      0x00, /* ----- */
3481:      0xc6, /* ##---##- */
3482:      0xc6, /* ##---##- */
3483:      0xc6, /* ##---##- */
3484:      0xc6, /* ##---##- */

```

drivers/video/font-lat9-8x14.c

Page 53/62

```

3485:      0xc6,    /* ##---##- */
3486:      0xc6,    /* ##---##- */
3487:      0x7c,    /* -#####- */
3488:      0x00,    /* ----- */
3489:      0x00,    /* ----- */
3490:      0x00,    /* ----- */
3491:
3492:          /* 0xda          */
3493:      0x0c,    /* ----##- */
3494:      0x18,    /* ---##--- */
3495:      0x30,    /* --##---- */
3496:      0x00,    /* ----- */
3497:      0xc6,    /* ##---##- */
3498:      0xc6,    /* ##---##- */
3499:      0xc6,    /* ##---##- */
3500:      0xc6,    /* ##---##- */
3501:      0xc6,    /* ##---##- */
3502:      0xc6,    /* ##---##- */
3503:      0x7c,    /* -#####- */
3504:      0x00,    /* ----- */
3505:      0x00,    /* ----- */
3506:      0x00,    /* ----- */
3507:
3508:          /* 0xdb          */
3509:      0x10,    /* ---#----- */
3510:      0x38,    /* --###--- */
3511:      0x6c,    /* -##-##- */
3512:      0x00,    /* ----- */
3513:      0xc6,    /* ##---##- */
3514:      0xc6,    /* ##---##- */
3515:      0xc6,    /* ##---##- */
3516:      0xc6,    /* ##---##- */
3517:      0xc6,    /* ##---##- */
3518:      0xc6,    /* ##---##- */
3519:      0x7c,    /* -#####- */
3520:      0x00,    /* ----- */
3521:      0x00,    /* ----- */
3522:      0x00,    /* ----- */
3523:
3524:          /* 0xdc          */
3525:      0x00,    /* ----- */
3526:      0x6c,    /* -##-##- */
3527:      0x6c,    /* -##-##- */
3528:      0x00,    /* ----- */
3529:      0xc6,    /* ##---##- */
3530:      0xc6,    /* ##---##- */
3531:      0xc6,    /* ##---##- */
3532:      0xc6,    /* ##---##- */
3533:      0xc6,    /* ##---##- */
3534:      0xc6,    /* ##---##- */
3535:      0x7c,    /* -#####- */
3536:      0x00,    /* ----- */
3537:      0x00,    /* ----- */
3538:      0x00,    /* ----- */
3539:
3540:          /* 0xdd          */
3541:      0x06,    /* -----##- */
3542:      0x0c,    /* ----##- */
3543:      0x18,    /* ---##--- */
3544:      0x00,    /* ----- */
3545:      0x66,    /* -##---##- */
3546:      0x66,    /* -##---##- */
3547:      0x66,    /* -##---##- */
3548:      0x3c,    /* --#####- */
3549:      0x18,    /* ---##--- */
3550:      0x18,    /* ---##--- */
3551:      0x3c,    /* --#####- */

```

drivers/video/font-lat9-8x14.c

Page 54/62

```

3552:      0x00, /* ----- */
3553:      0x00, /* ----- */
3554:      0x00, /* ----- */
3555:
3556:          /* 0xde */
3557:      0x00, /* ----- */
3558:      0x00, /* ----- */
3559:      0xf0, /* #####- */
3560:      0x60, /* -##----- */
3561:      0x7c, /* -#####- */
3562:      0x66, /* -##-##- */
3563:      0x66, /* -##-##- */
3564:      0x66, /* -##-##- */
3565:      0x7c, /* -#####- */
3566:      0x60, /* -##----- */
3567:      0xf0, /* #####- */
3568:      0x00, /* ----- */
3569:      0x00, /* ----- */
3570:      0x00, /* ----- */
3571:
3572:          /* 0xdf */
3573:      0x00, /* ----- */
3574:      0x00, /* ----- */
3575:      0x7c, /* -#####- */
3576:      0xc6, /* ##---##- */
3577:      0xc6, /* ##---##- */
3578:      0xc6, /* ##---##- */
3579:      0xcc, /* ##---##- */
3580:      0xc6, /* ##---##- */
3581:      0xc6, /* ##---##- */
3582:      0xd6, /* ##-##-##- */
3583:      0xdc, /* ##-##-##- */
3584:      0x80, /* #----- */
3585:      0x00, /* ----- */
3586:      0x00, /* ----- */
3587:
3588:          /* 0xe0 */
3589:      0x00, /* ----- */
3590:      0x60, /* -##----- */
3591:      0x30, /* --##----- */
3592:      0x18, /* ---##----- */
3593:      0x00, /* ----- */
3594:      0x78, /* -#####- */
3595:      0x0c, /* ----##- */
3596:      0x7c, /* -#####- */
3597:      0xcc, /* ##---##- */
3598:      0xdc, /* ##-##-##- */
3599:      0x76, /* -##-##- */
3600:      0x00, /* ----- */
3601:      0x00, /* ----- */
3602:      0x00, /* ----- */
3603:
3604:          /* 0xe1 */
3605:      0x00, /* ----- */
3606:      0x18, /* ----##- */
3607:      0x30, /* --##----- */
3608:      0x60, /* -##----- */
3609:      0x00, /* ----- */
3610:      0x78, /* -#####- */
3611:      0x0c, /* ----##- */
3612:      0x7c, /* -#####- */
3613:      0xcc, /* ##---##- */
3614:      0xdc, /* ##-##-##- */
3615:      0x76, /* -##-##- */
3616:      0x00, /* ----- */
3617:      0x00, /* ----- */
3618:      0x00, /* ----- */

```

drivers/video/font-lat9-8x14.c

Page 55/62

```

3619:
3620:          /* 0xe2          */
3621:    0x00,  /* ----- */
3622:    0x30,  /* --##---- */
3623:    0x78,  /* -####--- */
3624:    0xcc,  /* ##--##-- */
3625:    0x00,  /* ----- */
3626:    0x78,  /* -####--- */
3627:    0x0c,  /* ----##-- */
3628:    0x7c,  /* -#####-- */
3629:    0xcc,  /* ##--##-- */
3630:    0xdc,  /* #-###-- */
3631:    0x76,  /* -###-##- */
3632:    0x00,  /* ----- */
3633:    0x00,  /* ----- */
3634:    0x00,  /* ----- */
3635:
3636:          /* 0xe3          */
3637:    0x00,  /* ----- */
3638:    0x00,  /* ----- */
3639:    0x76,  /* -###-##- */
3640:    0xdc,  /* #-###-- */
3641:    0x00,  /* ----- */
3642:    0x78,  /* -####--- */
3643:    0x0c,  /* ----##-- */
3644:    0x7c,  /* -#####-- */
3645:    0xcc,  /* ##--##-- */
3646:    0xdc,  /* #-###-- */
3647:    0x76,  /* -###-##- */
3648:    0x00,  /* ----- */
3649:    0x00,  /* ----- */
3650:    0x00,  /* ----- */
3651:
3652:          /* 0xe4          */
3653:    0x00,  /* ----- */
3654:    0x00,  /* ----- */
3655:    0x6c,  /* -##-##-- */
3656:    0x6c,  /* -##-##-- */
3657:    0x00,  /* ----- */
3658:    0x78,  /* -####--- */
3659:    0x0c,  /* ----##-- */
3660:    0x7c,  /* -#####-- */
3661:    0xcc,  /* ##--##-- */
3662:    0xdc,  /* #-###-- */
3663:    0x76,  /* -###-##- */
3664:    0x00,  /* ----- */
3665:    0x00,  /* ----- */
3666:    0x00,  /* ----- */
3667:
3668:          /* 0xe5          */
3669:    0x00,  /* ----- */
3670:    0x38,  /* ---###--- */
3671:    0x6c,  /* -##-##-- */
3672:    0x38,  /* --###--- */
3673:    0x00,  /* ----- */
3674:    0x78,  /* -####--- */
3675:    0x0c,  /* ----##-- */
3676:    0x7c,  /* -#####-- */
3677:    0xcc,  /* ##--##-- */
3678:    0xdc,  /* #-###-- */
3679:    0x76,  /* -###-##- */
3680:    0x00,  /* ----- */
3681:    0x00,  /* ----- */
3682:    0x00,  /* ----- */
3683:
3684:          /* 0xe6          */
3685:    0x00,  /* ----- */

```

drivers/video/font-lat9-8x14.c

Page 56/62

```

3686:      0x00,    /* ----- */
3687:      0x00,    /* ----- */
3688:      0x00,    /* ----- */
3689:      0x7e,    /* -#####- */
3690:      0xdb,    /* ##-##-## */
3691:      0x1b,    /* ---##-## */
3692:      0x7f,    /* -##### */
3693:      0xd8,    /* ##-##-   */
3694:      0xdb,    /* ##-##-## */
3695:      0x7e,    /* -#####- */
3696:      0x00,    /* ----- */
3697:      0x00,    /* ----- */
3698:      0x00,    /* ----- */
3699:
3700:          /* 0xe7      */
3701:      0x00,    /* ----- */
3702:      0x00,    /* ----- */
3703:      0x00,    /* ----- */
3704:      0x00,    /* ----- */
3705:      0x00,    /* ----- */
3706:      0x7c,    /* -#####- */
3707:      0xc6,    /* ##-   ##- */
3708:      0xc0,    /* ##-   -   */
3709:      0xc0,    /* ##-   -   */
3710:      0xc6,    /* ##-   ##- */
3711:      0x7c,    /* -#####- */
3712:      0x18,    /* ---##-   */
3713:      0x6c,    /* -##-##-   */
3714:      0x38,    /* --###-   */
3715:
3716:          /* 0xe8      */
3717:      0x00,    /* ----- */
3718:      0x30,    /* --##-   */
3719:      0x18,    /* ---##-   */
3720:      0x0c,    /* ----##-  */
3721:      0x00,    /* ----- */
3722:      0x7c,    /* -#####- */
3723:      0xc6,    /* ##-   ##- */
3724:      0xfe,    /* #####-   */
3725:      0xc0,    /* ##-   -   */
3726:      0xc6,    /* ##-   ##- */
3727:      0x7c,    /* -#####- */
3728:      0x00,    /* ----- */
3729:      0x00,    /* ----- */
3730:      0x00,    /* ----- */
3731:
3732:          /* 0xe9      */
3733:      0x00,    /* ----- */
3734:      0x0c,    /* ----##-  */
3735:      0x18,    /* ---##-   */
3736:      0x30,    /* --##-   */
3737:      0x00,    /* ----- */
3738:      0x7c,    /* -#####- */
3739:      0xc6,    /* ##-   ##- */
3740:      0xfe,    /* #####-   */
3741:      0xc0,    /* ##-   -   */
3742:      0xc6,    /* ##-   ##- */
3743:      0x7c,    /* -#####- */
3744:      0x00,    /* ----- */
3745:      0x00,    /* ----- */
3746:      0x00,    /* ----- */
3747:
3748:          /* 0xea      */
3749:      0x00,    /* ----- */
3750:      0x10,    /* ---#-   */
3751:      0x38,    /* --###-   */
3752:      0x6c,    /* -##-##-  */

```


drivers/video/font-lat9-8x14.c

Page 57/62

```

3753:      0x00, /* ----- */
3754:      0x7c, /* -##### */
3755:      0xc6, /* ##----## */
3756:      0xfe, /* #####- */
3757:      0xc0, /* ##----- */
3758:      0xc6, /* ##----## */
3759:      0x7c, /* -##### */
3760:      0x00, /* ----- */
3761:      0x00, /* ----- */
3762:      0x00, /* ----- */
3763:
3764:      /* 0xeb */
3765:      0x00, /* ----- */
3766:      0x00, /* ----- */
3767:      0x6c, /* -##-##- */
3768:      0x6c, /* -##-##- */
3769:      0x00, /* ----- */
3770:      0x7c, /* -##### */
3771:      0xc6, /* ##----## */
3772:      0xfe, /* #####- */
3773:      0xc0, /* ##----- */
3774:      0xc6, /* ##----## */
3775:      0x7c, /* -##### */
3776:      0x00, /* ----- */
3777:      0x00, /* ----- */
3778:      0x00, /* ----- */
3779:
3780:      /* 0xec */
3781:      0x00, /* ----- */
3782:      0x60, /* -##----- */
3783:      0x30, /* --##----- */
3784:      0x18, /* ---##----- */
3785:      0x00, /* ----- */
3786:      0x38, /* --###----- */
3787:      0x18, /* ---##----- */
3788:      0x18, /* ---##----- */
3789:      0x18, /* ---##----- */
3790:      0x18, /* ---##----- */
3791:      0x3c, /* --####----- */
3792:      0x00, /* ----- */
3793:      0x00, /* ----- */
3794:      0x00, /* ----- */
3795:
3796:      /* 0xed */
3797:      0x00, /* ----- */
3798:      0x0c, /* ----##----- */
3799:      0x18, /* ---##----- */
3800:      0x30, /* --##----- */
3801:      0x00, /* ----- */
3802:      0x38, /* --###----- */
3803:      0x18, /* ---##----- */
3804:      0x18, /* ---##----- */
3805:      0x18, /* ---##----- */
3806:      0x18, /* ---##----- */
3807:      0x3c, /* --####----- */
3808:      0x00, /* ----- */
3809:      0x00, /* ----- */
3810:      0x00, /* ----- */
3811:
3812:      /* 0xee */
3813:      0x00, /* ----- */
3814:      0x18, /* ---##----- */
3815:      0x3c, /* --####----- */
3816:      0x66, /* -##-##- */
3817:      0x00, /* ----- */
3818:      0x38, /* --###----- */
3819:      0x18, /* ---##----- */

```

drivers/video/font-lat9-8x14.c

Page 58/62

```

3820:      0x18,    /* ----##---- */
3821:      0x18,    /* ----##---- */
3822:      0x18,    /* ----##---- */
3823:      0x3c,    /* --####-- */
3824:      0x00,    /* ----- */
3825:      0x00,    /* ----- */
3826:      0x00,    /* ----- */
3827:
3828:          /* 0xef */
3829:      0x00,    /* ----- */
3830:      0x00,    /* ----- */
3831:      0x6c,    /* -##-##- */
3832:      0x6c,    /* -##-##- */
3833:      0x00,    /* ----- */
3834:      0x38,    /* --###--- */
3835:      0x18,    /* ----##---- */
3836:      0x18,    /* ----##---- */
3837:      0x18,    /* ----##---- */
3838:      0x18,    /* ----##---- */
3839:      0x3c,    /* --####-- */
3840:      0x00,    /* ----- */
3841:      0x00,    /* ----- */
3842:      0x00,    /* ----- */
3843:
3844:          /* 0xf0 */
3845:      0x00,    /* ----- */
3846:      0x78,    /* -####--- */
3847:      0x30,    /* --##----- */
3848:      0x78,    /* -####--- */
3849:      0x0c,    /* ----##-- */
3850:      0x7e,    /* -#####- */
3851:      0xc6,    /* ##---##- */
3852:      0xc6,    /* ##---##- */
3853:      0xc6,    /* ##---##- */
3854:      0xc6,    /* ##---##- */
3855:      0x7c,    /* -#####- */
3856:      0x00,    /* ----- */
3857:      0x00,    /* ----- */
3858:      0x00,    /* ----- */
3859:
3860:          /* 0xf1 */
3861:      0x00,    /* ----- */
3862:      0x00,    /* ----- */
3863:      0x76,    /* -###-##- */
3864:      0xdc,    /* ##-###-- */
3865:      0x00,    /* ----- */
3866:      0xdc,    /* ##-###-- */
3867:      0x66,    /* -##-##- */
3868:      0x66,    /* -##-##- */
3869:      0x66,    /* -##-##- */
3870:      0x66,    /* -##-##- */
3871:      0x66,    /* -##-##- */
3872:      0x00,    /* ----- */
3873:      0x00,    /* ----- */
3874:      0x00,    /* ----- */
3875:
3876:          /* 0xf2 */
3877:      0x00,    /* ----- */
3878:      0x60,    /* -##----- */
3879:      0x30,    /* --##----- */
3880:      0x18,    /* ----##---- */
3881:      0x00,    /* ----- */
3882:      0x7c,    /* -#####- */
3883:      0xc6,    /* ##---##- */
3884:      0xc6,    /* ##---##- */
3885:      0xc6,    /* ##---##- */
3886:      0xc6,    /* ##---##- */

```

drivers/video/font-lat9-8x14.c

Page 59/62

```

3887:      0x7c, /* #####-- */
3888:      0x00, /* ----- */
3889:      0x00, /* ----- */
3890:      0x00, /* ----- */
3891:
3892:          /* 0xf3 */
3893:      0x00, /* ----- */
3894:      0x0c, /* ----##-- */
3895:      0x18, /* ---##--- */
3896:      0x30, /* --##---- */
3897:      0x00, /* ----- */
3898:      0x7c, /* #####-- */
3899:      0xc6, /* ##---##- */
3900:      0xc6, /* ##---##- */
3901:      0xc6, /* ##---##- */
3902:      0xc6, /* ##---##- */
3903:      0x7c, /* #####-- */
3904:      0x00, /* ----- */
3905:      0x00, /* ----- */
3906:      0x00, /* ----- */
3907:
3908:          /* 0xf4 */
3909:      0x00, /* ----- */
3910:      0x10, /* ---#---- */
3911:      0x38, /* --###--- */
3912:      0x6c, /* -##-##-- */
3913:      0x00, /* ----- */
3914:      0x7c, /* #####-- */
3915:      0xc6, /* ##---##- */
3916:      0xc6, /* ##---##- */
3917:      0xc6, /* ##---##- */
3918:      0xc6, /* ##---##- */
3919:      0x7c, /* #####-- */
3920:      0x00, /* ----- */
3921:      0x00, /* ----- */
3922:      0x00, /* ----- */
3923:
3924:          /* 0xf5 */
3925:      0x00, /* ----- */
3926:      0x00, /* ----- */
3927:      0x76, /* -###-##- */
3928:      0xdc, /* ##-###-- */
3929:      0x00, /* ----- */
3930:      0x7c, /* #####-- */
3931:      0xc6, /* ##---##- */
3932:      0xc6, /* ##---##- */
3933:      0xc6, /* ##---##- */
3934:      0xc6, /* ##---##- */
3935:      0x7c, /* #####-- */
3936:      0x00, /* ----- */
3937:      0x00, /* ----- */
3938:      0x00, /* ----- */
3939:
3940:          /* 0xf6 */
3941:      0x00, /* ----- */
3942:      0x00, /* ----- */
3943:      0x6c, /* -##-##-- */
3944:      0x6c, /* -##-##-- */
3945:      0x00, /* ----- */
3946:      0x7c, /* #####-- */
3947:      0xc6, /* ##---##- */
3948:      0xc6, /* ##---##- */
3949:      0xc6, /* ##---##- */
3950:      0xc6, /* ##---##- */
3951:      0x7c, /* #####-- */
3952:      0x00, /* ----- */
3953:      0x00, /* ----- */

```

drivers/video/font-lat9-8x14.c

Page 60/62

```

3954:      0x00,    /* ----- */
3955:
3956:          /* 0xf7 */
3957:      0x00,    /* ----- */
3958:      0x00,    /* ----- */
3959:      0x00,    /* ----- */
3960:      0x18,    /* ---##--- */
3961:      0x18,    /* ---##--- */
3962:      0x00,    /* ----- */
3963:      0x7e,    /* -#####- */
3964:      0x00,    /* ----- */
3965:      0x18,    /* ---##--- */
3966:      0x18,    /* ---##--- */
3967:      0x00,    /* ----- */
3968:      0x00,    /* ----- */
3969:      0x00,    /* ----- */
3970:      0x00,    /* ----- */
3971:
3972:          /* 0xf8 */
3973:      0x00,    /* ----- */
3974:      0x00,    /* ----- */
3975:      0x00,    /* ----- */
3976:      0x00,    /* ----- */
3977:      0x00,    /* ----- */
3978:      0x7e,    /* -#####- */
3979:      0xce,    /* ##---##- */
3980:      0xde,    /* ##-####- */
3981:      0xf6,    /* ####-##- */
3982:      0xe6,    /* ###---##- */
3983:      0xfc,    /* #####--- */
3984:      0x00,    /* ----- */
3985:      0x00,    /* ----- */
3986:      0x00,    /* ----- */
3987:
3988:          /* 0xf9 */
3989:      0x00,    /* ----- */
3990:      0xc0,    /* ##----- */
3991:      0x60,    /* -##----- */
3992:      0x30,    /* --##----- */
3993:      0x00,    /* ----- */
3994:      0xcc,    /* ##---##- */
3995:      0xcc,    /* ##---##- */
3996:      0xcc,    /* ##---##- */
3997:      0xcc,    /* ##---##- */
3998:      0xcc,    /* ##---##- */
3999:      0x76,    /* -###-##- */
4000:      0x00,    /* ----- */
4001:      0x00,    /* ----- */
4002:      0x00,    /* ----- */
4003:
4004:          /* 0xfa */
4005:      0x00,    /* ----- */
4006:      0x0c,    /* ----##-- */
4007:      0x18,    /* ---##--- */
4008:      0x30,    /* --##----- */
4009:      0x00,    /* ----- */
4010:      0xcc,    /* ##---##- */
4011:      0xcc,    /* ##---##- */
4012:      0xcc,    /* ##---##- */
4013:      0xcc,    /* ##---##- */
4014:      0xcc,    /* ##---##- */
4015:      0x76,    /* -###-##- */
4016:      0x00,    /* ----- */
4017:      0x00,    /* ----- */
4018:      0x00,    /* ----- */
4019:
4020:          /* 0xfb */

```

drivers/video/font-lat9-8x14.c

Page 61/62

```

4021:      0x00,    /* ----- */
4022:      0x30,    /* --##---- */
4023:      0x78,    /* -####--- */
4024:      0xcc,    /* ##--##-- */
4025:      0x00,    /* ----- */
4026:      0xcc,    /* ##--##-- */
4027:      0xcc,    /* ##--##-- */
4028:      0xcc,    /* ##--##-- */
4029:      0xcc,    /* ##--##-- */
4030:      0xcc,    /* ##--##-- */
4031:      0x76,    /* -###-##- */
4032:      0x00,    /* ----- */
4033:      0x00,    /* ----- */
4034:      0x00,    /* ----- */
4035:
4036:          /* 0xfc */
4037:      0x00,    /* ----- */
4038:      0x00,    /* ----- */
4039:      0xcc,    /* ##--##-- */
4040:      0xcc,    /* ##--##-- */
4041:      0x00,    /* ----- */
4042:      0xcc,    /* ##--##-- */
4043:      0xcc,    /* ##--##-- */
4044:      0xcc,    /* ##--##-- */
4045:      0xcc,    /* ##--##-- */
4046:      0xcc,    /* ##--##-- */
4047:      0x76,    /* -###-##- */
4048:      0x00,    /* ----- */
4049:      0x00,    /* ----- */
4050:      0x00,    /* ----- */
4051:
4052:          /* 0xfd */
4053:      0x00,    /* ----- */
4054:      0x0c,    /* ----##-- */
4055:      0x18,    /* ---##--- */
4056:      0x30,    /* --##---- */
4057:      0x00,    /* ----- */
4058:      0xc6,    /* ###---##- */
4059:      0xc6,    /* ##---##- */
4060:      0xc6,    /* ##---##- */
4061:      0xce,    /* ##--###- */
4062:      0x76,    /* -###-##- */
4063:      0x06,    /* -----##- */
4064:      0xc6,    /* ##---##- */
4065:      0x7c,    /* -#####- */
4066:      0x00,    /* ----- */
4067:
4068:          /* 0xfe */
4069:      0x00,    /* ----- */
4070:      0x00,    /* ----- */
4071:      0x00,    /* ----- */
4072:      0xf0,    /* #####---- */
4073:      0x60,    /* -##----- */
4074:      0x60,    /* -##----- */
4075:      0x78,    /* -####--- */
4076:      0x6c,    /* -##-##-- */
4077:      0x6c,    /* -##-##-- */
4078:      0x6c,    /* -##-##-- */
4079:      0x78,    /* -####--- */
4080:      0x60,    /* -##----- */
4081:      0x60,    /* -##----- */
4082:      0xf0,    /* #####---- */
4083:
4084:          /* 0xff */
4085:      0x00,    /* ----- */
4086:      0x00,    /* ----- */
4087:      0xc6,    /* ##---##- */

```

drivers/video/font-lat9-8x14.c

Page 62/62

```
4088:      0xc6,    /* ##---##- */
4089:      0x00,    /* ----- */
4090:      0xc6,    /* ##---##- */
4091:      0xc6,    /* ##---##- */
4092:      0xc6,    /* ##---##- */
4093:      0xce,    /* ##--###- */
4094:      0x76,    /* -###-##- */
4095:      0x06,    /* -----##- */
4096:      0xc6,    /* ##---##- */
4097:      0x7c,    /* -#####- */
4098:      0x00,    /* ----- */
4099:
4100: };
4101:
4102: static unsigned char cursorshape_8x14[] = {
4103:     0x00,    /* ----- */
4104:     0x00,    /* ----- */
4105:     0x00,    /* ----- */
4106:     0x00,    /* ----- */
4107:     0x00,    /* ----- */
4108:     0x00,    /* ----- */
4109:     0x00,    /* ----- */
4110:     0x00,    /* ----- */
4111:     0x00,    /* ----- */
4112:     0x00,    /* ----- */
4113:     0x00,    /* ----- */
4114:     0x00,    /* ----- */
4115:     0xFF,    /* ##### */
4116:     0xFF,    /* ##### */
4117: };
4118:
4119: struct fbcon_font_desc font_lat9_8x14 = {
4120:     "VGA8x14",
4121:     8,
4122:     14,
4123:     fontdata_8x14,
4124:     cursorshape_8x14
4125: };
```

drivers/video/font-lat9-8x16.c

Page 1/70

```

1: #include <fiwix/font.h>
2:
3: static unsigned char fontdata_8x16[] = {
4:     /* 0x00          */
5:     0x00, /* ----- */
6:     0x00, /* ----- */
7:     0x7e, /* -#####- */
8:     0xc3, /* ##-----# */
9:     0x99, /* #--#--# */
10:    0x99, /* #--#--# */
11:    0xf3, /* #####--# */
12:    0xe7, /* ###--### */
13:    0xe7, /* ###--### */
14:    0xff, /* ######## */
15:    0xe7, /* ###--### */
16:    0xe7, /* ###--### */
17:    0x7e, /* -#####- */
18:    0x00, /* ----- */
19:    0x00, /* ----- */
20:    0x00, /* ----- */
21:
22:     /* 0x01          */
23:     0x00, /* ----- */
24:     0x00, /* ----- */
25:     0x00, /* ----- */
26:     0x00, /* ----- */
27:     0x00, /* ----- */
28:     0x76, /* -###-##- */
29:     0xdc, /* #-###-- */
30:     0x00, /* ----- */
31:     0x76, /* -###-##- */
32:     0xdc, /* #-###-- */
33:     0x00, /* ----- */
34:     0x00, /* ----- */
35:     0x00, /* ----- */
36:     0x00, /* ----- */
37:     0x00, /* ----- */
38:     0x00, /* ----- */
39:
40:     /* 0x02          */
41:     0x00, /* ----- */
42:     0x00, /* ----- */
43:     0x6e, /* -##-###- */
44:     0xf8, /* #####--- */
45:     0xd8, /* ##-##--- */
46:     0xd8, /* ##-##--- */
47:     0xdc, /* #-###-- */
48:     0xd8, /* #-##--- */
49:     0xd8, /* #-##--- */
50:     0xd8, /* #-##--- */
51:     0xf8, /* #####--- */
52:     0x6e, /* -##-###- */
53:     0x00, /* ----- */
54:     0x00, /* ----- */
55:     0x00, /* ----- */
56:     0x00, /* ----- */
57:
58:     /* 0x03          */
59:     0x00, /* ----- */
60:     0x00, /* ----- */
61:     0x00, /* ----- */
62:     0x00, /* ----- */
63:     0x00, /* ----- */
64:     0x6e, /* -##-###- */
65:     0xdb, /* #-##-## */
66:     0xdb, /* #-##-## */
67:     0xdf, /* #-##### */

```

drivers/video/font-lat9-8x16.c Page 2/70

```

68:      0xd8,    /* ##-##--- */
69:      0xdb,    /* ##-##-## */
70:      0x6e,    /* -##-##-# */
71:      0x00,    /* ----- */
72:      0x00,    /* ----- */
73:      0x00,    /* ----- */
74:      0x00,    /* ----- */
75:
76:                /* 0x04 */
77:      0x00,    /* ----- */
78:      0x00,    /* ----- */
79:      0x00,    /* ----- */
80:      0x00,    /* ----- */
81:      0x10,    /* ---#---- */
82:      0x38,    /* --###--- */
83:      0x7c,    /* -#####- */
84:      0xfe,    /* #####-#- */
85:      0x7c,    /* -#####- */
86:      0x38,    /* --###--- */
87:      0x10,    /* ---#---- */
88:      0x00,    /* ----- */
89:      0x00,    /* ----- */
90:      0x00,    /* ----- */
91:      0x00,    /* ----- */
92:      0x00,    /* ----- */
93:
94:                /* 0x05 */
95:      0x00,    /* ----- */
96:      0x88,    /* #---#--- */
97:      0x88,    /* #---#--- */
98:      0xf8,    /* #####--- */
99:      0x88,    /* #---#--- */
100:     0x88,    /* #---#--- */
101:     0x00,    /* ----- */
102:     0x3e,    /* --#####- */
103:     0x08,    /* ----#--- */
104:     0x08,    /* ----#--- */
105:     0x08,    /* ----#--- */
106:     0x08,    /* ----#--- */
107:     0x00,    /* ----- */
108:     0x00,    /* ----- */
109:     0x00,    /* ----- */
110:     0x00,    /* ----- */
111:
112:                /* 0x06 */
113:     0x00,    /* ----- */
114:     0xf8,    /* #####--- */
115:     0x80,    /* #----- */
116:     0xe0,    /* ###----- */
117:     0x80,    /* #----- */
118:     0x80,    /* #----- */
119:     0x00,    /* ----- */
120:     0x3e,    /* --#####- */
121:     0x20,    /* --#----- */
122:     0x38,    /* --###--- */
123:     0x20,    /* --#----- */
124:     0x20,    /* --#----- */
125:     0x00,    /* ----- */
126:     0x00,    /* ----- */
127:     0x00,    /* ----- */
128:     0x00,    /* ----- */
129:
130:                /* 0x07 */
131:     0x00,    /* ----- */
132:     0x70,    /* -###----- */
133:     0x88,    /* #---#--- */
134:     0x80,    /* #----- */

```


drivers/video/font-lat9-8x16.c Page 3/70

```

135:      0x88,    /* #---#--- */
136:      0x70,    /* -###----- */
137:      0x00,    /* ----- */
138:      0x3c,    /* --####-- */
139:      0x22,    /* --#---#- */
140:      0x3c,    /* --####-- */
141:      0x24,    /* --#--#-- */
142:      0x22,    /* --#---#- */
143:      0x00,    /* ----- */
144:      0x00,    /* ----- */
145:      0x00,    /* ----- */
146:      0x00,    /* ----- */
147:
148:          /* 0x08 */
149:      0x00,    /* ----- */
150:      0x80,    /* #----- */
151:      0x80,    /* #----- */
152:      0x80,    /* #----- */
153:      0x80,    /* #----- */
154:      0xf8,    /* #####--- */
155:      0x00,    /* ----- */
156:      0x3e,    /* --#####- */
157:      0x20,    /* --#----- */
158:      0x38,    /* --###--- */
159:      0x20,    /* --#----- */
160:      0x20,    /* --#----- */
161:      0x00,    /* ----- */
162:      0x00,    /* ----- */
163:      0x00,    /* ----- */
164:      0x00,    /* ----- */
165:
166:          /* 0x09 */
167:      0x11,    /* ---#---# */
168:      0x44,    /* -#---#-- */
169:      0x11,    /* ---#---# */
170:      0x44,    /* -#---#-- */
171:      0x11,    /* ---#---# */
172:      0x44,    /* -#---#-- */
173:      0x11,    /* ---#---# */
174:      0x44,    /* -#---#-- */
175:      0x11,    /* ---#---# */
176:      0x44,    /* -#---#-- */
177:      0x11,    /* ---#---# */
178:      0x44,    /* -#---#-- */
179:      0x11,    /* ---#---# */
180:      0x44,    /* -#---#-- */
181:      0x11,    /* ---#---# */
182:      0x44,    /* -#---#-- */
183:
184:          /* 0x0a */
185:      0x55,    /* -#-#-#-# */
186:      0xaa,    /* #-#-#-#- */
187:      0x55,    /* -#-#-#-# */
188:      0xaa,    /* #-#-#-#- */
189:      0x55,    /* -#-#-#-# */
190:      0xaa,    /* #-#-#-#- */
191:      0x55,    /* -#-#-#-# */
192:      0xaa,    /* #-#-#-#- */
193:      0x55,    /* -#-#-#-# */
194:      0xaa,    /* #-#-#-#- */
195:      0x55,    /* -#-#-#-# */
196:      0xaa,    /* #-#-#-#- */
197:      0x55,    /* -#-#-#-# */
198:      0xaa,    /* #-#-#-#- */
199:      0x55,    /* -#-#-#-# */
200:      0xaa,    /* #-#-#-#- */
201:

```

drivers/video/font-lat9-8x16.c

Page 4/70

```

202:                /* 0x0b          */
203:                0xdd, /* ##-##-# */
204:                0x77, /* -##-## */
205:                0xdd, /* ##-##-# */
206:                0x77, /* -##-## */
207:                0xdd, /* ##-##-# */
208:                0x77, /* -##-## */
209:                0xdd, /* ##-##-# */
210:                0x77, /* -##-## */
211:                0xdd, /* ##-##-# */
212:                0x77, /* -##-## */
213:                0xdd, /* ##-##-# */
214:                0x77, /* -##-## */
215:                0xdd, /* ##-##-# */
216:                0x77, /* -##-## */
217:                0xdd, /* ##-##-# */
218:                0x77, /* -##-## */
219:
220:                /* 0x0c          */
221:                0xff, /* ##### */
222:                0xff, /* ##### */
223:                0xff, /* ##### */
224:                0xff, /* ##### */
225:                0xff, /* ##### */
226:                0xff, /* ##### */
227:                0xff, /* ##### */
228:                0xff, /* ##### */
229:                0xff, /* ##### */
230:                0xff, /* ##### */
231:                0xff, /* ##### */
232:                0xff, /* ##### */
233:                0xff, /* ##### */
234:                0xff, /* ##### */
235:                0xff, /* ##### */
236:                0xff, /* ##### */
237:
238:                /* 0x0d          */
239:                0x00, /* ----- */
240:                0x00, /* ----- */
241:                0x00, /* ----- */
242:                0x00, /* ----- */
243:                0x00, /* ----- */
244:                0x00, /* ----- */
245:                0x00, /* ----- */
246:                0xff, /* ##### */
247:                0xff, /* ##### */
248:                0xff, /* ##### */
249:                0xff, /* ##### */
250:                0xff, /* ##### */
251:                0xff, /* ##### */
252:                0xff, /* ##### */
253:                0xff, /* ##### */
254:                0xff, /* ##### */
255:
256:                /* 0x0e          */
257:                0xff, /* ##### */
258:                0xff, /* ##### */
259:                0xff, /* ##### */
260:                0xff, /* ##### */
261:                0xff, /* ##### */
262:                0xff, /* ##### */
263:                0xff, /* ##### */
264:                0x00, /* ----- */
265:                0x00, /* ----- */
266:                0x00, /* ----- */
267:                0x00, /* ----- */
268:                0x00, /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 5/70

```

269:      0x00, /* ----- */
270:      0x00, /* ----- */
271:      0x00, /* ----- */
272:      0x00, /* ----- */
273:
274:          /* 0x0f */
275:      0xf0, /* #####----- */
276:      0xf0, /* #####----- */
277:      0xf0, /* #####----- */
278:      0xf0, /* #####----- */
279:      0xf0, /* #####----- */
280:      0xf0, /* #####----- */
281:      0xf0, /* #####----- */
282:      0xf0, /* #####----- */
283:      0xf0, /* #####----- */
284:      0xf0, /* #####----- */
285:      0xf0, /* #####----- */
286:      0xf0, /* #####----- */
287:      0xf0, /* #####----- */
288:      0xf0, /* #####----- */
289:      0xf0, /* #####----- */
290:      0xf0, /* #####----- */
291:
292:          /* 0x10 */
293:      0x0f, /* ----##### */
294:      0x0f, /* ----##### */
295:      0x0f, /* ----##### */
296:      0x0f, /* ----##### */
297:      0x0f, /* ----##### */
298:      0x0f, /* ----##### */
299:      0x0f, /* ----##### */
300:      0x0f, /* ----##### */
301:      0x0f, /* ----##### */
302:      0x0f, /* ----##### */
303:      0x0f, /* ----##### */
304:      0x0f, /* ----##### */
305:      0x0f, /* ----##### */
306:      0x0f, /* ----##### */
307:      0x0f, /* ----##### */
308:      0x0f, /* ----##### */
309:
310:          /* 0x11 */
311:      0x00, /* ----- */
312:      0x88, /* #---#--- */
313:      0xc8, /* ##--#--- */
314:      0xa8, /* #-#-#--- */
315:      0x98, /* #--##--- */
316:      0x88, /* #---#--- */
317:      0x00, /* ----- */
318:      0x20, /* --#----- */
319:      0x20, /* --#----- */
320:      0x20, /* --#----- */
321:      0x20, /* --#----- */
322:      0x3e, /* --#####- */
323:      0x00, /* ----- */
324:      0x00, /* ----- */
325:      0x00, /* ----- */
326:      0x00, /* ----- */
327:
328:          /* 0x12 */
329:      0x00, /* ----- */
330:      0x88, /* #---#--- */
331:      0x88, /* #---#--- */
332:      0x50, /* -#-#--- */
333:      0x50, /* -#-#--- */
334:      0x20, /* --#----- */
335:      0x00, /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 6/70

```

336:      0x3e, /* ----###- */
337:      0x08, /* -----#- */
338:      0x08, /* -----#- */
339:      0x08, /* -----#- */
340:      0x08, /* -----#- */
341:      0x00, /* ----- */
342:      0x00, /* ----- */
343:      0x00, /* ----- */
344:      0x00, /* ----- */
345:
346:      /* 0x13 */
347:      0x00, /* ----- */
348:      0x00, /* ----- */
349:      0x00, /* ----- */
350:      0x00, /* ----- */
351:      0x0e, /* ----###- */
352:      0x38, /* ---###- */
353:      0xe0, /* ###----- */
354:      0x38, /* ---###- */
355:      0x0e, /* ----###- */
356:      0x00, /* ----- */
357:      0xfe, /* #####- */
358:      0x00, /* ----- */
359:      0x00, /* ----- */
360:      0x00, /* ----- */
361:      0x00, /* ----- */
362:      0x00, /* ----- */
363:
364:      /* 0x14 */
365:      0x00, /* ----- */
366:      0x00, /* ----- */
367:      0x00, /* ----- */
368:      0x00, /* ----- */
369:      0xe0, /* ###----- */
370:      0x38, /* ---###- */
371:      0x0e, /* ----###- */
372:      0x38, /* ---###- */
373:      0xe0, /* ###----- */
374:      0x00, /* ----- */
375:      0xfe, /* #####- */
376:      0x00, /* ----- */
377:      0x00, /* ----- */
378:      0x00, /* ----- */
379:      0x00, /* ----- */
380:      0x00, /* ----- */
381:
382:      /* 0x15 */
383:      0x00, /* ----- */
384:      0x00, /* ----- */
385:      0x00, /* ----- */
386:      0x06, /* -----##- */
387:      0x0c, /* -----##- */
388:      0xfe, /* #####- */
389:      0x18, /* ---##- */
390:      0x30, /* ---##- */
391:      0xfe, /* #####- */
392:      0x60, /* -##----- */
393:      0xc0, /* ##----- */
394:      0x00, /* ----- */
395:      0x00, /* ----- */
396:      0x00, /* ----- */
397:      0x00, /* ----- */
398:      0x00, /* ----- */
399:
400:      /* 0x16 */
401:      0x00, /* ----- */
402:      0x00, /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 7/70

```

403:      0x00, /* ----- */
404:      0x00, /* ----- */
405:      0x06, /* -----##- */
406:      0x1e, /* ----#####- */
407:      0x7e, /* -#####- */
408:      0xfe, /* #####- */
409:      0x7e, /* -#####- */
410:      0x1e, /* ----#####- */
411:      0x06, /* -----##- */
412:      0x00, /* ----- */
413:      0x00, /* ----- */
414:      0x00, /* ----- */
415:      0x00, /* ----- */
416:      0x00, /* ----- */
417:
418:          /* 0x17 */
419:      0x00, /* ----- */
420:      0x00, /* ----- */
421:      0x00, /* ----- */
422:      0x00, /* ----- */
423:      0xc0, /* ##----- */
424:      0xf0, /* #####----- */
425:      0xfc, /* #####- */
426:      0xfe, /* #####- */
427:      0xfc, /* #####- */
428:      0xf0, /* #####----- */
429:      0xc0, /* ##----- */
430:      0x00, /* ----- */
431:      0x00, /* ----- */
432:      0x00, /* ----- */
433:      0x00, /* ----- */
434:      0x00, /* ----- */
435:
436:          /* 0x18 */
437:      0x00, /* ----- */
438:      0x00, /* ----- */
439:      0x18, /* ----##---- */
440:      0x3c, /* --#####- */
441:      0x7e, /* -#####- */
442:      0x18, /* ----##---- */
443:      0x18, /* ----##---- */
444:      0x18, /* ----##---- */
445:      0x18, /* ----##---- */
446:      0x18, /* ----##---- */
447:      0x18, /* ----##---- */
448:      0x18, /* ----##---- */
449:      0x00, /* ----- */
450:      0x00, /* ----- */
451:      0x00, /* ----- */
452:      0x00, /* ----- */
453:
454:          /* 0x19 */
455:      0x00, /* ----- */
456:      0x00, /* ----- */
457:      0x18, /* ----##---- */
458:      0x18, /* ----##---- */
459:      0x18, /* ----##---- */
460:      0x18, /* ----##---- */
461:      0x18, /* ----##---- */
462:      0x18, /* ----##---- */
463:      0x18, /* ----##---- */
464:      0x7e, /* -#####- */
465:      0x3c, /* --#####- */
466:      0x18, /* ----##---- */
467:      0x00, /* ----- */
468:      0x00, /* ----- */
469:      0x00, /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 8/70

```

470:      0x00,    /* ----- */
471:
472:      /* 0x1a    */
473:      0x00,    /* ----- */
474:      0x00,    /* ----- */
475:      0x00,    /* ----- */
476:      0x00,    /* ----- */
477:      0x00,    /* ----- */
478:      0x18,    /* ---##--- */
479:      0x0c,    /* ---##--- */
480:      0xfe,    /* #####- */
481:      0x0c,    /* ---##--- */
482:      0x18,    /* ---##--- */
483:      0x00,    /* ----- */
484:      0x00,    /* ----- */
485:      0x00,    /* ----- */
486:      0x00,    /* ----- */
487:      0x00,    /* ----- */
488:      0x00,    /* ----- */
489:
490:      /* 0x1b    */
491:      0x00,    /* ----- */
492:      0x00,    /* ----- */
493:      0x00,    /* ----- */
494:      0x00,    /* ----- */
495:      0x00,    /* ----- */
496:      0x30,    /* --##---- */
497:      0x60,    /* -##----- */
498:      0xfe,    /* #####- */
499:      0x60,    /* -##----- */
500:      0x30,    /* --##---- */
501:      0x00,    /* ----- */
502:      0x00,    /* ----- */
503:      0x00,    /* ----- */
504:      0x00,    /* ----- */
505:      0x00,    /* ----- */
506:      0x00,    /* ----- */
507:
508:      /* 0x1c    */
509:      0x00,    /* ----- */
510:      0x00,    /* ----- */
511:      0x18,    /* ---##--- */
512:      0x3c,    /* ---####-- */
513:      0x7e,    /* -#####- */
514:      0x18,    /* ---##--- */
515:      0x18,    /* ---##--- */
516:      0x18,    /* ---##--- */
517:      0x18,    /* ---##--- */
518:      0x7e,    /* -#####- */
519:      0x3c,    /* ---####-- */
520:      0x18,    /* ---##--- */
521:      0x00,    /* ----- */
522:      0x00,    /* ----- */
523:      0x00,    /* ----- */
524:      0x00,    /* ----- */
525:
526:      /* 0x1d    */
527:      0x00,    /* ----- */
528:      0x00,    /* ----- */
529:      0x00,    /* ----- */
530:      0x00,    /* ----- */
531:      0x00,    /* ----- */
532:      0x28,    /* --#-#--- */
533:      0x6c,    /* -##-##--- */
534:      0xfe,    /* #####- */
535:      0x6c,    /* -##-##--- */
536:      0x28,    /* --#-#--- */

```

drivers/video/font-lat9-8x16.c

Page 9/70

```

537:      0x00, /* ----- */
538:      0x00, /* ----- */
539:      0x00, /* ----- */
540:      0x00, /* ----- */
541:      0x00, /* ----- */
542:      0x00, /* ----- */
543:
544:          /* 0x1e */
545:      0x00, /* ----- */
546:      0x00, /* ----- */
547:      0x00, /* ----- */
548:      0x00, /* ----- */
549:      0x06, /* -----##- */
550:      0x36, /* ---##-##- */
551:      0x66, /* -##-##- */
552:      0xfe, /* #####- */
553:      0x60, /* -##- */
554:      0x30, /* ---##- */
555:      0x00, /* ----- */
556:      0x00, /* ----- */
557:      0x00, /* ----- */
558:      0x00, /* ----- */
559:      0x00, /* ----- */
560:      0x00, /* ----- */
561:
562:          /* 0x1f */
563:      0x00, /* ----- */
564:      0x00, /* ----- */
565:      0x00, /* ----- */
566:      0x00, /* ----- */
567:      0x00, /* ----- */
568:      0x00, /* ----- */
569:      0xfe, /* #####- */
570:      0x6c, /* -##-##- */
571:      0x6c, /* -##-##- */
572:      0x6c, /* -##-##- */
573:      0x6c, /* -##-##- */
574:      0x6c, /* -##-##- */
575:      0x00, /* ----- */
576:      0x00, /* ----- */
577:      0x00, /* ----- */
578:      0x00, /* ----- */
579:
580:          /* 0x20 (' ') */
581:      0x00, /* ----- */
582:      0x00, /* ----- */
583:      0x00, /* ----- */
584:      0x00, /* ----- */
585:      0x00, /* ----- */
586:      0x00, /* ----- */
587:      0x00, /* ----- */
588:      0x00, /* ----- */
589:      0x00, /* ----- */
590:      0x00, /* ----- */
591:      0x00, /* ----- */
592:      0x00, /* ----- */
593:      0x00, /* ----- */
594:      0x00, /* ----- */
595:      0x00, /* ----- */
596:      0x00, /* ----- */
597:
598:          /* 0x21 ('!') */
599:      0x00, /* ----- */
600:      0x00, /* ----- */
601:      0x18, /* ---##- */
602:      0x3c, /* --####- */
603:      0x3c, /* --####- */

```

drivers/video/font-lat9-8x16.c

Page 10/70

```

604:      0x3c, /* ---###-- */
605:      0x18, /* ----##---- */
606:      0x18, /* ----##---- */
607:      0x18, /* ----##---- */
608:      0x00, /* ----- */
609:      0x18, /* ----##---- */
610:      0x18, /* ----##---- */
611:      0x00, /* ----- */
612:      0x00, /* ----- */
613:      0x00, /* ----- */
614:      0x00, /* ----- */
615:
616:          /* 0x22 ('"') */
617:      0x00, /* ----- */
618:      0x66, /* -##-##- */
619:      0x66, /* -##-##- */
620:      0x24, /* --#-#- */
621:      0x00, /* ----- */
622:      0x00, /* ----- */
623:      0x00, /* ----- */
624:      0x00, /* ----- */
625:      0x00, /* ----- */
626:      0x00, /* ----- */
627:      0x00, /* ----- */
628:      0x00, /* ----- */
629:      0x00, /* ----- */
630:      0x00, /* ----- */
631:      0x00, /* ----- */
632:      0x00, /* ----- */
633:
634:          /* 0x23 ('#') */
635:      0x00, /* ----- */
636:      0x00, /* ----- */
637:      0x00, /* ----- */
638:      0x6c, /* -##-##- */
639:      0x6c, /* -##-##- */
640:      0xfe, /* #####- */
641:      0x6c, /* -##-##- */
642:      0x6c, /* -##-##- */
643:      0x6c, /* -##-##- */
644:      0xfe, /* #####- */
645:      0x6c, /* -##-##- */
646:      0x6c, /* -##-##- */
647:      0x00, /* ----- */
648:      0x00, /* ----- */
649:      0x00, /* ----- */
650:      0x00, /* ----- */
651:
652:          /* 0x24 ('$') */
653:      0x00, /* ----- */
654:      0x10, /* ----#---- */
655:      0x10, /* ----#---- */
656:      0x7c, /* -#####- */
657:      0xd6, /* ##-#-##- */
658:      0xd0, /* ##-#-##- */
659:      0xd0, /* ##-#-##- */
660:      0x7c, /* -#####- */
661:      0x16, /* ---#-##- */
662:      0x16, /* ---#-##- */
663:      0xd6, /* ##-#-##- */
664:      0x7c, /* -#####- */
665:      0x10, /* ----#---- */
666:      0x10, /* ----#---- */
667:      0x00, /* ----- */
668:      0x00, /* ----- */
669:
670:          /* 0x25 ('%') */

```


drivers/video/font-lat9-8x16.c

Page 11/70

```

671:      0x00, /* ----- */
672:      0x00, /* ----- */
673:      0x00, /* ----- */
674:      0x00, /* ----- */
675:      0xc2, /* ##----# */
676:      0xc6, /* ##---## */
677:      0x0c, /* ----##-- */
678:      0x18, /* ----##-- */
679:      0x30, /* --##----- */
680:      0x60, /* -##----- */
681:      0xc6, /* ##---## */
682:      0x86, /* #----##- */
683:      0x00, /* ----- */
684:      0x00, /* ----- */
685:      0x00, /* ----- */
686:      0x00, /* ----- */
687:
688:      /* 0x26 ('&') */
689:      0x00, /* ----- */
690:      0x00, /* ----- */
691:      0x38, /* --###---- */
692:      0x6c, /* -##-##-- */
693:      0x6c, /* -##-##-- */
694:      0x38, /* --###---- */
695:      0x76, /* -##-##- */
696:      0xdc, /* ##-##-#- */
697:      0xcc, /* ##--##-- */
698:      0xcc, /* ##--##-- */
699:      0xcc, /* ##--##-- */
700:      0x76, /* -##-##- */
701:      0x00, /* ----- */
702:      0x00, /* ----- */
703:      0x00, /* ----- */
704:      0x00, /* ----- */
705:
706:      /* 0x27 ('''') */
707:      0x00, /* ----- */
708:      0x18, /* ----##-- */
709:      0x18, /* ----##-- */
710:      0x18, /* ----##-- */
711:      0x30, /* --##----- */
712:      0x00, /* ----- */
713:      0x00, /* ----- */
714:      0x00, /* ----- */
715:      0x00, /* ----- */
716:      0x00, /* ----- */
717:      0x00, /* ----- */
718:      0x00, /* ----- */
719:      0x00, /* ----- */
720:      0x00, /* ----- */
721:      0x00, /* ----- */
722:      0x00, /* ----- */
723:
724:      /* 0x28 ('(') */
725:      0x00, /* ----- */
726:      0x00, /* ----- */
727:      0x0c, /* ----##-- */
728:      0x18, /* ----##-- */
729:      0x30, /* --##----- */
730:      0x30, /* --##----- */
731:      0x30, /* --##----- */
732:      0x30, /* --##----- */
733:      0x30, /* --##----- */
734:      0x30, /* --##----- */
735:      0x18, /* ----##-- */
736:      0x0c, /* ----##-- */
737:      0x00, /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 12/70

```

738:      0x00, /* ----- */
739:      0x00, /* ----- */
740:      0x00, /* ----- */
741:
742:      /* 0x29 ('') */
743:      0x00, /* ----- */
744:      0x00, /* ----- */
745:      0x30, /* --##----- */
746:      0x18, /* ---##----- */
747:      0x0c, /* ----##----- */
748:      0x0c, /* ----##----- */
749:      0x0c, /* ----##----- */
750:      0x0c, /* ----##----- */
751:      0x0c, /* ----##----- */
752:      0x0c, /* ----##----- */
753:      0x18, /* ---##----- */
754:      0x30, /* --##----- */
755:      0x00, /* ----- */
756:      0x00, /* ----- */
757:      0x00, /* ----- */
758:      0x00, /* ----- */
759:
760:      /* 0x2a ('*') */
761:      0x00, /* ----- */
762:      0x00, /* ----- */
763:      0x00, /* ----- */
764:      0x00, /* ----- */
765:      0x00, /* ----- */
766:      0x66, /* -##-##----- */
767:      0x3c, /* --####----- */
768:      0xff, /* #####----- */
769:      0x3c, /* --####----- */
770:      0x66, /* -##-##----- */
771:      0x00, /* ----- */
772:      0x00, /* ----- */
773:      0x00, /* ----- */
774:      0x00, /* ----- */
775:      0x00, /* ----- */
776:      0x00, /* ----- */
777:
778:      /* 0x2b ('+') */
779:      0x00, /* ----- */
780:      0x00, /* ----- */
781:      0x00, /* ----- */
782:      0x00, /* ----- */
783:      0x00, /* ----- */
784:      0x18, /* ---##----- */
785:      0x18, /* ---##----- */
786:      0x7e, /* -#####----- */
787:      0x18, /* ---##----- */
788:      0x18, /* ---##----- */
789:      0x00, /* ----- */
790:      0x00, /* ----- */
791:      0x00, /* ----- */
792:      0x00, /* ----- */
793:      0x00, /* ----- */
794:      0x00, /* ----- */
795:
796:      /* 0x2c (',') */
797:      0x00, /* ----- */
798:      0x00, /* ----- */
799:      0x00, /* ----- */
800:      0x00, /* ----- */
801:      0x00, /* ----- */
802:      0x00, /* ----- */
803:      0x00, /* ----- */
804:      0x00, /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 13/70

```

805:      0x00, /* ----- */
806:      0x18, /* ---##--- */
807:      0x18, /* ---##--- */
808:      0x18, /* ---##--- */
809:      0x30, /* --##---- */
810:      0x00, /* ----- */
811:      0x00, /* ----- */
812:      0x00, /* ----- */
813:
814:          /* 0x2d ('-') */
815:      0x00, /* ----- */
816:      0x00, /* ----- */
817:      0x00, /* ----- */
818:      0x00, /* ----- */
819:      0x00, /* ----- */
820:      0x00, /* ----- */
821:      0x00, /* ----- */
822:      0xfe, /* #####- */
823:      0x00, /* ----- */
824:      0x00, /* ----- */
825:      0x00, /* ----- */
826:      0x00, /* ----- */
827:      0x00, /* ----- */
828:      0x00, /* ----- */
829:      0x00, /* ----- */
830:      0x00, /* ----- */
831:
832:          /* 0x2e ('.') */
833:      0x00, /* ----- */
834:      0x00, /* ----- */
835:      0x00, /* ----- */
836:      0x00, /* ----- */
837:      0x00, /* ----- */
838:      0x00, /* ----- */
839:      0x00, /* ----- */
840:      0x00, /* ----- */
841:      0x00, /* ----- */
842:      0x00, /* ----- */
843:      0x18, /* ---##--- */
844:      0x18, /* ---##--- */
845:      0x00, /* ----- */
846:      0x00, /* ----- */
847:      0x00, /* ----- */
848:      0x00, /* ----- */
849:
850:          /* 0x2f ('/') */
851:      0x00, /* ----- */
852:      0x00, /* ----- */
853:      0x00, /* ----- */
854:      0x00, /* ----- */
855:      0x00, /* ----- */
856:      0x06, /* -----##- */
857:      0x0c, /* -----##- */
858:      0x18, /* ---##--- */
859:      0x30, /* --##---- */
860:      0x60, /* -##----- */
861:      0xc0, /* ##----- */
862:      0x00, /* ----- */
863:      0x00, /* ----- */
864:      0x00, /* ----- */
865:      0x00, /* ----- */
866:      0x00, /* ----- */
867:
868:          /* 0x30 ('0') */
869:      0x00, /* ----- */
870:      0x00, /* ----- */
871:      0x7c, /* -#####- */

```

drivers/video/font-lat9-8x16.c

Page 14/70

```

872:      0xc6, /* ##---##- */
873:      0xc6, /* ##---##- */
874:      0xc6, /* ##---##- */
875:      0xd6, /* ##-##-##- */
876:      0xd6, /* ##-##-##- */
877:      0xc6, /* ##---##- */
878:      0xc6, /* ##---##- */
879:      0xc6, /* ##---##- */
880:      0x7c, /* -#####- */
881:      0x00, /* ----- */
882:      0x00, /* ----- */
883:      0x00, /* ----- */
884:      0x00, /* ----- */
885:
886:          /* 0x31 ('1') */
887:      0x00, /* ----- */
888:      0x00, /* ----- */
889:      0x18, /* ---##--- */
890:      0x38, /* --###--- */
891:      0x78, /* -####--- */
892:      0x18, /* ---##--- */
893:      0x18, /* ---##--- */
894:      0x18, /* ---##--- */
895:      0x18, /* ---##--- */
896:      0x18, /* ---##--- */
897:      0x18, /* ---##--- */
898:      0x7e, /* -#####- */
899:      0x00, /* ----- */
900:      0x00, /* ----- */
901:      0x00, /* ----- */
902:      0x00, /* ----- */
903:
904:          /* 0x32 ('2') */
905:      0x00, /* ----- */
906:      0x00, /* ----- */
907:      0x7c, /* -#####- */
908:      0xc6, /* ##---##- */
909:      0x06, /* -----##- */
910:      0x0c, /* -----##- */
911:      0x18, /* ---##--- */
912:      0x30, /* --##----- */
913:      0x60, /* -##----- */
914:      0xc0, /* ##----- */
915:      0xc6, /* ##---##- */
916:      0xfe, /* #####-#- */
917:      0x00, /* ----- */
918:      0x00, /* ----- */
919:      0x00, /* ----- */
920:      0x00, /* ----- */
921:
922:          /* 0x33 ('3') */
923:      0x00, /* ----- */
924:      0x00, /* ----- */
925:      0x7c, /* -#####- */
926:      0xc6, /* ##---##- */
927:      0x06, /* -----##- */
928:      0x06, /* -----##- */
929:      0x3c, /* --####--- */
930:      0x06, /* -----##- */
931:      0x06, /* -----##- */
932:      0x06, /* -----##- */
933:      0xc6, /* ##---##- */
934:      0x7c, /* -#####- */
935:      0x00, /* ----- */
936:      0x00, /* ----- */
937:      0x00, /* ----- */
938:      0x00, /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 15/70

```

939:
940:          /* 0x34 ('4') */
941:    0x00,  /* ----- */
942:    0x00,  /* ----- */
943:    0x0c,  /* ----##-- */
944:    0x1c,  /* ---###-- */
945:    0x3c,  /* --####-- */
946:    0x6c,  /* -##-##-- */
947:    0xc0,  /* ##--##-- */
948:    0xfe,  /* #####-- */
949:    0x0c,  /* ----##-- */
950:    0x0c,  /* ----##-- */
951:    0x0c,  /* ----##-- */
952:    0x1e,  /* ---####-- */
953:    0x00,  /* ----- */
954:    0x00,  /* ----- */
955:    0x00,  /* ----- */
956:    0x00,  /* ----- */
957:
958:          /* 0x35 ('5') */
959:    0x00,  /* ----- */
960:    0x00,  /* ----- */
961:    0xfe,  /* #####-- */
962:    0xc0,  /* ##----- */
963:    0xc0,  /* ##----- */
964:    0xc0,  /* ##----- */
965:    0xfc,  /* #####-- */
966:    0x06,  /* -----##- */
967:    0x06,  /* -----##- */
968:    0x06,  /* -----##- */
969:    0xc6,  /* ##----##- */
970:    0x7c,  /* -#####-- */
971:    0x00,  /* ----- */
972:    0x00,  /* ----- */
973:    0x00,  /* ----- */
974:    0x00,  /* ----- */
975:
976:          /* 0x36 ('6') */
977:    0x00,  /* ----- */
978:    0x00,  /* ----- */
979:    0x38,  /* ---###-- */
980:    0x60,  /* -##----- */
981:    0xc0,  /* ##----- */
982:    0xc0,  /* ##----- */
983:    0xfc,  /* #####-- */
984:    0xc6,  /* ##----##- */
985:    0xc6,  /* ##----##- */
986:    0xc6,  /* ##----##- */
987:    0xc6,  /* ##----##- */
988:    0x7c,  /* -#####-- */
989:    0x00,  /* ----- */
990:    0x00,  /* ----- */
991:    0x00,  /* ----- */
992:    0x00,  /* ----- */
993:
994:          /* 0x37 ('7') */
995:    0x00,  /* ----- */
996:    0x00,  /* ----- */
997:    0xfe,  /* #####-- */
998:    0xc6,  /* ##----##- */
999:    0x06,  /* -----##- */
1000:   0x06,  /* -----##- */
1001:   0x0c,  /* ----##-- */
1002:   0x18,  /* ---###-- */
1003:   0x30,  /* --####-- */
1004:   0x30,  /* --####-- */
1005:   0x30,  /* --####-- */

```

drivers/video/font-lat9-8x16.c

Page 16/70

```

1006:      0x30,    /* --##----- */
1007:      0x00,    /* ----- */
1008:      0x00,    /* ----- */
1009:      0x00,    /* ----- */
1010:      0x00,    /* ----- */
1011:
1012:          /* 0x38 ('8') */
1013:      0x00,    /* ----- */
1014:      0x00,    /* ----- */
1015:      0x7c,    /* -#####- */
1016:      0xc6,    /* ##---##- */
1017:      0xc6,    /* ##---##- */
1018:      0xc6,    /* ##---##- */
1019:      0x7c,    /* -#####- */
1020:      0xc6,    /* ##---##- */
1021:      0xc6,    /* ##---##- */
1022:      0xc6,    /* ##---##- */
1023:      0xc6,    /* ##---##- */
1024:      0x7c,    /* -#####- */
1025:      0x00,    /* ----- */
1026:      0x00,    /* ----- */
1027:      0x00,    /* ----- */
1028:      0x00,    /* ----- */
1029:
1030:          /* 0x39 ('9') */
1031:      0x00,    /* ----- */
1032:      0x00,    /* ----- */
1033:      0x7c,    /* -#####- */
1034:      0xc6,    /* ##---##- */
1035:      0xc6,    /* ##---##- */
1036:      0xc6,    /* ##---##- */
1037:      0x7e,    /* -#####- */
1038:      0x06,    /* -----##- */
1039:      0x06,    /* -----##- */
1040:      0x06,    /* -----##- */
1041:      0x0c,    /* ----##-- */
1042:      0x78,    /* -###--- */
1043:      0x00,    /* ----- */
1044:      0x00,    /* ----- */
1045:      0x00,    /* ----- */
1046:      0x00,    /* ----- */
1047:
1048:          /* 0x3a (':') */
1049:      0x00,    /* ----- */
1050:      0x00,    /* ----- */
1051:      0x00,    /* ----- */
1052:      0x00,    /* ----- */
1053:      0x18,    /* ---##--- */
1054:      0x18,    /* ---##--- */
1055:      0x00,    /* ----- */
1056:      0x00,    /* ----- */
1057:      0x00,    /* ----- */
1058:      0x18,    /* ---##--- */
1059:      0x18,    /* ---##--- */
1060:      0x00,    /* ----- */
1061:      0x00,    /* ----- */
1062:      0x00,    /* ----- */
1063:      0x00,    /* ----- */
1064:      0x00,    /* ----- */
1065:
1066:          /* 0x3b (';') */
1067:      0x00,    /* ----- */
1068:      0x00,    /* ----- */
1069:      0x00,    /* ----- */
1070:      0x00,    /* ----- */
1071:      0x18,    /* ---##--- */
1072:      0x18,    /* ---##--- */

```

drivers/video/font-lat9-8x16.c

Page 17/70

```

1073:      0x00, /* ----- */
1074:      0x00, /* ----- */
1075:      0x00, /* ----- */
1076:      0x18, /* ---##--- */
1077:      0x18, /* ---##--- */
1078:      0x30, /* --##---- */
1079:      0x00, /* ----- */
1080:      0x00, /* ----- */
1081:      0x00, /* ----- */
1082:      0x00, /* ----- */
1083:
1084:          /* 0x3c ('<') */
1085:      0x00, /* ----- */
1086:      0x00, /* ----- */
1087:      0x00, /* ----- */
1088:      0x06, /* -----##- */
1089:      0x0c, /* -----##- */
1090:      0x18, /* ---##--- */
1091:      0x30, /* --##---- */
1092:      0x60, /* -##----- */
1093:      0x30, /* --##---- */
1094:      0x18, /* ---##--- */
1095:      0x0c, /* -----##- */
1096:      0x06, /* -----##- */
1097:      0x00, /* ----- */
1098:      0x00, /* ----- */
1099:      0x00, /* ----- */
1100:      0x00, /* ----- */
1101:
1102:          /* 0x3d ('=') */
1103:      0x00, /* ----- */
1104:      0x00, /* ----- */
1105:      0x00, /* ----- */
1106:      0x00, /* ----- */
1107:      0x00, /* ----- */
1108:      0xfe, /* #####- */
1109:      0x00, /* ----- */
1110:      0x00, /* ----- */
1111:      0xfe, /* #####- */
1112:      0x00, /* ----- */
1113:      0x00, /* ----- */
1114:      0x00, /* ----- */
1115:      0x00, /* ----- */
1116:      0x00, /* ----- */
1117:      0x00, /* ----- */
1118:      0x00, /* ----- */
1119:
1120:          /* 0x3e ('>') */
1121:      0x00, /* ----- */
1122:      0x00, /* ----- */
1123:      0x00, /* ----- */
1124:      0x60, /* -##----- */
1125:      0x30, /* --##---- */
1126:      0x18, /* ---##--- */
1127:      0x0c, /* -----##- */
1128:      0x06, /* -----##- */
1129:      0x0c, /* -----##- */
1130:      0x18, /* ---##--- */
1131:      0x30, /* --##---- */
1132:      0x60, /* -##----- */
1133:      0x00, /* ----- */
1134:      0x00, /* ----- */
1135:      0x00, /* ----- */
1136:      0x00, /* ----- */
1137:
1138:          /* 0x3f ('?') */
1139:      0x00, /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 18/70

```

1140:      0x00, /* ----- */
1141:      0x7c, /* -#####- */
1142:      0xc6, /* ##---##- */
1143:      0xc6, /* ##---##- */
1144:      0x0c, /* ----##- */
1145:      0x18, /* ---##--- */
1146:      0x18, /* ---##--- */
1147:      0x18, /* ---##--- */
1148:      0x00, /* ----- */
1149:      0x18, /* ---##--- */
1150:      0x18, /* ---##--- */
1151:      0x00, /* ----- */
1152:      0x00, /* ----- */
1153:      0x00, /* ----- */
1154:      0x00, /* ----- */
1155:
1156:          /* 0x40 ('@') */
1157:      0x00, /* ----- */
1158:      0x00, /* ----- */
1159:      0x7c, /* -#####- */
1160:      0xc6, /* ##---##- */
1161:      0xc6, /* ##---##- */
1162:      0xc6, /* ##---##- */
1163:      0xde, /* ##-####- */
1164:      0xde, /* ##-####- */
1165:      0xde, /* ##-####- */
1166:      0xdc, /* ##-####- */
1167:      0xc0, /* ##----- */
1168:      0x7c, /* -#####- */
1169:      0x00, /* ----- */
1170:      0x00, /* ----- */
1171:      0x00, /* ----- */
1172:      0x00, /* ----- */
1173:
1174:          /* 0x41 ('A') */
1175:      0x00, /* ----- */
1176:      0x00, /* ----- */
1177:      0x10, /* ---#---- */
1178:      0x38, /* --###--- */
1179:      0x6c, /* -##-##- */
1180:      0xc6, /* ##---##- */
1181:      0xc6, /* ##---##- */
1182:      0xfe, /* #####- */
1183:      0xc6, /* ##---##- */
1184:      0xc6, /* ##---##- */
1185:      0xc6, /* ##---##- */
1186:      0xc6, /* ##---##- */
1187:      0x00, /* ----- */
1188:      0x00, /* ----- */
1189:      0x00, /* ----- */
1190:      0x00, /* ----- */
1191:
1192:          /* 0x42 ('B') */
1193:      0x00, /* ----- */
1194:      0x00, /* ----- */
1195:      0xfc, /* #####- */
1196:      0x66, /* -##-##- */
1197:      0x66, /* -##-##- */
1198:      0x66, /* -##-##- */
1199:      0x7c, /* -#####- */
1200:      0x66, /* -##-##- */
1201:      0x66, /* -##-##- */
1202:      0x66, /* -##-##- */
1203:      0x66, /* -##-##- */
1204:      0xfc, /* #####- */
1205:      0x00, /* ----- */
1206:      0x00, /* ----- */

```


drivers/video/font-lat9-8x16.c

Page 19/70

```

1207:      0x00, /* ----- */
1208:      0x00, /* ----- */
1209:
1210:          /* 0x43 ('C') */
1211:      0x00, /* ----- */
1212:      0x00, /* ----- */
1213:      0x3c, /* ---###-- */
1214:      0x66, /* -##--##- */
1215:      0xc2, /* ##-----#- */
1216:      0xc0, /* ##----- */
1217:      0xc0, /* ##----- */
1218:      0xc0, /* ##----- */
1219:      0xc0, /* ##----- */
1220:      0xc2, /* ##-----#- */
1221:      0x66, /* -##--##- */
1222:      0x3c, /* ---###-- */
1223:      0x00, /* ----- */
1224:      0x00, /* ----- */
1225:      0x00, /* ----- */
1226:      0x00, /* ----- */
1227:
1228:          /* 0x44 ('D') */
1229:      0x00, /* ----- */
1230:      0x00, /* ----- */
1231:      0xf8, /* #####--- */
1232:      0x6c, /* -##-##-- */
1233:      0x66, /* -##--##- */
1234:      0x66, /* -##--##- */
1235:      0x66, /* -##--##- */
1236:      0x66, /* -##--##- */
1237:      0x66, /* -##--##- */
1238:      0x66, /* -##--##- */
1239:      0x6c, /* -##-##-- */
1240:      0xf8, /* #####--- */
1241:      0x00, /* ----- */
1242:      0x00, /* ----- */
1243:      0x00, /* ----- */
1244:      0x00, /* ----- */
1245:
1246:          /* 0x45 ('E') */
1247:      0x00, /* ----- */
1248:      0x00, /* ----- */
1249:      0xfe, /* #####--- */
1250:      0x66, /* -##--##- */
1251:      0x62, /* -##--##- */
1252:      0x68, /* -##-##-- */
1253:      0x78, /* -####--- */
1254:      0x68, /* -##-##-- */
1255:      0x60, /* -##----- */
1256:      0x62, /* -##--##- */
1257:      0x66, /* -##--##- */
1258:      0xfe, /* #####--- */
1259:      0x00, /* ----- */
1260:      0x00, /* ----- */
1261:      0x00, /* ----- */
1262:      0x00, /* ----- */
1263:
1264:          /* 0x46 ('F') */
1265:      0x00, /* ----- */
1266:      0x00, /* ----- */
1267:      0xfe, /* #####--- */
1268:      0x66, /* -##--##- */
1269:      0x62, /* -##--##- */
1270:      0x68, /* -##-##-- */
1271:      0x78, /* -####--- */
1272:      0x68, /* -##-##-- */
1273:      0x60, /* -##----- */

```

drivers/video/font-lat9-8x16.c

Page 20/70

```

1274:      0x60,    /* -##----- */
1275:      0x60,    /* -##----- */
1276:      0xf0,    /* ####----- */
1277:      0x00,    /* ----- */
1278:      0x00,    /* ----- */
1279:      0x00,    /* ----- */
1280:      0x00,    /* ----- */
1281:
1282:          /* 0x47 ('G') */
1283:      0x00,    /* ----- */
1284:      0x00,    /* ----- */
1285:      0x3c,    /* --####-- */
1286:      0x66,    /* -##--##- */
1287:      0xc2,    /* ##-----#- */
1288:      0xc0,    /* ##----- */
1289:      0xc0,    /* ##----- */
1290:      0xde,    /* ##-####- */
1291:      0xc6,    /* ##---##- */
1292:      0xc6,    /* ##---##- */
1293:      0x66,    /* -##---##- */
1294:      0x3a,    /* --####-#- */
1295:      0x00,    /* ----- */
1296:      0x00,    /* ----- */
1297:      0x00,    /* ----- */
1298:      0x00,    /* ----- */
1299:
1300:          /* 0x48 ('H') */
1301:      0x00,    /* ----- */
1302:      0x00,    /* ----- */
1303:      0xc6,    /* ##---##- */
1304:      0xc6,    /* ##---##- */
1305:      0xc6,    /* ##---##- */
1306:      0xc6,    /* ##---##- */
1307:      0xfe,    /* #####---#- */
1308:      0xc6,    /* ##---##- */
1309:      0xc6,    /* ##---##- */
1310:      0xc6,    /* ##---##- */
1311:      0xc6,    /* ##---##- */
1312:      0xc6,    /* ##---##- */
1313:      0x00,    /* ----- */
1314:      0x00,    /* ----- */
1315:      0x00,    /* ----- */
1316:      0x00,    /* ----- */
1317:
1318:          /* 0x49 ('I') */
1319:      0x00,    /* ----- */
1320:      0x00,    /* ----- */
1321:      0x3c,    /* --####-- */
1322:      0x18,    /* ---##--- */
1323:      0x18,    /* ---##--- */
1324:      0x18,    /* ---##--- */
1325:      0x18,    /* ---##--- */
1326:      0x18,    /* ---##--- */
1327:      0x18,    /* ---##--- */
1328:      0x18,    /* ---##--- */
1329:      0x18,    /* ---##--- */
1330:      0x3c,    /* --####-- */
1331:      0x00,    /* ----- */
1332:      0x00,    /* ----- */
1333:      0x00,    /* ----- */
1334:      0x00,    /* ----- */
1335:
1336:          /* 0x4a ('J') */
1337:      0x00,    /* ----- */
1338:      0x00,    /* ----- */
1339:      0x1e,    /* ---####- */
1340:      0x0c,    /* ----##-- */

```

drivers/video/font-lat9-8x16.c

Page 21/70

```

1341:      0x0c, /* ----##-- */
1342:      0x0c, /* ----##-- */
1343:      0x0c, /* ----##-- */
1344:      0x0c, /* ----##-- */
1345:      0xc c, /* ##--##-- */
1346:      0xc c, /* ##--##-- */
1347:      0xc c, /* ##--##-- */
1348:      0x78, /* -####-- */
1349:      0x00, /* ----- */
1350:      0x00, /* ----- */
1351:      0x00, /* ----- */
1352:      0x00, /* ----- */
1353:
1354:          /* 0x4b ('K') */
1355:      0x00, /* ----- */
1356:      0x00, /* ----- */
1357:      0xe6, /* ###--##- */
1358:      0x66, /* -##--##- */
1359:      0x66, /* -##--##- */
1360:      0x6c, /* -##-##-- */
1361:      0x78, /* -####-- */
1362:      0x78, /* -####-- */
1363:      0x6c, /* -##-##-- */
1364:      0x66, /* -##--##- */
1365:      0x66, /* -##--##- */
1366:      0xe6, /* ###--##- */
1367:      0x00, /* ----- */
1368:      0x00, /* ----- */
1369:      0x00, /* ----- */
1370:      0x00, /* ----- */
1371:
1372:          /* 0x4c ('L') */
1373:      0x00, /* ----- */
1374:      0x00, /* ----- */
1375:      0xf0, /* #####-- */
1376:      0x60, /* -##----- */
1377:      0x60, /* -##----- */
1378:      0x60, /* -##----- */
1379:      0x60, /* -##----- */
1380:      0x60, /* -##----- */
1381:      0x60, /* -##----- */
1382:      0x62, /* -##---#- */
1383:      0x66, /* -##--##- */
1384:      0xfe, /* #####--- */
1385:      0x00, /* ----- */
1386:      0x00, /* ----- */
1387:      0x00, /* ----- */
1388:      0x00, /* ----- */
1389:
1390:          /* 0x4d ('M') */
1391:      0x00, /* ----- */
1392:      0x00, /* ----- */
1393:      0xc6, /* ##---##- */
1394:      0xee, /* ###-###- */
1395:      0xfe, /* #####--- */
1396:      0xfe, /* #####--- */
1397:      0xd6, /* ##-#-##- */
1398:      0xc6, /* ##---##- */
1399:      0xc6, /* ##---##- */
1400:      0xc6, /* ##---##- */
1401:      0xc6, /* ##---##- */
1402:      0xc6, /* ##---##- */
1403:      0x00, /* ----- */
1404:      0x00, /* ----- */
1405:      0x00, /* ----- */
1406:      0x00, /* ----- */
1407:

```

drivers/video/font-lat9-8x16.c

Page 22/70

```

1408:                /* 0x4e ('N') */
1409:                0x00, /* ----- */
1410:                0x00, /* ----- */
1411:                0xc6, /* ##---##- */
1412:                0xe6, /* ###---##- */
1413:                0xf6, /* ####---##- */
1414:                0xfe, /* #####--- */
1415:                0xde, /* ##-####- */
1416:                0xce, /* ##---##- */
1417:                0xc6, /* ##---##- */
1418:                0xc6, /* ##---##- */
1419:                0xc6, /* ##---##- */
1420:                0xc6, /* ##---##- */
1421:                0x00, /* ----- */
1422:                0x00, /* ----- */
1423:                0x00, /* ----- */
1424:                0x00, /* ----- */
1425:
1426:                /* 0x4f ('O') */
1427:                0x00, /* ----- */
1428:                0x00, /* ----- */
1429:                0x7c, /* -#####- */
1430:                0xc6, /* ##---##- */
1431:                0xc6, /* ##---##- */
1432:                0xc6, /* ##---##- */
1433:                0xc6, /* ##---##- */
1434:                0xc6, /* ##---##- */
1435:                0xc6, /* ##---##- */
1436:                0xc6, /* ##---##- */
1437:                0xc6, /* ##---##- */
1438:                0x7c, /* -#####- */
1439:                0x00, /* ----- */
1440:                0x00, /* ----- */
1441:                0x00, /* ----- */
1442:                0x00, /* ----- */
1443:
1444:                /* 0x50 ('P') */
1445:                0x00, /* ----- */
1446:                0x00, /* ----- */
1447:                0xfc, /* #####--- */
1448:                0x66, /* -##---##- */
1449:                0x66, /* -##---##- */
1450:                0x66, /* -##---##- */
1451:                0x7c, /* -#####- */
1452:                0x60, /* -##----- */
1453:                0x60, /* -##----- */
1454:                0x60, /* -##----- */
1455:                0x60, /* -##----- */
1456:                0xf0, /* #####----- */
1457:                0x00, /* ----- */
1458:                0x00, /* ----- */
1459:                0x00, /* ----- */
1460:                0x00, /* ----- */
1461:
1462:                /* 0x51 ('Q') */
1463:                0x00, /* ----- */
1464:                0x00, /* ----- */
1465:                0x7c, /* -#####- */
1466:                0xc6, /* ##---##- */
1467:                0xc6, /* ##---##- */
1468:                0xc6, /* ##---##- */
1469:                0xc6, /* ##---##- */
1470:                0xc6, /* ##---##- */
1471:                0xc6, /* ##---##- */
1472:                0xd6, /* ##-##--- */
1473:                0xde, /* ##-####- */
1474:                0x7c, /* -#####- */

```

drivers/video/font-lat9-8x16.c

Page 23/70

```

1475:      0x0c, /* ----##-- */
1476:      0x0e, /* ----###- */
1477:      0x00, /* ----- */
1478:      0x00, /* ----- */
1479:
1480:          /* 0x52 ('R') */
1481:      0x00, /* ----- */
1482:      0x00, /* ----- */
1483:      0xf0, /* #####-- */
1484:      0x66, /* -##-##- */
1485:      0x66, /* -##-##- */
1486:      0x66, /* -##-##- */
1487:      0x7c, /* -#####- */
1488:      0x6c, /* -##-##-- */
1489:      0x66, /* -##-##- */
1490:      0x66, /* -##-##- */
1491:      0x66, /* -##-##- */
1492:      0xe6, /* ###-##- */
1493:      0x00, /* ----- */
1494:      0x00, /* ----- */
1495:      0x00, /* ----- */
1496:      0x00, /* ----- */
1497:
1498:          /* 0x53 ('S') */
1499:      0x00, /* ----- */
1500:      0x00, /* ----- */
1501:      0x7c, /* -#####- */
1502:      0xc6, /* ##---##- */
1503:      0xc6, /* ##---##- */
1504:      0x64, /* -##-##-- */
1505:      0x38, /* --###--- */
1506:      0x0c, /* ----##-- */
1507:      0x06, /* -----##- */
1508:      0xc6, /* ##---##- */
1509:      0xc6, /* ##---##- */
1510:      0x7c, /* -#####- */
1511:      0x00, /* ----- */
1512:      0x00, /* ----- */
1513:      0x00, /* ----- */
1514:      0x00, /* ----- */
1515:
1516:          /* 0x54 ('T') */
1517:      0x00, /* ----- */
1518:      0x00, /* ----- */
1519:      0x7e, /* -#####- */
1520:      0x7e, /* -#####- */
1521:      0x5a, /* -#-##-##- */
1522:      0x18, /* ---##--- */
1523:      0x18, /* ---##--- */
1524:      0x18, /* ---##--- */
1525:      0x18, /* ---##--- */
1526:      0x18, /* ---##--- */
1527:      0x18, /* ---##--- */
1528:      0x3c, /* --####-- */
1529:      0x00, /* ----- */
1530:      0x00, /* ----- */
1531:      0x00, /* ----- */
1532:      0x00, /* ----- */
1533:
1534:          /* 0x55 ('U') */
1535:      0x00, /* ----- */
1536:      0x00, /* ----- */
1537:      0xc6, /* ##---##- */
1538:      0xc6, /* ##---##- */
1539:      0xc6, /* ##---##- */
1540:      0xc6, /* ##---##- */
1541:      0xc6, /* ##---##- */

```

drivers/video/font-lat9-8x16.c

Page 24/70

```

1542:      0xc6, /* ##---##- */
1543:      0xc6, /* ##---##- */
1544:      0xc6, /* ##---##- */
1545:      0xc6, /* ##---##- */
1546:      0x7c, /* -#####- */
1547:      0x00, /* ----- */
1548:      0x00, /* ----- */
1549:      0x00, /* ----- */
1550:      0x00, /* ----- */
1551:
1552:          /* 0x56 ('V') */
1553:      0x00, /* ----- */
1554:      0x00, /* ----- */
1555:      0xc6, /* ##---##- */
1556:      0xc6, /* ##---##- */
1557:      0xc6, /* ##---##- */
1558:      0xc6, /* ##---##- */
1559:      0xc6, /* ##---##- */
1560:      0xc6, /* ##---##- */
1561:      0xc6, /* ##---##- */
1562:      0x6c, /* -##-##- */
1563:      0x38, /* --###--- */
1564:      0x10, /* ---#---- */
1565:      0x00, /* ----- */
1566:      0x00, /* ----- */
1567:      0x00, /* ----- */
1568:      0x00, /* ----- */
1569:
1570:          /* 0x57 ('W') */
1571:      0x00, /* ----- */
1572:      0x00, /* ----- */
1573:      0xc6, /* ##---##- */
1574:      0xc6, /* ##---##- */
1575:      0xc6, /* ##---##- */
1576:      0xc6, /* ##---##- */
1577:      0xd6, /* ##-##-##- */
1578:      0xd6, /* ##-##-##- */
1579:      0xd6, /* ##-##-##- */
1580:      0xfe, /* #####- */
1581:      0xee, /* ###-###- */
1582:      0x6c, /* -##-##- */
1583:      0x00, /* ----- */
1584:      0x00, /* ----- */
1585:      0x00, /* ----- */
1586:      0x00, /* ----- */
1587:
1588:          /* 0x58 ('X') */
1589:      0x00, /* ----- */
1590:      0x00, /* ----- */
1591:      0xc6, /* ##---##- */
1592:      0xc6, /* ##---##- */
1593:      0x6c, /* -##-##- */
1594:      0x7c, /* -#####- */
1595:      0x38, /* --###--- */
1596:      0x38, /* --###--- */
1597:      0x7c, /* -#####- */
1598:      0x6c, /* -##-##- */
1599:      0xc6, /* ##---##- */
1600:      0xc6, /* ##---##- */
1601:      0x00, /* ----- */
1602:      0x00, /* ----- */
1603:      0x00, /* ----- */
1604:      0x00, /* ----- */
1605:
1606:          /* 0x59 ('Y') */
1607:      0x00, /* ----- */
1608:      0x00, /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 25/70

```

1609:      0x66,    /* --##--##- */
1610:      0x66,    /* --##--##- */
1611:      0x66,    /* --##--##- */
1612:      0x66,    /* --##--##- */
1613:      0x3c,    /* ---#####- */
1614:      0x18,    /* ----##---- */
1615:      0x18,    /* ----##---- */
1616:      0x18,    /* ----##---- */
1617:      0x18,    /* ----##---- */
1618:      0x3c,    /* ---#####- */
1619:      0x00,    /* ----- */
1620:      0x00,    /* ----- */
1621:      0x00,    /* ----- */
1622:      0x00,    /* ----- */
1623:
1624:          /* 0x5a ('Z') */
1625:      0x00,    /* ----- */
1626:      0x00,    /* ----- */
1627:      0xfe,    /* #####- */
1628:      0xc6,    /* ##----##- */
1629:      0x86,    /* #----##- */
1630:      0x0c,    /* ----##-- */
1631:      0x18,    /* ----##---- */
1632:      0x30,    /* --##---- */
1633:      0x60,    /* -##----- */
1634:      0xc2,    /* ##-----#- */
1635:      0xc6,    /* ##----##- */
1636:      0xfe,    /* #####- */
1637:      0x00,    /* ----- */
1638:      0x00,    /* ----- */
1639:      0x00,    /* ----- */
1640:      0x00,    /* ----- */
1641:
1642:          /* 0x5b ('[') */
1643:      0x00,    /* ----- */
1644:      0x00,    /* ----- */
1645:      0x3c,    /* ---#####- */
1646:      0x30,    /* --##---- */
1647:      0x30,    /* --##---- */
1648:      0x30,    /* --##---- */
1649:      0x30,    /* --##---- */
1650:      0x30,    /* --##---- */
1651:      0x30,    /* --##---- */
1652:      0x30,    /* --##---- */
1653:      0x30,    /* --##---- */
1654:      0x3c,    /* ---#####- */
1655:      0x00,    /* ----- */
1656:      0x00,    /* ----- */
1657:      0x00,    /* ----- */
1658:      0x00,    /* ----- */
1659:
1660:          /* 0x5c ('\') */
1661:      0x00,    /* ----- */
1662:      0x00,    /* ----- */
1663:      0x00,    /* ----- */
1664:      0x00,    /* ----- */
1665:      0x00,    /* ----- */
1666:      0xc0,    /* ##----- */
1667:      0x60,    /* -##----- */
1668:      0x30,    /* --##---- */
1669:      0x18,    /* ----##---- */
1670:      0x0c,    /* ----##-- */
1671:      0x06,    /* -----##- */
1672:      0x00,    /* ----- */
1673:      0x00,    /* ----- */
1674:      0x00,    /* ----- */
1675:      0x00,    /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 26/70

```

1676:      0x00,    /* ----- */
1677:
1678:      /* 0x5d ('']') */
1679:      0x00,    /* ----- */
1680:      0x00,    /* ----- */
1681:      0x3c,    /* ---###-- */
1682:      0x0c,    /* ----##-- */
1683:      0x0c,    /* ----##-- */
1684:      0x0c,    /* ----##-- */
1685:      0x0c,    /* ----##-- */
1686:      0x0c,    /* ----##-- */
1687:      0x0c,    /* ----##-- */
1688:      0x0c,    /* ----##-- */
1689:      0x0c,    /* ----##-- */
1690:      0x3c,    /* ---###-- */
1691:      0x00,    /* ----- */
1692:      0x00,    /* ----- */
1693:      0x00,    /* ----- */
1694:      0x00,    /* ----- */
1695:
1696:      /* 0x5e ('^') */
1697:      0x10,    /* ---#---- */
1698:      0x38,    /* ---###--- */
1699:      0x6c,    /* -##-##-- */
1700:      0xc6,    /* ##---##- */
1701:      0x00,    /* ----- */
1702:      0x00,    /* ----- */
1703:      0x00,    /* ----- */
1704:      0x00,    /* ----- */
1705:      0x00,    /* ----- */
1706:      0x00,    /* ----- */
1707:      0x00,    /* ----- */
1708:      0x00,    /* ----- */
1709:      0x00,    /* ----- */
1710:      0x00,    /* ----- */
1711:      0x00,    /* ----- */
1712:      0x00,    /* ----- */
1713:
1714:      /* 0x5f ('_') */
1715:      0x00,    /* ----- */
1716:      0x00,    /* ----- */
1717:      0x00,    /* ----- */
1718:      0x00,    /* ----- */
1719:      0x00,    /* ----- */
1720:      0x00,    /* ----- */
1721:      0x00,    /* ----- */
1722:      0x00,    /* ----- */
1723:      0x00,    /* ----- */
1724:      0x00,    /* ----- */
1725:      0x00,    /* ----- */
1726:      0x00,    /* ----- */
1727:      0x00,    /* ----- */
1728:      0x00,    /* ----- */
1729:      0xff,    /* ##### */
1730:      0x00,    /* ----- */
1731:
1732:      /* 0x60 ('`') */
1733:      0x00,    /* ----- */
1734:      0x30,    /* --##---- */
1735:      0x30,    /* --##---- */
1736:      0x30,    /* --##---- */
1737:      0x18,    /* ----##-- */
1738:      0x00,    /* ----- */
1739:      0x00,    /* ----- */
1740:      0x00,    /* ----- */
1741:      0x00,    /* ----- */
1742:      0x00,    /* ----- */

```


drivers/video/font-lat9-8x16.c

Page 27/70

```

1743:      0x00, /* ----- */
1744:      0x00, /* ----- */
1745:      0x00, /* ----- */
1746:      0x00, /* ----- */
1747:      0x00, /* ----- */
1748:      0x00, /* ----- */
1749:
1750:          /* 0x61 ('a') */
1751:      0x00, /* ----- */
1752:      0x00, /* ----- */
1753:      0x00, /* ----- */
1754:      0x00, /* ----- */
1755:      0x00, /* ----- */
1756:      0x78, /* -####--- */
1757:      0x0c, /* ----##-- */
1758:      0x7c, /* -#####-- */
1759:      0xc, /* ##--##-- */
1760:      0xc, /* ##--##-- */
1761:      0xc, /* ##--##-- */
1762:      0x76, /* -###-##- */
1763:      0x00, /* ----- */
1764:      0x00, /* ----- */
1765:      0x00, /* ----- */
1766:      0x00, /* ----- */
1767:
1768:          /* 0x62 ('b') */
1769:      0x00, /* ----- */
1770:      0x00, /* ----- */
1771:      0xe0, /* ###----- */
1772:      0x60, /* -##----- */
1773:      0x60, /* -##----- */
1774:      0x78, /* -#####--- */
1775:      0x6c, /* -##-##-- */
1776:      0x66, /* -##-##- */
1777:      0x66, /* -##-##- */
1778:      0x66, /* -##-##- */
1779:      0x66, /* -##-##- */
1780:      0x7c, /* -#####-- */
1781:      0x00, /* ----- */
1782:      0x00, /* ----- */
1783:      0x00, /* ----- */
1784:      0x00, /* ----- */
1785:
1786:          /* 0x63 ('c') */
1787:      0x00, /* ----- */
1788:      0x00, /* ----- */
1789:      0x00, /* ----- */
1790:      0x00, /* ----- */
1791:      0x00, /* ----- */
1792:      0x7c, /* -#####-- */
1793:      0xc6, /* ##----- */
1794:      0xc0, /* ##----- */
1795:      0xc0, /* ##----- */
1796:      0xc0, /* ##----- */
1797:      0xc6, /* ##----- */
1798:      0x7c, /* -#####-- */
1799:      0x00, /* ----- */
1800:      0x00, /* ----- */
1801:      0x00, /* ----- */
1802:      0x00, /* ----- */
1803:
1804:          /* 0x64 ('d') */
1805:      0x00, /* ----- */
1806:      0x00, /* ----- */
1807:      0x1c, /* ---###-- */
1808:      0x0c, /* ----##-- */
1809:      0x0c, /* ----##-- */

```

drivers/video/font-lat9-8x16.c

Page 28/70

```

1810:      0x3c, /* ---###-- */
1811:      0x6c, /* -##-##-- */
1812:      0xcc, /* ##--##-- */
1813:      0xcc, /* ##--##-- */
1814:      0xcc, /* ##--##-- */
1815:      0xcc, /* ##--##-- */
1816:      0x76, /* -###-##- */
1817:      0x00, /* ----- */
1818:      0x00, /* ----- */
1819:      0x00, /* ----- */
1820:      0x00, /* ----- */
1821:
1822:          /* 0x65 ('e') */
1823:      0x00, /* ----- */
1824:      0x00, /* ----- */
1825:      0x00, /* ----- */
1826:      0x00, /* ----- */
1827:      0x00, /* ----- */
1828:      0x7c, /* -#####-- */
1829:      0xc6, /* ##----##- */
1830:      0xfe, /* #####--- */
1831:      0xc0, /* ##----- */
1832:      0xc0, /* ##----- */
1833:      0xc6, /* ##----##- */
1834:      0x7c, /* -#####-- */
1835:      0x00, /* ----- */
1836:      0x00, /* ----- */
1837:      0x00, /* ----- */
1838:      0x00, /* ----- */
1839:
1840:          /* 0x66 ('f') */
1841:      0x00, /* ----- */
1842:      0x00, /* ----- */
1843:      0x38, /* --###---- */
1844:      0x6c, /* -##-##-- */
1845:      0x64, /* -##-##-- */
1846:      0x60, /* -##----- */
1847:      0xf0, /* #####---- */
1848:      0x60, /* -##----- */
1849:      0x60, /* -##----- */
1850:      0x60, /* -##----- */
1851:      0x60, /* -##----- */
1852:      0xf0, /* #####---- */
1853:      0x00, /* ----- */
1854:      0x00, /* ----- */
1855:      0x00, /* ----- */
1856:      0x00, /* ----- */
1857:
1858:          /* 0x67 ('g') */
1859:      0x00, /* ----- */
1860:      0x00, /* ----- */
1861:      0x00, /* ----- */
1862:      0x00, /* ----- */
1863:      0x00, /* ----- */
1864:      0x76, /* -###-##- */
1865:      0xcc, /* ##--##-- */
1866:      0xcc, /* ##--##-- */
1867:      0xcc, /* ##--##-- */
1868:      0xcc, /* ##--##-- */
1869:      0xcc, /* ##--##-- */
1870:      0x7c, /* -#####-- */
1871:      0x0c, /* ----##-- */
1872:      0xcc, /* ##--##-- */
1873:      0x78, /* -#####-- */
1874:      0x00, /* ----- */
1875:
1876:          /* 0x68 ('h') */

```

drivers/video/font-lat9-8x16.c

Page 29/70

```

1877:      0x00,    /* ----- */
1878:      0x00,    /* ----- */
1879:      0xe0,    /* ###----- */
1880:      0x60,    /* -##----- */
1881:      0x60,    /* -##----- */
1882:      0x6c,    /* -##-##-- */
1883:      0x76,    /* -###-##- */
1884:      0x66,    /* -##--##- */
1885:      0x66,    /* -##--##- */
1886:      0x66,    /* -##--##- */
1887:      0x66,    /* -##--##- */
1888:      0xe6,    /* ###--##- */
1889:      0x00,    /* ----- */
1890:      0x00,    /* ----- */
1891:      0x00,    /* ----- */
1892:      0x00,    /* ----- */
1893:
1894:          /* 0x69 ('i') */
1895:      0x00,    /* ----- */
1896:      0x00,    /* ----- */
1897:      0x18,    /* ---##---- */
1898:      0x18,    /* ---##---- */
1899:      0x00,    /* ----- */
1900:      0x38,    /* --###---- */
1901:      0x18,    /* ---##---- */
1902:      0x18,    /* ---##---- */
1903:      0x18,    /* ---##---- */
1904:      0x18,    /* ---##---- */
1905:      0x18,    /* ---##---- */
1906:      0x3c,    /* --####--- */
1907:      0x00,    /* ----- */
1908:      0x00,    /* ----- */
1909:      0x00,    /* ----- */
1910:      0x00,    /* ----- */
1911:
1912:          /* 0x6a ('j') */
1913:      0x00,    /* ----- */
1914:      0x00,    /* ----- */
1915:      0x06,    /* -----##- */
1916:      0x06,    /* -----##- */
1917:      0x00,    /* ----- */
1918:      0x0e,    /* -----###- */
1919:      0x06,    /* -----##- */
1920:      0x06,    /* -----##- */
1921:      0x06,    /* -----##- */
1922:      0x06,    /* -----##- */
1923:      0x06,    /* -----##- */
1924:      0x06,    /* -----##- */
1925:      0x66,    /* -##--##- */
1926:      0x66,    /* -##--##- */
1927:      0x3c,    /* --####--- */
1928:      0x00,    /* ----- */
1929:
1930:          /* 0x6b ('k') */
1931:      0x00,    /* ----- */
1932:      0x00,    /* ----- */
1933:      0xe0,    /* ###----- */
1934:      0x60,    /* -##----- */
1935:      0x60,    /* -##----- */
1936:      0x66,    /* -##--##- */
1937:      0x6c,    /* -##-##-- */
1938:      0x78,    /* -####---- */
1939:      0x78,    /* -####---- */
1940:      0x6c,    /* -##-##-- */
1941:      0x66,    /* -##--##- */
1942:      0xe6,    /* ###--##- */
1943:      0x00,    /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 30/70

```

1944:      0x00, /* ----- */
1945:      0x00, /* ----- */
1946:      0x00, /* ----- */
1947:
1948:          /* 0x6c ('l') */
1949:      0x00, /* ----- */
1950:      0x00, /* ----- */
1951:      0x38, /* ---###--- */
1952:      0x18, /* ---##--- */
1953:      0x18, /* ---##--- */
1954:      0x18, /* ---##--- */
1955:      0x18, /* ---##--- */
1956:      0x18, /* ---##--- */
1957:      0x18, /* ---##--- */
1958:      0x18, /* ---##--- */
1959:      0x18, /* ---##--- */
1960:      0x3c, /* ---####--- */
1961:      0x00, /* ----- */
1962:      0x00, /* ----- */
1963:      0x00, /* ----- */
1964:      0x00, /* ----- */
1965:
1966:          /* 0x6d ('m') */
1967:      0x00, /* ----- */
1968:      0x00, /* ----- */
1969:      0x00, /* ----- */
1970:      0x00, /* ----- */
1971:      0x00, /* ----- */
1972:      0xec, /* ###-##--- */
1973:      0xfe, /* #####--- */
1974:      0xd6, /* #-#-##--- */
1975:      0xd6, /* #-#-##--- */
1976:      0xd6, /* #-#-##--- */
1977:      0xd6, /* #-#-##--- */
1978:      0xc6, /* #----##- */
1979:      0x00, /* ----- */
1980:      0x00, /* ----- */
1981:      0x00, /* ----- */
1982:      0x00, /* ----- */
1983:
1984:          /* 0x6e ('n') */
1985:      0x00, /* ----- */
1986:      0x00, /* ----- */
1987:      0x00, /* ----- */
1988:      0x00, /* ----- */
1989:      0x00, /* ----- */
1990:      0xdc, /* #-###--- */
1991:      0x66, /* -##---##- */
1992:      0x66, /* -##---##- */
1993:      0x66, /* -##---##- */
1994:      0x66, /* -##---##- */
1995:      0x66, /* -##---##- */
1996:      0x66, /* -##---##- */
1997:      0x00, /* ----- */
1998:      0x00, /* ----- */
1999:      0x00, /* ----- */
2000:      0x00, /* ----- */
2001:
2002:          /* 0x6f ('o') */
2003:      0x00, /* ----- */
2004:      0x00, /* ----- */
2005:      0x00, /* ----- */
2006:      0x00, /* ----- */
2007:      0x00, /* ----- */
2008:      0x7c, /* -#####--- */
2009:      0xc6, /* #----##- */
2010:      0xc6, /* #----##- */

```

drivers/video/font-lat9-8x16.c Page 31/70

```

2011:      0xc6,      /* ##---##- */
2012:      0xc6,      /* ##---##- */
2013:      0xc6,      /* ##---##- */
2014:      0x7c,      /* -#####- */
2015:      0x00,      /* ----- */
2016:      0x00,      /* ----- */
2017:      0x00,      /* ----- */
2018:      0x00,      /* ----- */
2019:
2020:      /* 0x70 ('p') */
2021:      0x00,      /* ----- */
2022:      0x00,      /* ----- */
2023:      0x00,      /* ----- */
2024:      0x00,      /* ----- */
2025:      0x00,      /* ----- */
2026:      0xdc,      /* ##-###- */
2027:      0x66,      /* -##-##- */
2028:      0x66,      /* -##-##- */
2029:      0x66,      /* -##-##- */
2030:      0x66,      /* -##-##- */
2031:      0x66,      /* -##-##- */
2032:      0x7c,      /* -#####- */
2033:      0x60,      /* -##----- */
2034:      0x60,      /* -##----- */
2035:      0xf0,      /* ####----- */
2036:      0x00,      /* ----- */
2037:
2038:      /* 0x71 ('q') */
2039:      0x00,      /* ----- */
2040:      0x00,      /* ----- */
2041:      0x00,      /* ----- */
2042:      0x00,      /* ----- */
2043:      0x00,      /* ----- */
2044:      0x76,      /* -###-##- */
2045:      0xcc,      /* ##-##- */
2046:      0xcc,      /* ##-##- */
2047:      0xcc,      /* ##-##- */
2048:      0xcc,      /* ##-##- */
2049:      0xcc,      /* ##-##- */
2050:      0x7c,      /* -#####- */
2051:      0x0c,      /* ----##- */
2052:      0x0c,      /* ----##- */
2053:      0x1e,      /* ----###- */
2054:      0x00,      /* ----- */
2055:
2056:      /* 0x72 ('r') */
2057:      0x00,      /* ----- */
2058:      0x00,      /* ----- */
2059:      0x00,      /* ----- */
2060:      0x00,      /* ----- */
2061:      0x00,      /* ----- */
2062:      0xdc,      /* ##-###- */
2063:      0x76,      /* -###-##- */
2064:      0x66,      /* -##-##- */
2065:      0x60,      /* -##----- */
2066:      0x60,      /* -##----- */
2067:      0x60,      /* -##----- */
2068:      0xf0,      /* ####----- */
2069:      0x00,      /* ----- */
2070:      0x00,      /* ----- */
2071:      0x00,      /* ----- */
2072:      0x00,      /* ----- */
2073:
2074:      /* 0x73 ('s') */
2075:      0x00,      /* ----- */
2076:      0x00,      /* ----- */
2077:      0x00,      /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 32/70

```

2078:      0x00, /* ----- */
2079:      0x00, /* ----- */
2080:      0x7c, /* #####-- */
2081:      0xc6, /* ##---##- */
2082:      0x60, /* -##----- */
2083:      0x38, /* --###---- */
2084:      0x0c, /* ----##--- */
2085:      0xc6, /* ##---##- */
2086:      0x7c, /* #####-- */
2087:      0x00, /* ----- */
2088:      0x00, /* ----- */
2089:      0x00, /* ----- */
2090:      0x00, /* ----- */
2091:
2092:          /* 0x74 ('t') */
2093:      0x00, /* ----- */
2094:      0x00, /* ----- */
2095:      0x10, /* ---#----- */
2096:      0x30, /* --##----- */
2097:      0x30, /* --##----- */
2098:      0xfc, /* #####--- */
2099:      0x30, /* --##----- */
2100:      0x30, /* --##----- */
2101:      0x30, /* --##----- */
2102:      0x30, /* --##----- */
2103:      0x36, /* --##-##- */
2104:      0x1c, /* ----###--- */
2105:      0x00, /* ----- */
2106:      0x00, /* ----- */
2107:      0x00, /* ----- */
2108:      0x00, /* ----- */
2109:
2110:          /* 0x75 ('u') */
2111:      0x00, /* ----- */
2112:      0x00, /* ----- */
2113:      0x00, /* ----- */
2114:      0x00, /* ----- */
2115:      0x00, /* ----- */
2116:      0xcc, /* ##---##- */
2117:      0xcc, /* ##---##- */
2118:      0xcc, /* ##---##- */
2119:      0xcc, /* ##---##- */
2120:      0xcc, /* ##---##- */
2121:      0xcc, /* ##---##- */
2122:      0x76, /* -###-##- */
2123:      0x00, /* ----- */
2124:      0x00, /* ----- */
2125:      0x00, /* ----- */
2126:      0x00, /* ----- */
2127:
2128:          /* 0x76 ('v') */
2129:      0x00, /* ----- */
2130:      0x00, /* ----- */
2131:      0x00, /* ----- */
2132:      0x00, /* ----- */
2133:      0x00, /* ----- */
2134:      0x66, /* -##---##- */
2135:      0x66, /* -##---##- */
2136:      0x66, /* -##---##- */
2137:      0x66, /* -##---##- */
2138:      0x66, /* -##---##- */
2139:      0x3c, /* --##### */
2140:      0x18, /* ----##--- */
2141:      0x00, /* ----- */
2142:      0x00, /* ----- */
2143:      0x00, /* ----- */
2144:      0x00, /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 33/70

```

2145:
2146:          /* 0x77 ('w') */
2147:    0x00,  /* ----- */
2148:    0x00,  /* ----- */
2149:    0x00,  /* ----- */
2150:    0x00,  /* ----- */
2151:    0x00,  /* ----- */
2152:    0xc6,  /* ##---##- */
2153:    0xc6,  /* ##---##- */
2154:    0xd6,  /* ##-##-##- */
2155:    0xd6,  /* ##-##-##- */
2156:    0xd6,  /* ##-##-##- */
2157:    0xfe,  /* #####-#- */
2158:    0x6c,  /* -##-##-#- */
2159:    0x00,  /* ----- */
2160:    0x00,  /* ----- */
2161:    0x00,  /* ----- */
2162:    0x00,  /* ----- */
2163:
2164:          /* 0x78 ('x') */
2165:    0x00,  /* ----- */
2166:    0x00,  /* ----- */
2167:    0x00,  /* ----- */
2168:    0x00,  /* ----- */
2169:    0x00,  /* ----- */
2170:    0xc6,  /* ##---##- */
2171:    0x6c,  /* -##-##-#- */
2172:    0x38,  /* --###---#- */
2173:    0x38,  /* --###---#- */
2174:    0x38,  /* --###---#- */
2175:    0x6c,  /* -##-##-#- */
2176:    0xc6,  /* ##---##- */
2177:    0x00,  /* ----- */
2178:    0x00,  /* ----- */
2179:    0x00,  /* ----- */
2180:    0x00,  /* ----- */
2181:
2182:          /* 0x79 ('y') */
2183:    0x00,  /* ----- */
2184:    0x00,  /* ----- */
2185:    0x00,  /* ----- */
2186:    0x00,  /* ----- */
2187:    0x00,  /* ----- */
2188:    0xc6,  /* ##---##- */
2189:    0xc6,  /* ##---##- */
2190:    0xc6,  /* ##---##- */
2191:    0xc6,  /* ##---##- */
2192:    0xc6,  /* ##---##- */
2193:    0xc6,  /* ##---##- */
2194:    0x7e,  /* -#####-#- */
2195:    0x06,  /* -----##- */
2196:    0x0c,  /* -----##- */
2197:    0xf8,  /* #####---#- */
2198:    0x00,  /* ----- */
2199:
2200:          /* 0x7a ('z') */
2201:    0x00,  /* ----- */
2202:    0x00,  /* ----- */
2203:    0x00,  /* ----- */
2204:    0x00,  /* ----- */
2205:    0x00,  /* ----- */
2206:    0xfe,  /* #####-#- */
2207:    0xcc,  /* ##---##-#- */
2208:    0x18,  /* ---##---#- */
2209:    0x30,  /* --##----- */
2210:    0x60,  /* -##-----#- */
2211:    0xc6,  /* ##---##-#- */

```

drivers/video/font-lat9-8x16.c

Page 34/70

```

2212:      0xfe, /* #####- */
2213:      0x00, /* ----- */
2214:      0x00, /* ----- */
2215:      0x00, /* ----- */
2216:      0x00, /* ----- */
2217:
2218:          /* 0x7b ('{') */
2219:      0x00, /* ----- */
2220:      0x00, /* ----- */
2221:      0x0e, /* ----###- */
2222:      0x18, /* ---##--- */
2223:      0x18, /* ---##--- */
2224:      0x18, /* ---##--- */
2225:      0x70, /* -###---- */
2226:      0x18, /* ---##--- */
2227:      0x18, /* ---##--- */
2228:      0x18, /* ---##--- */
2229:      0x18, /* ---##--- */
2230:      0x0e, /* ----###- */
2231:      0x00, /* ----- */
2232:      0x00, /* ----- */
2233:      0x00, /* ----- */
2234:      0x00, /* ----- */
2235:
2236:          /* 0x7c ('|') */
2237:      0x00, /* ----- */
2238:      0x00, /* ----- */
2239:      0x18, /* ---##--- */
2240:      0x18, /* ---##--- */
2241:      0x18, /* ---##--- */
2242:      0x18, /* ---##--- */
2243:      0x18, /* ---##--- */
2244:      0x18, /* ---##--- */
2245:      0x18, /* ---##--- */
2246:      0x18, /* ---##--- */
2247:      0x18, /* ---##--- */
2248:      0x18, /* ---##--- */
2249:      0x00, /* ----- */
2250:      0x00, /* ----- */
2251:      0x00, /* ----- */
2252:      0x00, /* ----- */
2253:
2254:          /* 0x7d ('}') */
2255:      0x00, /* ----- */
2256:      0x00, /* ----- */
2257:      0x70, /* -###---- */
2258:      0x18, /* ---##--- */
2259:      0x18, /* ---##--- */
2260:      0x18, /* ---##--- */
2261:      0x0e, /* ----###- */
2262:      0x18, /* ---##--- */
2263:      0x18, /* ---##--- */
2264:      0x18, /* ---##--- */
2265:      0x18, /* ---##--- */
2266:      0x70, /* -###---- */
2267:      0x00, /* ----- */
2268:      0x00, /* ----- */
2269:      0x00, /* ----- */
2270:      0x00, /* ----- */
2271:
2272:          /* 0x7e ('~') */
2273:      0x00, /* ----- */
2274:      0x00, /* ----- */
2275:      0x76, /* -###-##- */
2276:      0xdc, /* ##-##-#- */
2277:      0x00, /* ----- */
2278:      0x00, /* ----- */

```


drivers/video/font-lat9-8x16.c

Page 35/70

```

2279:      0x00, /* ----- */
2280:      0x00, /* ----- */
2281:      0x00, /* ----- */
2282:      0x00, /* ----- */
2283:      0x00, /* ----- */
2284:      0x00, /* ----- */
2285:      0x00, /* ----- */
2286:      0x00, /* ----- */
2287:      0x00, /* ----- */
2288:      0x00, /* ----- */
2289:
2290:          /* 0x7f */
2291:      0x00, /* ----- */
2292:      0x66, /* -##-##- */
2293:      0x00, /* ----- */
2294:      0x66, /* -##-##- */
2295:      0x66, /* -##-##- */
2296:      0x66, /* -##-##- */
2297:      0x66, /* -##-##- */
2298:      0x3c, /* ---###--- */
2299:      0x18, /* ---##--- */
2300:      0x18, /* ---##--- */
2301:      0x18, /* ---##--- */
2302:      0x3c, /* ---###--- */
2303:      0x00, /* ----- */
2304:      0x00, /* ----- */
2305:      0x00, /* ----- */
2306:      0x00, /* ----- */
2307:
2308:          /* 0x80 */
2309:      0x00, /* ----- */
2310:      0x00, /* ----- */
2311:      0x00, /* ----- */
2312:      0x00, /* ----- */
2313:      0xff, /* ##### */
2314:      0x00, /* ----- */
2315:      0x00, /* ----- */
2316:      0x00, /* ----- */
2317:      0x00, /* ----- */
2318:      0x00, /* ----- */
2319:      0x00, /* ----- */
2320:      0x00, /* ----- */
2321:      0x00, /* ----- */
2322:      0x00, /* ----- */
2323:      0x00, /* ----- */
2324:      0x00, /* ----- */
2325:
2326:          /* 0x81 */
2327:      0x18, /* ---##--- */
2328:      0x18, /* ---##--- */
2329:      0x18, /* ---##--- */
2330:      0x18, /* ---##--- */
2331:      0x18, /* ---##--- */
2332:      0x18, /* ---##--- */
2333:      0x18, /* ---##--- */
2334:      0x18, /* ---##--- */
2335:      0x00, /* ----- */
2336:      0x00, /* ----- */
2337:      0x00, /* ----- */
2338:      0x00, /* ----- */
2339:      0x00, /* ----- */
2340:      0x00, /* ----- */
2341:      0x00, /* ----- */
2342:      0x00, /* ----- */
2343:
2344:          /* 0x82 */
2345:      0x00, /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 36/70

```

2346:      0x00,    /* ----- */
2347:      0x00,    /* ----- */
2348:      0x00,    /* ----- */
2349:      0x00,    /* ----- */
2350:      0x00,    /* ----- */
2351:      0x00,    /* ----- */
2352:      0x1f,    /* ----##### */
2353:      0x00,    /* ----- */
2354:      0x00,    /* ----- */
2355:      0x00,    /* ----- */
2356:      0x00,    /* ----- */
2357:      0x00,    /* ----- */
2358:      0x00,    /* ----- */
2359:      0x00,    /* ----- */
2360:      0x00,    /* ----- */
2361:
2362:          /* 0x83      */
2363:      0x18,    /* ---##--- */
2364:      0x18,    /* ---##--- */
2365:      0x18,    /* ---##--- */
2366:      0x18,    /* ---##--- */
2367:      0x18,    /* ---##--- */
2368:      0x18,    /* ---##--- */
2369:      0x18,    /* ---##--- */
2370:      0x1f,    /* ----##### */
2371:      0x00,    /* ----- */
2372:      0x00,    /* ----- */
2373:      0x00,    /* ----- */
2374:      0x00,    /* ----- */
2375:      0x00,    /* ----- */
2376:      0x00,    /* ----- */
2377:      0x00,    /* ----- */
2378:      0x00,    /* ----- */
2379:
2380:          /* 0x84      */
2381:      0x00,    /* ----- */
2382:      0x00,    /* ----- */
2383:      0x00,    /* ----- */
2384:      0x00,    /* ----- */
2385:      0x00,    /* ----- */
2386:      0x00,    /* ----- */
2387:      0x00,    /* ----- */
2388:      0x18,    /* ---##--- */
2389:      0x18,    /* ---##--- */
2390:      0x18,    /* ---##--- */
2391:      0x18,    /* ---##--- */
2392:      0x18,    /* ---##--- */
2393:      0x18,    /* ---##--- */
2394:      0x18,    /* ---##--- */
2395:      0x18,    /* ---##--- */
2396:      0x18,    /* ---##--- */
2397:
2398:          /* 0x85      */
2399:      0x18,    /* ---##--- */
2400:      0x18,    /* ---##--- */
2401:      0x18,    /* ---##--- */
2402:      0x18,    /* ---##--- */
2403:      0x18,    /* ---##--- */
2404:      0x18,    /* ---##--- */
2405:      0x18,    /* ---##--- */
2406:      0x18,    /* ---##--- */
2407:      0x18,    /* ---##--- */
2408:      0x18,    /* ---##--- */
2409:      0x18,    /* ---##--- */
2410:      0x18,    /* ---##--- */
2411:      0x18,    /* ---##--- */
2412:      0x18,    /* ---##--- */

```

drivers/video/font-lat9-8x16.c

Page 37/70

```

2413:      0x18, /* ---##--- */
2414:      0x18, /* ---##--- */
2415:
2416:          /* 0x86 */
2417:      0x00, /* ----- */
2418:      0x00, /* ----- */
2419:      0x00, /* ----- */
2420:      0x00, /* ----- */
2421:      0x00, /* ----- */
2422:      0x00, /* ----- */
2423:      0x00, /* ----- */
2424:      0x1f, /* ---##### */
2425:      0x18, /* ---##--- */
2426:      0x18, /* ---##--- */
2427:      0x18, /* ---##--- */
2428:      0x18, /* ---##--- */
2429:      0x18, /* ---##--- */
2430:      0x18, /* ---##--- */
2431:      0x18, /* ---##--- */
2432:      0x18, /* ---##--- */
2433:
2434:          /* 0x87 */
2435:      0x18, /* ---##--- */
2436:      0x18, /* ---##--- */
2437:      0x18, /* ---##--- */
2438:      0x18, /* ---##--- */
2439:      0x18, /* ---##--- */
2440:      0x18, /* ---##--- */
2441:      0x18, /* ---##--- */
2442:      0x1f, /* ---##### */
2443:      0x18, /* ---##--- */
2444:      0x18, /* ---##--- */
2445:      0x18, /* ---##--- */
2446:      0x18, /* ---##--- */
2447:      0x18, /* ---##--- */
2448:      0x18, /* ---##--- */
2449:      0x18, /* ---##--- */
2450:      0x18, /* ---##--- */
2451:
2452:          /* 0x88 */
2453:      0x00, /* ----- */
2454:      0x00, /* ----- */
2455:      0x00, /* ----- */
2456:      0x00, /* ----- */
2457:      0x00, /* ----- */
2458:      0x00, /* ----- */
2459:      0x00, /* ----- */
2460:      0xf8, /* #####--- */
2461:      0x00, /* ----- */
2462:      0x00, /* ----- */
2463:      0x00, /* ----- */
2464:      0x00, /* ----- */
2465:      0x00, /* ----- */
2466:      0x00, /* ----- */
2467:      0x00, /* ----- */
2468:      0x00, /* ----- */
2469:
2470:          /* 0x89 */
2471:      0x18, /* ---##--- */
2472:      0x18, /* ---##--- */
2473:      0x18, /* ---##--- */
2474:      0x18, /* ---##--- */
2475:      0x18, /* ---##--- */
2476:      0x18, /* ---##--- */
2477:      0x18, /* ---##--- */
2478:      0xf8, /* #####--- */
2479:      0x00, /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 38/70

```

2480:      0x00, /* ----- */
2481:      0x00, /* ----- */
2482:      0x00, /* ----- */
2483:      0x00, /* ----- */
2484:      0x00, /* ----- */
2485:      0x00, /* ----- */
2486:      0x00, /* ----- */
2487:
2488:          /* 0x8a */
2489:      0x00, /* ----- */
2490:      0x00, /* ----- */
2491:      0x00, /* ----- */
2492:      0x00, /* ----- */
2493:      0x00, /* ----- */
2494:      0x00, /* ----- */
2495:      0x00, /* ----- */
2496:      0xff, /* ##### */
2497:      0x00, /* ----- */
2498:      0x00, /* ----- */
2499:      0x00, /* ----- */
2500:      0x00, /* ----- */
2501:      0x00, /* ----- */
2502:      0x00, /* ----- */
2503:      0x00, /* ----- */
2504:      0x00, /* ----- */
2505:
2506:          /* 0x8b */
2507:      0x18, /* ---#--- */
2508:      0x18, /* ---#--- */
2509:      0x18, /* ---#--- */
2510:      0x18, /* ---#--- */
2511:      0x18, /* ---#--- */
2512:      0x18, /* ---#--- */
2513:      0x18, /* ---#--- */
2514:      0xff, /* ##### */
2515:      0x00, /* ----- */
2516:      0x00, /* ----- */
2517:      0x00, /* ----- */
2518:      0x00, /* ----- */
2519:      0x00, /* ----- */
2520:      0x00, /* ----- */
2521:      0x00, /* ----- */
2522:      0x00, /* ----- */
2523:
2524:          /* 0x8c */
2525:      0x00, /* ----- */
2526:      0x00, /* ----- */
2527:      0x00, /* ----- */
2528:      0x00, /* ----- */
2529:      0x00, /* ----- */
2530:      0x00, /* ----- */
2531:      0x00, /* ----- */
2532:      0xf8, /* ####--- */
2533:      0x18, /* ---#--- */
2534:      0x18, /* ---#--- */
2535:      0x18, /* ---#--- */
2536:      0x18, /* ---#--- */
2537:      0x18, /* ---#--- */
2538:      0x18, /* ---#--- */
2539:      0x18, /* ---#--- */
2540:      0x18, /* ---#--- */
2541:
2542:          /* 0x8d */
2543:      0x18, /* ---#--- */
2544:      0x18, /* ---#--- */
2545:      0x18, /* ---#--- */
2546:      0x18, /* ---#--- */

```

drivers/video/font-lat9-8x16.c

Page 39/70

```

2547:      0x18, /* ----##---- */
2548:      0x18, /* ----##---- */
2549:      0x18, /* ----##---- */
2550:      0xf8, /* #####---- */
2551:      0x18, /* ----##---- */
2552:      0x18, /* ----##---- */
2553:      0x18, /* ----##---- */
2554:      0x18, /* ----##---- */
2555:      0x18, /* ----##---- */
2556:      0x18, /* ----##---- */
2557:      0x18, /* ----##---- */
2558:      0x18, /* ----##---- */
2559:
2560:          /* 0x8e */
2561:      0x00, /* ----- */
2562:      0x00, /* ----- */
2563:      0x00, /* ----- */
2564:      0x00, /* ----- */
2565:      0x00, /* ----- */
2566:      0x00, /* ----- */
2567:      0x00, /* ----- */
2568:      0xff, /* ##### */
2569:      0x18, /* ----##---- */
2570:      0x18, /* ----##---- */
2571:      0x18, /* ----##---- */
2572:      0x18, /* ----##---- */
2573:      0x18, /* ----##---- */
2574:      0x18, /* ----##---- */
2575:      0x18, /* ----##---- */
2576:      0x18, /* ----##---- */
2577:
2578:          /* 0x8f */
2579:      0x18, /* ----##---- */
2580:      0x18, /* ----##---- */
2581:      0x18, /* ----##---- */
2582:      0x18, /* ----##---- */
2583:      0x18, /* ----##---- */
2584:      0x18, /* ----##---- */
2585:      0x18, /* ----##---- */
2586:      0xff, /* ##### */
2587:      0x18, /* ----##---- */
2588:      0x18, /* ----##---- */
2589:      0x18, /* ----##---- */
2590:      0x18, /* ----##---- */
2591:      0x18, /* ----##---- */
2592:      0x18, /* ----##---- */
2593:      0x18, /* ----##---- */
2594:      0x18, /* ----##---- */
2595:
2596:          /* 0x90 */
2597:      0x00, /* ----- */
2598:      0x00, /* ----- */
2599:      0x00, /* ----- */
2600:      0x00, /* ----- */
2601:      0x00, /* ----- */
2602:      0x00, /* ----- */
2603:      0x00, /* ----- */
2604:      0x00, /* ----- */
2605:      0x00, /* ----- */
2606:      0x00, /* ----- */
2607:      0xff, /* ##### */
2608:      0x00, /* ----- */
2609:      0x00, /* ----- */
2610:      0x00, /* ----- */
2611:      0x00, /* ----- */
2612:      0x00, /* ----- */
2613:

```

drivers/video/font-lat9-8x16.c

Page 40/70

```

2614:                /* 0x91          */
2615:                0x6c, /* -##-##-- */
2616:                0x6c, /* -##-##-- */
2617:                0x6c, /* -##-##-- */
2618:                0x6c, /* -##-##-- */
2619:                0x6c, /* -##-##-- */
2620:                0x6c, /* -##-##-- */
2621:                0x6c, /* -##-##-- */
2622:                0x6c, /* -##-##-- */
2623:                0x7c, /* -#####-- */
2624:                0x00, /* ----- */
2625:                0x00, /* ----- */
2626:                0x00, /* ----- */
2627:                0x00, /* ----- */
2628:                0x00, /* ----- */
2629:                0x00, /* ----- */
2630:                0x00, /* ----- */
2631:
2632:                /* 0x92          */
2633:                0x00, /* ----- */
2634:                0x00, /* ----- */
2635:                0x00, /* ----- */
2636:                0x00, /* ----- */
2637:                0x00, /* ----- */
2638:                0x00, /* ----- */
2639:                0x7f, /* -##### */
2640:                0x60, /* -##----- */
2641:                0x7f, /* -##### */
2642:                0x00, /* ----- */
2643:                0x00, /* ----- */
2644:                0x00, /* ----- */
2645:                0x00, /* ----- */
2646:                0x00, /* ----- */
2647:                0x00, /* ----- */
2648:                0x00, /* ----- */
2649:
2650:                /* 0x93          */
2651:                0x6c, /* -##-##-- */
2652:                0x6c, /* -##-##-- */
2653:                0x6c, /* -##-##-- */
2654:                0x6c, /* -##-##-- */
2655:                0x6c, /* -##-##-- */
2656:                0x6c, /* -##-##-- */
2657:                0x6f, /* -##-#### */
2658:                0x60, /* -##----- */
2659:                0x7f, /* -##### */
2660:                0x00, /* ----- */
2661:                0x00, /* ----- */
2662:                0x00, /* ----- */
2663:                0x00, /* ----- */
2664:                0x00, /* ----- */
2665:                0x00, /* ----- */
2666:                0x00, /* ----- */
2667:
2668:                /* 0x94          */
2669:                0x00, /* ----- */
2670:                0x00, /* ----- */
2671:                0x00, /* ----- */
2672:                0x00, /* ----- */
2673:                0x00, /* ----- */
2674:                0x00, /* ----- */
2675:                0x7c, /* -#####-- */
2676:                0x6c, /* -##-##-- */
2677:                0x6c, /* -##-##-- */
2678:                0x6c, /* -##-##-- */
2679:                0x6c, /* -##-##-- */
2680:                0x6c, /* -##-##-- */

```

drivers/video/font-lat9-8x16.c

Page 41/70

```

2681:      0x6c, /* -##-##-- */
2682:      0x6c, /* -##-##-- */
2683:      0x6c, /* -##-##-- */
2684:      0x6c, /* -##-##-- */
2685:
2686:          /* 0x95 */
2687:      0x6c, /* -##-##-- */
2688:      0x6c, /* -##-##-- */
2689:      0x6c, /* -##-##-- */
2690:      0x6c, /* -##-##-- */
2691:      0x6c, /* -##-##-- */
2692:      0x6c, /* -##-##-- */
2693:      0x6c, /* -##-##-- */
2694:      0x6c, /* -##-##-- */
2695:      0x6c, /* -##-##-- */
2696:      0x6c, /* -##-##-- */
2697:      0x6c, /* -##-##-- */
2698:      0x6c, /* -##-##-- */
2699:      0x6c, /* -##-##-- */
2700:      0x6c, /* -##-##-- */
2701:      0x6c, /* -##-##-- */
2702:      0x6c, /* -##-##-- */
2703:
2704:          /* 0x96 */
2705:      0x00, /* ----- */
2706:      0x00, /* ----- */
2707:      0x00, /* ----- */
2708:      0x00, /* ----- */
2709:      0x00, /* ----- */
2710:      0x00, /* ----- */
2711:      0x7f, /* ##### */
2712:      0x60, /* -##----- */
2713:      0x6f, /* -##-#### */
2714:      0x6c, /* -##-##-- */
2715:      0x6c, /* -##-##-- */
2716:      0x6c, /* -##-##-- */
2717:      0x6c, /* -##-##-- */
2718:      0x6c, /* -##-##-- */
2719:      0x6c, /* -##-##-- */
2720:      0x6c, /* -##-##-- */
2721:
2722:          /* 0x97 */
2723:      0x6c, /* -##-##-- */
2724:      0x6c, /* -##-##-- */
2725:      0x6c, /* -##-##-- */
2726:      0x6c, /* -##-##-- */
2727:      0x6c, /* -##-##-- */
2728:      0x6c, /* -##-##-- */
2729:      0x6f, /* -##-#### */
2730:      0x60, /* -##----- */
2731:      0x6f, /* -##-#### */
2732:      0x6c, /* -##-##-- */
2733:      0x6c, /* -##-##-- */
2734:      0x6c, /* -##-##-- */
2735:      0x6c, /* -##-##-- */
2736:      0x6c, /* -##-##-- */
2737:      0x6c, /* -##-##-- */
2738:      0x6c, /* -##-##-- */
2739:
2740:          /* 0x98 */
2741:      0x00, /* ----- */
2742:      0x00, /* ----- */
2743:      0x00, /* ----- */
2744:      0x00, /* ----- */
2745:      0x00, /* ----- */
2746:      0x00, /* ----- */
2747:      0xfC, /* #####-- */

```

drivers/video/font-lat9-8x16.c

Page 42/70

```

2748:      0x0c, /* ----##-- */
2749:      0xfc, /* #####-- */
2750:      0x00, /* ----- */
2751:      0x00, /* ----- */
2752:      0x00, /* ----- */
2753:      0x00, /* ----- */
2754:      0x00, /* ----- */
2755:      0x00, /* ----- */
2756:      0x00, /* ----- */
2757:
2758:          /* 0x99 */
2759:      0x6c, /* -##-##-- */
2760:      0x6c, /* -##-##-- */
2761:      0x6c, /* -##-##-- */
2762:      0x6c, /* -##-##-- */
2763:      0x6c, /* -##-##-- */
2764:      0x6c, /* -##-##-- */
2765:      0xec, /* ###-##-- */
2766:      0x0c, /* ----##-- */
2767:      0xfc, /* #####-- */
2768:      0x00, /* ----- */
2769:      0x00, /* ----- */
2770:      0x00, /* ----- */
2771:      0x00, /* ----- */
2772:      0x00, /* ----- */
2773:      0x00, /* ----- */
2774:      0x00, /* ----- */
2775:
2776:          /* 0x9a */
2777:      0x00, /* ----- */
2778:      0x00, /* ----- */
2779:      0x00, /* ----- */
2780:      0x00, /* ----- */
2781:      0x00, /* ----- */
2782:      0x00, /* ----- */
2783:      0xff, /* ##### */
2784:      0x00, /* ----- */
2785:      0xff, /* ##### */
2786:      0x00, /* ----- */
2787:      0x00, /* ----- */
2788:      0x00, /* ----- */
2789:      0x00, /* ----- */
2790:      0x00, /* ----- */
2791:      0x00, /* ----- */
2792:      0x00, /* ----- */
2793:
2794:          /* 0x9b */
2795:      0x6c, /* -##-##-- */
2796:      0x6c, /* -##-##-- */
2797:      0x6c, /* -##-##-- */
2798:      0x6c, /* -##-##-- */
2799:      0x6c, /* -##-##-- */
2800:      0x6c, /* -##-##-- */
2801:      0xef, /* ###-#### */
2802:      0x00, /* ----- */
2803:      0xff, /* ##### */
2804:      0x00, /* ----- */
2805:      0x00, /* ----- */
2806:      0x00, /* ----- */
2807:      0x00, /* ----- */
2808:      0x00, /* ----- */
2809:      0x00, /* ----- */
2810:      0x00, /* ----- */
2811:
2812:          /* 0x9c */
2813:      0x00, /* ----- */
2814:      0x00, /* ----- */

```


drivers/video/font-lat9-8x16.c

Page 43/70

```

2815:      0x00,    /* ----- */
2816:      0x00,    /* ----- */
2817:      0x00,    /* ----- */
2818:      0x00,    /* ----- */
2819:      0xfc,    /* ##### */
2820:      0x0c,    /* ----## */
2821:      0xec,    /* ###-## */
2822:      0x6c,    /* -##-## */
2823:      0x6c,    /* -##-## */
2824:      0x6c,    /* -##-## */
2825:      0x6c,    /* -##-## */
2826:      0x6c,    /* -##-## */
2827:      0x6c,    /* -##-## */
2828:      0x6c,    /* -##-## */
2829:
2830:          /* 0x9d */
2831:      0x6c,    /* -##-## */
2832:      0x6c,    /* -##-## */
2833:      0x6c,    /* -##-## */
2834:      0x6c,    /* -##-## */
2835:      0x6c,    /* -##-## */
2836:      0x6c,    /* -##-## */
2837:      0xec,    /* ###-## */
2838:      0x0c,    /* ----## */
2839:      0xec,    /* ###-## */
2840:      0x6c,    /* -##-## */
2841:      0x6c,    /* -##-## */
2842:      0x6c,    /* -##-## */
2843:      0x6c,    /* -##-## */
2844:      0x6c,    /* -##-## */
2845:      0x6c,    /* -##-## */
2846:      0x6c,    /* -##-## */
2847:
2848:          /* 0x9e */
2849:      0x00,    /* ----- */
2850:      0x00,    /* ----- */
2851:      0x00,    /* ----- */
2852:      0x00,    /* ----- */
2853:      0x00,    /* ----- */
2854:      0x00,    /* ----- */
2855:      0xff,    /* ##### */
2856:      0x00,    /* ----- */
2857:      0xef,    /* ###-#### */
2858:      0x6c,    /* -##-## */
2859:      0x6c,    /* -##-## */
2860:      0x6c,    /* -##-## */
2861:      0x6c,    /* -##-## */
2862:      0x6c,    /* -##-## */
2863:      0x6c,    /* -##-## */
2864:      0x6c,    /* -##-## */
2865:
2866:          /* 0x9f */
2867:      0x6c,    /* -##-## */
2868:      0x6c,    /* -##-## */
2869:      0x6c,    /* -##-## */
2870:      0x6c,    /* -##-## */
2871:      0x6c,    /* -##-## */
2872:      0x6c,    /* -##-## */
2873:      0xef,    /* ###-#### */
2874:      0x00,    /* ----- */
2875:      0xef,    /* ###-#### */
2876:      0x6c,    /* -##-## */
2877:      0x6c,    /* -##-## */
2878:      0x6c,    /* -##-## */
2879:      0x6c,    /* -##-## */
2880:      0x6c,    /* -##-## */
2881:      0x6c,    /* -##-## */

```

drivers/video/font-lat9-8x16.c

Page 44/70

```

2882:      0x6c,    /* -##-##-- */
2883:
2884:          /* 0xa0 */
2885:      0x00,    /* ----- */
2886:      0x00,    /* ----- */
2887:      0x00,    /* ----- */
2888:      0x00,    /* ----- */
2889:      0x00,    /* ----- */
2890:      0x00,    /* ----- */
2891:      0x00,    /* ----- */
2892:      0x00,    /* ----- */
2893:      0x00,    /* ----- */
2894:      0x00,    /* ----- */
2895:      0x82,    /* #-----# */
2896:      0xfe,    /* #####-- */
2897:      0x00,    /* ----- */
2898:      0x00,    /* ----- */
2899:      0x00,    /* ----- */
2900:      0x00,    /* ----- */
2901:
2902:          /* 0xa1 */
2903:      0x00,    /* ----- */
2904:      0x00,    /* ----- */
2905:      0x00,    /* ----- */
2906:      0x00,    /* ----- */
2907:      0x18,    /* ---##--- */
2908:      0x18,    /* ---##--- */
2909:      0x00,    /* ----- */
2910:      0x18,    /* ---##--- */
2911:      0x18,    /* ---##--- */
2912:      0x18,    /* ---##--- */
2913:      0x3c,    /* --####-- */
2914:      0x3c,    /* --####-- */
2915:      0x3c,    /* --####-- */
2916:      0x18,    /* ---##--- */
2917:      0x00,    /* ----- */
2918:      0x00,    /* ----- */
2919:
2920:          /* 0xa2 */
2921:      0x00,    /* ----- */
2922:      0x00,    /* ----- */
2923:      0x00,    /* ----- */
2924:      0x00,    /* ----- */
2925:      0x10,    /* ---#---- */
2926:      0x7c,    /* -#####-- */
2927:      0xd6,    /* ##-##-##- */
2928:      0xd0,    /* ##-##-##- */
2929:      0xd0,    /* ##-##-##- */
2930:      0xd0,    /* ##-##-##- */
2931:      0xd6,    /* ##-##-##- */
2932:      0x7c,    /* -#####-- */
2933:      0x10,    /* ---#---- */
2934:      0x00,    /* ----- */
2935:      0x00,    /* ----- */
2936:      0x00,    /* ----- */
2937:
2938:          /* 0xa3 */
2939:      0x00,    /* ----- */
2940:      0x00,    /* ----- */
2941:      0x38,    /* --###--- */
2942:      0x6c,    /* -##-##-- */
2943:      0x60,    /* -##----- */
2944:      0x60,    /* -##----- */
2945:      0xf0,    /* #####----- */
2946:      0x60,    /* -##----- */
2947:      0x60,    /* -##----- */
2948:      0x66,    /* -##--##- */

```

drivers/video/font-lat9-8x16.c

Page 45/70

```

2949:      0xf6, /* ####-##- */
2950:      0x6c, /* -##-##-- */
2951:      0x00, /* ----- */
2952:      0x00, /* ----- */
2953:      0x00, /* ----- */
2954:      0x00, /* ----- */
2955:
2956:          /* 0xa4 */
2957:      0x00, /* ----- */
2958:      0x1c, /* ---###-- */
2959:      0x32, /* --##-##- */
2960:      0x60, /* -##----- */
2961:      0x60, /* -##----- */
2962:      0xfc, /* #####--- */
2963:      0x60, /* -##----- */
2964:      0xfc, /* #####--- */
2965:      0x60, /* -##----- */
2966:      0x60, /* -##----- */
2967:      0x32, /* --##-##- */
2968:      0x1c, /* ---###-- */
2969:      0x00, /* ----- */
2970:      0x00, /* ----- */
2971:      0x00, /* ----- */
2972:      0x00, /* ----- */
2973:
2974:          /* 0xa5 */
2975:      0x00, /* ----- */
2976:      0x00, /* ----- */
2977:      0x66, /* -##-##- */
2978:      0x66, /* -##-##- */
2979:      0x3c, /* --####-- */
2980:      0x18, /* ---##--- */
2981:      0x7e, /* -#####- */
2982:      0x18, /* ---##--- */
2983:      0x7e, /* -#####- */
2984:      0x18, /* ---##--- */
2985:      0x18, /* ---##--- */
2986:      0x18, /* ---##--- */
2987:      0x00, /* ----- */
2988:      0x00, /* ----- */
2989:      0x00, /* ----- */
2990:      0x00, /* ----- */
2991:
2992:          /* 0xa6 */
2993:      0x6c, /* -##-##-- */
2994:      0x38, /* --###--- */
2995:      0x00, /* ----- */
2996:      0x7c, /* -#####- */
2997:      0xc6, /* ##---##- */
2998:      0xc6, /* ##---##- */
2999:      0x60, /* -##----- */
3000:      0x38, /* --###--- */
3001:      0x0c, /* ----##-- */
3002:      0xc6, /* ##---##- */
3003:      0xc6, /* ##---##- */
3004:      0x7c, /* -#####- */
3005:      0x00, /* ----- */
3006:      0x00, /* ----- */
3007:      0x00, /* ----- */
3008:      0x00, /* ----- */
3009:
3010:          /* 0xa7 */
3011:      0x00, /* ----- */
3012:      0x7c, /* -#####- */
3013:      0xc6, /* ##---##- */
3014:      0x60, /* -##----- */
3015:      0x38, /* --###--- */

```

drivers/video/font-lat9-8x16.c

Page 46/70

```

3016:      0x6c,    /* ---##--- */
3017:      0xc6,    /* ##----##- */
3018:      0xc6,    /* ##----##- */
3019:      0x6c,    /* ---##--- */
3020:      0x38,    /* ---###--- */
3021:      0x0c,    /* ----##-- */
3022:      0xc6,    /* ##----##- */
3023:      0x7c,    /* -#####-- */
3024:      0x00,    /* ----- */
3025:      0x00,    /* ----- */
3026:      0x00,    /* ----- */
3027:
3028:          /* 0xa8 */
3029:      0x00,    /* ----- */
3030:      0x6c,    /* ---##--- */
3031:      0x38,    /* ---###--- */
3032:      0x00,    /* ----- */
3033:      0x00,    /* ----- */
3034:      0x7c,    /* -#####-- */
3035:      0xc6,    /* ##----##- */
3036:      0x60,    /* -##----- */
3037:      0x38,    /* ---##--- */
3038:      0x0c,    /* ----##-- */
3039:      0xc6,    /* ##----##- */
3040:      0x7c,    /* -#####-- */
3041:      0x00,    /* ----- */
3042:      0x00,    /* ----- */
3043:      0x00,    /* ----- */
3044:      0x00,    /* ----- */
3045:
3046:          /* 0xa9 */
3047:      0x00,    /* ----- */
3048:      0x00,    /* ----- */
3049:      0x3c,    /* ---###--- */
3050:      0x42,    /* -#----#- */
3051:      0x99,    /* #--##--# */
3052:      0xa5,    /* #-#--#-# */
3053:      0xa1,    /* #-#----# */
3054:      0xa5,    /* #-#--#-# */
3055:      0x99,    /* #--##--# */
3056:      0x42,    /* -#----#- */
3057:      0x3c,    /* ---###--- */
3058:      0x00,    /* ----- */
3059:      0x00,    /* ----- */
3060:      0x00,    /* ----- */
3061:      0x00,    /* ----- */
3062:      0x00,    /* ----- */
3063:
3064:          /* 0xaa */
3065:      0x00,    /* ----- */
3066:      0x00,    /* ----- */
3067:      0x3c,    /* ---###--- */
3068:      0x6c,    /* ---##--- */
3069:      0x6c,    /* ---##--- */
3070:      0x3e,    /* ---#####- */
3071:      0x00,    /* ----- */
3072:      0x7e,    /* -#####-# */
3073:      0x00,    /* ----- */
3074:      0x00,    /* ----- */
3075:      0x00,    /* ----- */
3076:      0x00,    /* ----- */
3077:      0x00,    /* ----- */
3078:      0x00,    /* ----- */
3079:      0x00,    /* ----- */
3080:      0x00,    /* ----- */
3081:
3082:          /* 0xab */

```

drivers/video/font-lat9-8x16.c

Page 47/70

```

3083:      0x00, /* ----- */
3084:      0x00, /* ----- */
3085:      0x00, /* ----- */
3086:      0x00, /* ----- */
3087:      0x00, /* ----- */
3088:      0x36, /* --##-##- */
3089:      0x6c, /* -##-##- */
3090:      0xd8, /* ##-##- */
3091:      0x6c, /* -##-##- */
3092:      0x36, /* --##-##- */
3093:      0x00, /* ----- */
3094:      0x00, /* ----- */
3095:      0x00, /* ----- */
3096:      0x00, /* ----- */
3097:      0x00, /* ----- */
3098:      0x00, /* ----- */
3099:
3100:          /* 0xac */
3101:      0x00, /* ----- */
3102:      0x00, /* ----- */
3103:      0x00, /* ----- */
3104:      0x00, /* ----- */
3105:      0x00, /* ----- */
3106:      0x00, /* ----- */
3107:      0xfe, /* #####- */
3108:      0x06, /* -----##- */
3109:      0x06, /* -----##- */
3110:      0x06, /* -----##- */
3111:      0x06, /* -----##- */
3112:      0x00, /* ----- */
3113:      0x00, /* ----- */
3114:      0x00, /* ----- */
3115:      0x00, /* ----- */
3116:      0x00, /* ----- */
3117:
3118:          /* 0xad */
3119:      0x00, /* ----- */
3120:      0x00, /* ----- */
3121:      0x00, /* ----- */
3122:      0x00, /* ----- */
3123:      0x00, /* ----- */
3124:      0x00, /* ----- */
3125:      0x00, /* ----- */
3126:      0x7e, /* -#####- */
3127:      0x00, /* ----- */
3128:      0x00, /* ----- */
3129:      0x00, /* ----- */
3130:      0x00, /* ----- */
3131:      0x00, /* ----- */
3132:      0x00, /* ----- */
3133:      0x00, /* ----- */
3134:      0x00, /* ----- */
3135:
3136:          /* 0xae */
3137:      0x00, /* ----- */
3138:      0x00, /* ----- */
3139:      0x3c, /* --####- */
3140:      0x42, /* -#----#- */
3141:      0xb9, /* #-###-#- */
3142:      0xa5, /* #-#--#-# */
3143:      0xb9, /* #-###-#- */
3144:      0xa5, /* #-#--#-# */
3145:      0xa5, /* #-#--#-# */
3146:      0x42, /* -#----#- */
3147:      0x3c, /* --####- */
3148:      0x00, /* ----- */
3149:      0x00, /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 48/70

```

3150:      0x00, /* ----- */
3151:      0x00, /* ----- */
3152:      0x00, /* ----- */
3153:
3154:          /* 0xaf */
3155:      0xff, /* ##### */
3156:      0x00, /* ----- */
3157:      0x00, /* ----- */
3158:      0x00, /* ----- */
3159:      0x00, /* ----- */
3160:      0x00, /* ----- */
3161:      0x00, /* ----- */
3162:      0x00, /* ----- */
3163:      0x00, /* ----- */
3164:      0x00, /* ----- */
3165:      0x00, /* ----- */
3166:      0x00, /* ----- */
3167:      0x00, /* ----- */
3168:      0x00, /* ----- */
3169:      0x00, /* ----- */
3170:      0x00, /* ----- */
3171:
3172:          /* 0xb0 */
3173:      0x00, /* ----- */
3174:      0x38, /* ---###--- */
3175:      0x6c, /* -##-##- */
3176:      0x6c, /* -##-##- */
3177:      0x38, /* ---###--- */
3178:      0x00, /* ----- */
3179:      0x00, /* ----- */
3180:      0x00, /* ----- */
3181:      0x00, /* ----- */
3182:      0x00, /* ----- */
3183:      0x00, /* ----- */
3184:      0x00, /* ----- */
3185:      0x00, /* ----- */
3186:      0x00, /* ----- */
3187:      0x00, /* ----- */
3188:      0x00, /* ----- */
3189:
3190:          /* 0xb1 */
3191:      0x00, /* ----- */
3192:      0x00, /* ----- */
3193:      0x00, /* ----- */
3194:      0x00, /* ----- */
3195:      0x00, /* ----- */
3196:      0x18, /* ---##--- */
3197:      0x18, /* ---##--- */
3198:      0x7e, /* -#####- */
3199:      0x18, /* ---##--- */
3200:      0x18, /* ---##--- */
3201:      0x00, /* ----- */
3202:      0x7e, /* -#####- */
3203:      0x00, /* ----- */
3204:      0x00, /* ----- */
3205:      0x00, /* ----- */
3206:      0x00, /* ----- */
3207:
3208:          /* 0xb2 */
3209:      0x38, /* ---###--- */
3210:      0x6c, /* -##-##- */
3211:      0x18, /* ---##--- */
3212:      0x30, /* --##---- */
3213:      0x7c, /* -#####- */
3214:      0x00, /* ----- */
3215:      0x00, /* ----- */
3216:      0x00, /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 49/70

```

3217:      0x00, /* ----- */
3218:      0x00, /* ----- */
3219:      0x00, /* ----- */
3220:      0x00, /* ----- */
3221:      0x00, /* ----- */
3222:      0x00, /* ----- */
3223:      0x00, /* ----- */
3224:      0x00, /* ----- */
3225:
3226:      /* 0xb3 */
3227:      0x38, /* ---###--- */
3228:      0x6c, /* -##-##- */
3229:      0x18, /* ---##--- */
3230:      0x6c, /* -##-##- */
3231:      0x38, /* ---###--- */
3232:      0x00, /* ----- */
3233:      0x00, /* ----- */
3234:      0x00, /* ----- */
3235:      0x00, /* ----- */
3236:      0x00, /* ----- */
3237:      0x00, /* ----- */
3238:      0x00, /* ----- */
3239:      0x00, /* ----- */
3240:      0x00, /* ----- */
3241:      0x00, /* ----- */
3242:      0x00, /* ----- */
3243:
3244:      /* 0xb4 */
3245:      0x6c, /* -##-##- */
3246:      0x38, /* ---###--- */
3247:      0x00, /* ----- */
3248:      0xfe, /* #####- */
3249:      0xc6, /* ##---##- */
3250:      0x8c, /* #---##- */
3251:      0x18, /* ---##--- */
3252:      0x30, /* --##---- */
3253:      0x60, /* -##----- */
3254:      0xc2, /* ##-----#- */
3255:      0xc6, /* ##---##- */
3256:      0xfe, /* #####- */
3257:      0x00, /* ----- */
3258:      0x00, /* ----- */
3259:      0x00, /* ----- */
3260:      0x00, /* ----- */
3261:
3262:      /* 0xb5 */
3263:      0x00, /* ----- */
3264:      0x00, /* ----- */
3265:      0x00, /* ----- */
3266:      0x00, /* ----- */
3267:      0x00, /* ----- */
3268:      0xcc, /* ##---##- */
3269:      0xcc, /* ##---##- */
3270:      0xcc, /* ##---##- */
3271:      0xcc, /* ##---##- */
3272:      0xcc, /* ##---##- */
3273:      0xcc, /* ##---##- */
3274:      0xf6, /* #####-##- */
3275:      0xc0, /* ##----- */
3276:      0xc0, /* ##----- */
3277:      0xc0, /* ##----- */
3278:      0x00, /* ----- */
3279:
3280:      /* 0xb6 */
3281:      0x00, /* ----- */
3282:      0x00, /* ----- */
3283:      0x7f, /* -##### */

```

drivers/video/font-lat9-8x16.c

Page 50/70

```

3284:      0xd6,      /* ##-##-##- */
3285:      0xd6,      /* ##-##-##- */
3286:      0x76,      /* -###-##- */
3287:      0x36,      /* --##-##- */
3288:      0x36,      /* --##-##- */
3289:      0x36,      /* --##-##- */
3290:      0x36,      /* --##-##- */
3291:      0x36,      /* --##-##- */
3292:      0x36,      /* --##-##- */
3293:      0x00,      /* ----- */
3294:      0x00,      /* ----- */
3295:      0x00,      /* ----- */
3296:      0x00,      /* ----- */
3297:
3298:          /* 0xb7          */
3299:      0x00,      /* ----- */
3300:      0x00,      /* ----- */
3301:      0x00,      /* ----- */
3302:      0x00,      /* ----- */
3303:      0x00,      /* ----- */
3304:      0x00,      /* ----- */
3305:      0x18,      /* ---##- */
3306:      0x18,      /* ---##- */
3307:      0x00,      /* ----- */
3308:      0x00,      /* ----- */
3309:      0x00,      /* ----- */
3310:      0x00,      /* ----- */
3311:      0x00,      /* ----- */
3312:      0x00,      /* ----- */
3313:      0x00,      /* ----- */
3314:      0x00,      /* ----- */
3315:
3316:          /* 0xb8          */
3317:      0x00,      /* ----- */
3318:      0x00,      /* ----- */
3319:      0x6c,      /* -##-##- */
3320:      0x38,      /* --##- */
3321:      0x00,      /* ----- */
3322:      0xfe,      /* #####- */
3323:      0xcc,      /* ##-##- */
3324:      0x18,      /* ---##- */
3325:      0x30,      /* --##- */
3326:      0x60,      /* -##- */
3327:      0xc6,      /* ##-##- */
3328:      0xfe,      /* #####- */
3329:      0x00,      /* ----- */
3330:      0x00,      /* ----- */
3331:      0x00,      /* ----- */
3332:      0x00,      /* ----- */
3333:
3334:          /* 0xb9          */
3335:      0x30,      /* ---##- */
3336:      0x70,      /* -##- */
3337:      0x30,      /* --##- */
3338:      0x30,      /* --##- */
3339:      0x78,      /* -####- */
3340:      0x00,      /* ----- */
3341:      0x00,      /* ----- */
3342:      0x00,      /* ----- */
3343:      0x00,      /* ----- */
3344:      0x00,      /* ----- */
3345:      0x00,      /* ----- */
3346:      0x00,      /* ----- */
3347:      0x00,      /* ----- */
3348:      0x00,      /* ----- */
3349:      0x00,      /* ----- */
3350:      0x00,      /* ----- */

```


drivers/video/font-lat9-8x16.c

Page 51/70

```

3351:
3352:          /* 0xba          */
3353:    0x00,  /* ----- */
3354:    0x00,  /* ----- */
3355:    0x38,  /* ---###--- */
3356:    0x6c,  /* -##-##-  */
3357:    0x6c,  /* -##-##-  */
3358:    0x38,  /* ---###--- */
3359:    0x00,  /* ----- */
3360:    0x7c,  /* -#####-  */
3361:    0x00,  /* ----- */
3362:    0x00,  /* ----- */
3363:    0x00,  /* ----- */
3364:    0x00,  /* ----- */
3365:    0x00,  /* ----- */
3366:    0x00,  /* ----- */
3367:    0x00,  /* ----- */
3368:    0x00,  /* ----- */
3369:
3370:          /* 0xbb          */
3371:    0x00,  /* ----- */
3372:    0x00,  /* ----- */
3373:    0x00,  /* ----- */
3374:    0x00,  /* ----- */
3375:    0x00,  /* ----- */
3376:    0xd8,  /* ##-##-   */
3377:    0x6c,  /* -##-##-  */
3378:    0x36,  /* --##-##- */
3379:    0x6c,  /* -##-##-  */
3380:    0xd8,  /* ##-##-   */
3381:    0x00,  /* ----- */
3382:    0x00,  /* ----- */
3383:    0x00,  /* ----- */
3384:    0x00,  /* ----- */
3385:    0x00,  /* ----- */
3386:    0x00,  /* ----- */
3387:
3388:          /* 0xbc          */
3389:    0x00,  /* ----- */
3390:    0x00,  /* ----- */
3391:    0x77,  /* -###-### */
3392:    0xcc,  /* ##-##-  */
3393:    0xcc,  /* ##-##-  */
3394:    0xcc,  /* ##-##-  */
3395:    0xcf,  /* ##-#### */
3396:    0xcf,  /* ##-#### */
3397:    0xcc,  /* ##-##-  */
3398:    0xcc,  /* ##-##-  */
3399:    0xcc,  /* ##-##-  */
3400:    0x77,  /* -###-### */
3401:    0x00,  /* ----- */
3402:    0x00,  /* ----- */
3403:    0x00,  /* ----- */
3404:    0x00,  /* ----- */
3405:
3406:          /* 0xbd          */
3407:    0x00,  /* ----- */
3408:    0x00,  /* ----- */
3409:    0x00,  /* ----- */
3410:    0x00,  /* ----- */
3411:    0x00,  /* ----- */
3412:    0x6e,  /* -##-###- */
3413:    0xdb,  /* ##-##-## */
3414:    0xdb,  /* ##-##-## */
3415:    0xdf,  /* ##-#### */
3416:    0xd8,  /* ##-##-   */
3417:    0xdb,  /* ##-##-## */

```

drivers/video/font-lat9-8x16.c

Page 52/70

```

3418:      0x6e,    /* ---###- */
3419:      0x00,    /* ----- */
3420:      0x00,    /* ----- */
3421:      0x00,    /* ----- */
3422:      0x00,    /* ----- */
3423:
3424:          /* 0xbe */
3425:      0x00,    /* ----- */
3426:      0x66,    /* ---###- */
3427:      0x00,    /* ----- */
3428:      0x66,    /* ---###- */
3429:      0x66,    /* ---###- */
3430:      0x66,    /* ---###- */
3431:      0x66,    /* ---###- */
3432:      0x3c,    /* ---###- */
3433:      0x18,    /* ---##- */
3434:      0x18,    /* ---##- */
3435:      0x18,    /* ---##- */
3436:      0x3c,    /* ---###- */
3437:      0x00,    /* ----- */
3438:      0x00,    /* ----- */
3439:      0x00,    /* ----- */
3440:      0x00,    /* ----- */
3441:
3442:          /* 0xbf */
3443:      0x00,    /* ----- */
3444:      0x00,    /* ----- */
3445:      0x00,    /* ----- */
3446:      0x00,    /* ----- */
3447:      0x30,    /* ---##- */
3448:      0x30,    /* ---##- */
3449:      0x00,    /* ----- */
3450:      0x30,    /* ---##- */
3451:      0x30,    /* ---##- */
3452:      0x30,    /* ---##- */
3453:      0x60,    /* ---##- */
3454:      0xc6,    /* ##---##- */
3455:      0xc6,    /* ##---##- */
3456:      0x7c,    /* ---###- */
3457:      0x00,    /* ----- */
3458:      0x00,    /* ----- */
3459:
3460:          /* 0xc0 */
3461:      0x60,    /* ---##- */
3462:      0x30,    /* ---##- */
3463:      0x00,    /* ----- */
3464:      0x38,    /* ---###- */
3465:      0x6c,    /* ---##-##- */
3466:      0xc6,    /* ##---##- */
3467:      0xc6,    /* ##---##- */
3468:      0xfe,    /* #####- */
3469:      0xc6,    /* ##---##- */
3470:      0xc6,    /* ##---##- */
3471:      0xc6,    /* ##---##- */
3472:      0xc6,    /* ##---##- */
3473:      0x00,    /* ----- */
3474:      0x00,    /* ----- */
3475:      0x00,    /* ----- */
3476:      0x00,    /* ----- */
3477:
3478:          /* 0xc1 */
3479:      0x0c,    /* ----##- */
3480:      0x18,    /* ---##- */
3481:      0x00,    /* ----- */
3482:      0x38,    /* ---###- */
3483:      0x6c,    /* ---##-##- */
3484:      0xc6,    /* ##---##- */

```

drivers/video/font-lat9-8x16.c

Page 53/70

```

3485:      0xc6,      /* ##---##- */
3486:      0xfe,      /* #####- */
3487:      0xc6,      /* ##---##- */
3488:      0xc6,      /* ##---##- */
3489:      0xc6,      /* ##---##- */
3490:      0xc6,      /* ##---##- */
3491:      0x00,      /* ----- */
3492:      0x00,      /* ----- */
3493:      0x00,      /* ----- */
3494:      0x00,      /* ----- */
3495:
3496:          /* 0xc2          */
3497:      0x10,      /* ---#---- */
3498:      0x38,      /* --###--- */
3499:      0x6c,      /* -##-##-- */
3500:      0x00,      /* ----- */
3501:      0x38,      /* --###--- */
3502:      0x6c,      /* -##-##-- */
3503:      0xc6,      /* ##---##- */
3504:      0xc6,      /* ##---##- */
3505:      0xfe,      /* #####- */
3506:      0xc6,      /* ##---##- */
3507:      0xc6,      /* ##---##- */
3508:      0xc6,      /* ##---##- */
3509:      0x00,      /* ----- */
3510:      0x00,      /* ----- */
3511:      0x00,      /* ----- */
3512:      0x00,      /* ----- */
3513:
3514:          /* 0xc3          */
3515:      0x76,      /* -###-##- */
3516:      0xdc,      /* ##-##-#- */
3517:      0x00,      /* ----- */
3518:      0x38,      /* --###--- */
3519:      0x6c,      /* -##-##-- */
3520:      0xc6,      /* ##---##- */
3521:      0xc6,      /* ##---##- */
3522:      0xfe,      /* #####- */
3523:      0xc6,      /* ##---##- */
3524:      0xc6,      /* ##---##- */
3525:      0xc6,      /* ##---##- */
3526:      0xc6,      /* ##---##- */
3527:      0x00,      /* ----- */
3528:      0x00,      /* ----- */
3529:      0x00,      /* ----- */
3530:      0x00,      /* ----- */
3531:
3532:          /* 0xc4          */
3533:      0x00,      /* ----- */
3534:      0x6c,      /* -##-##-- */
3535:      0x00,      /* ----- */
3536:      0x38,      /* --###--- */
3537:      0x6c,      /* -##-##-- */
3538:      0xc6,      /* ##---##- */
3539:      0xc6,      /* ##---##- */
3540:      0xfe,      /* #####- */
3541:      0xc6,      /* ##---##- */
3542:      0xc6,      /* ##---##- */
3543:      0xc6,      /* ##---##- */
3544:      0xc6,      /* ##---##- */
3545:      0x00,      /* ----- */
3546:      0x00,      /* ----- */
3547:      0x00,      /* ----- */
3548:      0x00,      /* ----- */
3549:
3550:          /* 0xc5          */
3551:      0x38,      /* --###--- */

```

drivers/video/font-lat9-8x16.c

Page 54/70

```

3552:      0x6c,    /* --##-##-- */
3553:      0x38,    /* ---###---- */
3554:      0x00,    /* ----- */
3555:      0x38,    /* --###--- */
3556:      0x6c,    /* -##-##-- */
3557:      0xc6,    /* ##---##- */
3558:      0xc6,    /* ##---##- */
3559:      0xfe,    /* #####- */
3560:      0xc6,    /* ##---##- */
3561:      0xc6,    /* ##---##- */
3562:      0xc6,    /* ##---##- */
3563:      0x00,    /* ----- */
3564:      0x00,    /* ----- */
3565:      0x00,    /* ----- */
3566:      0x00,    /* ----- */
3567:
3568:          /* 0xc6 */
3569:      0x00,    /* ----- */
3570:      0x00,    /* ----- */
3571:      0x3e,    /* ---#####- */
3572:      0x78,    /* -#####- */
3573:      0xd8,    /* ##-##---- */
3574:      0xd8,    /* ##-##---- */
3575:      0xfc,    /* #####- */
3576:      0xd8,    /* ##-##---- */
3577:      0xd8,    /* ##-##---- */
3578:      0xd8,    /* ##-##---- */
3579:      0xd8,    /* ##-##---- */
3580:      0xde,    /* ##-#####- */
3581:      0x00,    /* ----- */
3582:      0x00,    /* ----- */
3583:      0x00,    /* ----- */
3584:      0x00,    /* ----- */
3585:
3586:          /* 0xc7 */
3587:      0x00,    /* ----- */
3588:      0x00,    /* ----- */
3589:      0x3c,    /* --#####- */
3590:      0x66,    /* -##-##- */
3591:      0xc2,    /* ##----#- */
3592:      0xc0,    /* ##------ */
3593:      0xc0,    /* ##------ */
3594:      0xc0,    /* ##------ */
3595:      0xc0,    /* ##------ */
3596:      0xc2,    /* ##----#- */
3597:      0x66,    /* -##-##- */
3598:      0x3c,    /* --#####- */
3599:      0x0c,    /* ----##- */
3600:      0x66,    /* -##-##- */
3601:      0x3c,    /* --#####- */
3602:      0x00,    /* ----- */
3603:
3604:          /* 0xc8 */
3605:      0x60,    /* -##------ */
3606:      0x30,    /* ---##------ */
3607:      0x00,    /* ----- */
3608:      0xfe,    /* #####- */
3609:      0x66,    /* -##-##- */
3610:      0x60,    /* -##------ */
3611:      0x60,    /* -##------ */
3612:      0x7c,    /* -#####- */
3613:      0x60,    /* -##------ */
3614:      0x60,    /* -##------ */
3615:      0x66,    /* -##-##- */
3616:      0xfe,    /* #####- */
3617:      0x00,    /* ----- */
3618:      0x00,    /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 55/70

```

3619:      0x00, /* ----- */
3620:      0x00, /* ----- */
3621:
3622:          /* 0xc9 */
3623:      0x0c, /* ----##-- */
3624:      0x18, /* ---##--- */
3625:      0x00, /* ----- */
3626:      0xfe, /* #####- */
3627:      0x66, /* -##-##- */
3628:      0x60, /* -##----- */
3629:      0x60, /* -##----- */
3630:      0x7c, /* -#####- */
3631:      0x60, /* -##----- */
3632:      0x60, /* -##----- */
3633:      0x66, /* -##-##- */
3634:      0xfe, /* #####- */
3635:      0x00, /* ----- */
3636:      0x00, /* ----- */
3637:      0x00, /* ----- */
3638:      0x00, /* ----- */
3639:
3640:          /* 0xca */
3641:      0x10, /* ---#----- */
3642:      0x38, /* --###--- */
3643:      0x6c, /* -##-##- */
3644:      0x00, /* ----- */
3645:      0xfe, /* #####- */
3646:      0x66, /* -##-##- */
3647:      0x60, /* -##----- */
3648:      0x7c, /* -#####- */
3649:      0x60, /* -##----- */
3650:      0x60, /* -##----- */
3651:      0x66, /* -##-##- */
3652:      0xfe, /* #####- */
3653:      0x00, /* ----- */
3654:      0x00, /* ----- */
3655:      0x00, /* ----- */
3656:      0x00, /* ----- */
3657:
3658:          /* 0xcb */
3659:      0x00, /* ----- */
3660:      0x6c, /* -##-##- */
3661:      0x00, /* ----- */
3662:      0xfe, /* #####- */
3663:      0x66, /* -##-##- */
3664:      0x60, /* -##----- */
3665:      0x60, /* -##----- */
3666:      0x7c, /* -#####- */
3667:      0x60, /* -##----- */
3668:      0x60, /* -##----- */
3669:      0x66, /* -##-##- */
3670:      0xfe, /* #####- */
3671:      0x00, /* ----- */
3672:      0x00, /* ----- */
3673:      0x00, /* ----- */
3674:      0x00, /* ----- */
3675:
3676:          /* 0xcc */
3677:      0x60, /* -##----- */
3678:      0x30, /* --##----- */
3679:      0x00, /* ----- */
3680:      0x3c, /* --####--- */
3681:      0x18, /* ----##--- */
3682:      0x18, /* ----##--- */
3683:      0x18, /* ----##--- */
3684:      0x18, /* ----##--- */
3685:      0x18, /* ----##--- */

```

drivers/video/font-lat9-8x16.c

Page 56/70

```

3686:      0x18,    /* ----##---- */
3687:      0x18,    /* ----##---- */
3688:      0x3c,    /* --####-- */
3689:      0x00,    /* ----- */
3690:      0x00,    /* ----- */
3691:      0x00,    /* ----- */
3692:      0x00,    /* ----- */
3693:
3694:          /* 0xcd          */
3695:      0x06,    /* -----##- */
3696:      0x0c,    /* ----##---- */
3697:      0x00,    /* ----- */
3698:      0x3c,    /* --####-- */
3699:      0x18,    /* ----##---- */
3700:      0x18,    /* ----##---- */
3701:      0x18,    /* ----##---- */
3702:      0x18,    /* ----##---- */
3703:      0x18,    /* ----##---- */
3704:      0x18,    /* ----##---- */
3705:      0x18,    /* ----##---- */
3706:      0x3c,    /* --####-- */
3707:      0x00,    /* ----- */
3708:      0x00,    /* ----- */
3709:      0x00,    /* ----- */
3710:      0x00,    /* ----- */
3711:
3712:          /* 0xce          */
3713:      0x18,    /* ----##---- */
3714:      0x3c,    /* --####-- */
3715:      0x66,    /* -##--##- */
3716:      0x00,    /* ----- */
3717:      0x3c,    /* --####-- */
3718:      0x18,    /* ----##---- */
3719:      0x18,    /* ----##---- */
3720:      0x18,    /* ----##---- */
3721:      0x18,    /* ----##---- */
3722:      0x18,    /* ----##---- */
3723:      0x18,    /* ----##---- */
3724:      0x3c,    /* --####-- */
3725:      0x00,    /* ----- */
3726:      0x00,    /* ----- */
3727:      0x00,    /* ----- */
3728:      0x00,    /* ----- */
3729:
3730:          /* 0xcf          */
3731:      0x00,    /* ----- */
3732:      0x66,    /* -##--##- */
3733:      0x00,    /* ----- */
3734:      0x3c,    /* --####-- */
3735:      0x18,    /* ----##---- */
3736:      0x18,    /* ----##---- */
3737:      0x18,    /* ----##---- */
3738:      0x18,    /* ----##---- */
3739:      0x18,    /* ----##---- */
3740:      0x18,    /* ----##---- */
3741:      0x18,    /* ----##---- */
3742:      0x3c,    /* --####-- */
3743:      0x00,    /* ----- */
3744:      0x00,    /* ----- */
3745:      0x00,    /* ----- */
3746:      0x00,    /* ----- */
3747:
3748:          /* 0xd0          */
3749:      0x00,    /* ----- */
3750:      0x00,    /* ----- */
3751:      0xf8,    /* #####---- */
3752:      0x6c,    /* -##-##- */

```

drivers/video/font-lat9-8x16.c

Page 57/70

```

3753:      0x66,    /* ---##--- */
3754:      0x66,    /* ---##--- */
3755:      0xf6,    /* #####--- */
3756:      0x66,    /* ---##--- */
3757:      0x66,    /* ---##--- */
3758:      0x66,    /* ---##--- */
3759:      0x6c,    /* ---##--- */
3760:      0xf8,    /* #####--- */
3761:      0x00,    /* ----- */
3762:      0x00,    /* ----- */
3763:      0x00,    /* ----- */
3764:      0x00,    /* ----- */
3765:
3766:          /* 0xd1 */
3767:      0x76,    /* ---##--- */
3768:      0xdc,    /* ##-###--- */
3769:      0x00,    /* ----- */
3770:      0xc6,    /* ##---##- */
3771:      0xe6,    /* ###---##- */
3772:      0xf6,    /* #####--- */
3773:      0xfe,    /* ########- */
3774:      0xde,    /* ##-#####- */
3775:      0xce,    /* ##---##--- */
3776:      0xc6,    /* ##---##- */
3777:      0xc6,    /* ##---##- */
3778:      0xc6,    /* ##---##- */
3779:      0x00,    /* ----- */
3780:      0x00,    /* ----- */
3781:      0x00,    /* ----- */
3782:      0x00,    /* ----- */
3783:
3784:          /* 0xd2 */
3785:      0x60,    /* ---##----- */
3786:      0x30,    /* --##----- */
3787:      0x00,    /* ----- */
3788:      0x7c,    /* -#####--- */
3789:      0xc6,    /* ##---##--- */
3790:      0xc6,    /* ##---##--- */
3791:      0xc6,    /* ##---##--- */
3792:      0xc6,    /* ##---##--- */
3793:      0xc6,    /* ##---##--- */
3794:      0xc6,    /* ##---##--- */
3795:      0xc6,    /* ##---##--- */
3796:      0x7c,    /* -#####--- */
3797:      0x00,    /* ----- */
3798:      0x00,    /* ----- */
3799:      0x00,    /* ----- */
3800:      0x00,    /* ----- */
3801:
3802:          /* 0xd3 */
3803:      0x0c,    /* ----##--- */
3804:      0x18,    /* ----##--- */
3805:      0x00,    /* ----- */
3806:      0x7c,    /* -#####--- */
3807:      0xc6,    /* ##---##--- */
3808:      0xc6,    /* ##---##--- */
3809:      0xc6,    /* ##---##--- */
3810:      0xc6,    /* ##---##--- */
3811:      0xc6,    /* ##---##--- */
3812:      0xc6,    /* ##---##--- */
3813:      0xc6,    /* ##---##--- */
3814:      0x7c,    /* -#####--- */
3815:      0x00,    /* ----- */
3816:      0x00,    /* ----- */
3817:      0x00,    /* ----- */
3818:      0x00,    /* ----- */
3819:

```

drivers/video/font-lat9-8x16.c

Page 58/70

```

3820:                /* 0xd4          */
3821:                0x10, /* ----#---- */
3822:                0x38, /* --###--- */
3823:                0x6c, /* -##-##- */
3824:                0x00, /* ----- */
3825:                0x7c, /* -#####- */
3826:                0xc6, /* ##---##- */
3827:                0xc6, /* ##---##- */
3828:                0xc6, /* ##---##- */
3829:                0xc6, /* ##---##- */
3830:                0xc6, /* ##---##- */
3831:                0xc6, /* ##---##- */
3832:                0x7c, /* -#####- */
3833:                0x00, /* ----- */
3834:                0x00, /* ----- */
3835:                0x00, /* ----- */
3836:                0x00, /* ----- */
3837:
3838:                /* 0xd5          */
3839:                0x76, /* -###-##- */
3840:                0xdc, /* #-###--- */
3841:                0x00, /* ----- */
3842:                0x7c, /* -#####- */
3843:                0xc6, /* ##---##- */
3844:                0xc6, /* ##---##- */
3845:                0xc6, /* ##---##- */
3846:                0xc6, /* ##---##- */
3847:                0xc6, /* ##---##- */
3848:                0xc6, /* ##---##- */
3849:                0xc6, /* ##---##- */
3850:                0x7c, /* -#####- */
3851:                0x00, /* ----- */
3852:                0x00, /* ----- */
3853:                0x00, /* ----- */
3854:                0x00, /* ----- */
3855:
3856:                /* 0xd6          */
3857:                0x00, /* ----- */
3858:                0x6c, /* -##-##- */
3859:                0x00, /* ----- */
3860:                0x7c, /* -#####- */
3861:                0xc6, /* ##---##- */
3862:                0xc6, /* ##---##- */
3863:                0xc6, /* ##---##- */
3864:                0xc6, /* ##---##- */
3865:                0xc6, /* ##---##- */
3866:                0xc6, /* ##---##- */
3867:                0xc6, /* ##---##- */
3868:                0x7c, /* -#####- */
3869:                0x00, /* ----- */
3870:                0x00, /* ----- */
3871:                0x00, /* ----- */
3872:                0x00, /* ----- */
3873:
3874:                /* 0xd7          */
3875:                0x00, /* ----- */
3876:                0x00, /* ----- */
3877:                0x00, /* ----- */
3878:                0x00, /* ----- */
3879:                0x00, /* ----- */
3880:                0x66, /* -##-##- */
3881:                0x3c, /* --###--- */
3882:                0x18, /* ---##--- */
3883:                0x3c, /* --###--- */
3884:                0x66, /* -##-##- */
3885:                0x00, /* ----- */
3886:                0x00, /* ----- */

```


drivers/video/font-lat9-8x16.c

Page 59/70

```

3887:      0x00, /* ----- */
3888:      0x00, /* ----- */
3889:      0x00, /* ----- */
3890:      0x00, /* ----- */
3891:
3892:          /* 0xd8 */
3893:      0x00, /* ----- */
3894:      0x00, /* ----- */
3895:      0x7e, /* -#####- */
3896:      0xc6, /* ##---##- */
3897:      0xce, /* ##---##- */
3898:      0xce, /* ##---##- */
3899:      0xde, /* ##---##- */
3900:      0xf6, /* #####-##- */
3901:      0xe6, /* ###---##- */
3902:      0xe6, /* ###---##- */
3903:      0xc6, /* ##---##- */
3904:      0xfc, /* #####--- */
3905:      0x00, /* ----- */
3906:      0x00, /* ----- */
3907:      0x00, /* ----- */
3908:      0x00, /* ----- */
3909:
3910:          /* 0xd9 */
3911:      0x60, /* -##----- */
3912:      0x30, /* --##----- */
3913:      0x00, /* ----- */
3914:      0xc6, /* ##---##- */
3915:      0xc6, /* ##---##- */
3916:      0xc6, /* ##---##- */
3917:      0xc6, /* ##---##- */
3918:      0xc6, /* ##---##- */
3919:      0xc6, /* ##---##- */
3920:      0xc6, /* ##---##- */
3921:      0xc6, /* ##---##- */
3922:      0x7c, /* -#####- */
3923:      0x00, /* ----- */
3924:      0x00, /* ----- */
3925:      0x00, /* ----- */
3926:      0x00, /* ----- */
3927:
3928:          /* 0xda */
3929:      0x0c, /* -----##-- */
3930:      0x18, /* ---##----- */
3931:      0x00, /* ----- */
3932:      0xc6, /* ##---##- */
3933:      0xc6, /* ##---##- */
3934:      0xc6, /* ##---##- */
3935:      0xc6, /* ##---##- */
3936:      0xc6, /* ##---##- */
3937:      0xc6, /* ##---##- */
3938:      0xc6, /* ##---##- */
3939:      0xc6, /* ##---##- */
3940:      0x7c, /* -#####- */
3941:      0x00, /* ----- */
3942:      0x00, /* ----- */
3943:      0x00, /* ----- */
3944:      0x00, /* ----- */
3945:
3946:          /* 0xdb */
3947:      0x10, /* ----#----- */
3948:      0x38, /* --###----- */
3949:      0x6c, /* -##-##----- */
3950:      0x00, /* ----- */
3951:      0xc6, /* ##---##- */
3952:      0xc6, /* ##---##- */
3953:      0xc6, /* ##---##- */

```

drivers/video/font-lat9-8x16.c

Page 60/70

```

3954:      0xc6, /* ##---##- */
3955:      0xc6, /* ##---##- */
3956:      0xc6, /* ##---##- */
3957:      0xc6, /* ##---##- */
3958:      0x7c, /* -#####- */
3959:      0x00, /* ----- */
3960:      0x00, /* ----- */
3961:      0x00, /* ----- */
3962:      0x00, /* ----- */
3963:
3964:          /* 0xdc */
3965:      0x00, /* ----- */
3966:      0x6c, /* -##-##- */
3967:      0x00, /* ----- */
3968:      0xc6, /* ##---##- */
3969:      0xc6, /* ##---##- */
3970:      0xc6, /* ##---##- */
3971:      0xc6, /* ##---##- */
3972:      0xc6, /* ##---##- */
3973:      0xc6, /* ##---##- */
3974:      0xc6, /* ##---##- */
3975:      0xc6, /* ##---##- */
3976:      0x7c, /* -#####- */
3977:      0x00, /* ----- */
3978:      0x00, /* ----- */
3979:      0x00, /* ----- */
3980:      0x00, /* ----- */
3981:
3982:          /* 0xdd */
3983:      0x06, /* -----##- */
3984:      0x0c, /* -----##- */
3985:      0x00, /* ----- */
3986:      0x66, /* -##-##- */
3987:      0x66, /* -##-##- */
3988:      0x66, /* -##-##- */
3989:      0x66, /* -##-##- */
3990:      0x3c, /* --#####- */
3991:      0x18, /* ---##--- */
3992:      0x18, /* ---##--- */
3993:      0x18, /* ---##--- */
3994:      0x3c, /* --#####- */
3995:      0x00, /* ----- */
3996:      0x00, /* ----- */
3997:      0x00, /* ----- */
3998:      0x00, /* ----- */
3999:
4000:          /* 0xde */
4001:      0x00, /* ----- */
4002:      0x00, /* ----- */
4003:      0xf0, /* #####---- */
4004:      0x60, /* -##----- */
4005:      0x7c, /* -#####- */
4006:      0x66, /* -##-##- */
4007:      0x66, /* -##-##- */
4008:      0x66, /* -##-##- */
4009:      0x66, /* -##-##- */
4010:      0x7c, /* -#####- */
4011:      0x60, /* -##----- */
4012:      0xf0, /* #####---- */
4013:      0x00, /* ----- */
4014:      0x00, /* ----- */
4015:      0x00, /* ----- */
4016:      0x00, /* ----- */
4017:
4018:          /* 0xdf */
4019:      0x00, /* ----- */
4020:      0x00, /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 61/70

```

4021:      0x7c,    /* #####-- */
4022:      0xc6,    /* ##----##- */
4023:      0xc6,    /* ##----##- */
4024:      0xc6,    /* ##----##- */
4025:      0xcc,    /* ##--##-- */
4026:      0xc6,    /* ##----##- */
4027:      0xc6,    /* ##----##- */
4028:      0xc6,    /* ##----##- */
4029:      0xd6,    /* ##-##-##- */
4030:      0xdc,    /* ##-###-- */
4031:      0x80,    /* #----- */
4032:      0x00,    /* ----- */
4033:      0x00,    /* ----- */
4034:      0x00,    /* ----- */
4035:
4036:          /* 0xe0          */
4037:      0x00,    /* ----- */
4038:      0x60,    /* -##----- */
4039:      0x30,    /* --##----- */
4040:      0x18,    /* ----##---- */
4041:      0x00,    /* ----- */
4042:      0x78,    /* -####---- */
4043:      0x0c,    /* ----##-- */
4044:      0x7c,    /* -#####-- */
4045:      0xcc,    /* ##--##-- */
4046:      0xcc,    /* ##--##-- */
4047:      0xcc,    /* ##--##-- */
4048:      0x76,    /* -###-##- */
4049:      0x00,    /* ----- */
4050:      0x00,    /* ----- */
4051:      0x00,    /* ----- */
4052:      0x00,    /* ----- */
4053:
4054:          /* 0xe1          */
4055:      0x00,    /* ----- */
4056:      0x18,    /* ----##---- */
4057:      0x30,    /* --##----- */
4058:      0x60,    /* -##----- */
4059:      0x00,    /* ----- */
4060:      0x78,    /* -####---- */
4061:      0x0c,    /* ----##-- */
4062:      0x7c,    /* -#####-- */
4063:      0xcc,    /* ##--##-- */
4064:      0xcc,    /* ##--##-- */
4065:      0xcc,    /* ##--##-- */
4066:      0x76,    /* -###-##- */
4067:      0x00,    /* ----- */
4068:      0x00,    /* ----- */
4069:      0x00,    /* ----- */
4070:      0x00,    /* ----- */
4071:
4072:          /* 0xe2          */
4073:      0x00,    /* ----- */
4074:      0x10,    /* ----#----- */
4075:      0x38,    /* --###----- */
4076:      0x6c,    /* -##-##-- */
4077:      0x00,    /* ----- */
4078:      0x78,    /* -####---- */
4079:      0x0c,    /* ----##-- */
4080:      0x7c,    /* -#####-- */
4081:      0xcc,    /* ##--##-- */
4082:      0xcc,    /* ##--##-- */
4083:      0xcc,    /* ##--##-- */
4084:      0x76,    /* -###-##- */
4085:      0x00,    /* ----- */
4086:      0x00,    /* ----- */
4087:      0x00,    /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 62/70

```

4088:      0x00,    /* ----- */
4089:
4090:      /* 0xe3 */
4091:      0x00,    /* ----- */
4092:      0x00,    /* ----- */
4093:      0x76,    /* -###-##- */
4094:      0xdc,    /* ##-###-- */
4095:      0x00,    /* ----- */
4096:      0x78,    /* -####--- */
4097:      0x0c,    /* ----##-- */
4098:      0x7c,    /* -#####-- */
4099:      0xcc,    /* ##--##-- */
4100:      0xcc,    /* ##--##-- */
4101:      0xcc,    /* ##--##-- */
4102:      0x76,    /* -###-##- */
4103:      0x00,    /* ----- */
4104:      0x00,    /* ----- */
4105:      0x00,    /* ----- */
4106:      0x00,    /* ----- */
4107:
4108:      /* 0xe4 */
4109:      0x00,    /* ----- */
4110:      0x00,    /* ----- */
4111:      0x00,    /* ----- */
4112:      0x6c,    /* -##-##-- */
4113:      0x00,    /* ----- */
4114:      0x78,    /* -####--- */
4115:      0x0c,    /* ----##-- */
4116:      0x7c,    /* -#####-- */
4117:      0xcc,    /* ##--##-- */
4118:      0xcc,    /* ##--##-- */
4119:      0xcc,    /* ##--##-- */
4120:      0x76,    /* -###-##- */
4121:      0x00,    /* ----- */
4122:      0x00,    /* ----- */
4123:      0x00,    /* ----- */
4124:      0x00,    /* ----- */
4125:
4126:      /* 0xe5 */
4127:      0x00,    /* ----- */
4128:      0x38,    /* --###--- */
4129:      0x6c,    /* -##-##-- */
4130:      0x38,    /* --###--- */
4131:      0x00,    /* ----- */
4132:      0x78,    /* -####--- */
4133:      0x0c,    /* ----##-- */
4134:      0x7c,    /* -#####-- */
4135:      0xcc,    /* ##--##-- */
4136:      0xcc,    /* ##--##-- */
4137:      0xcc,    /* ##--##-- */
4138:      0x76,    /* -###-##- */
4139:      0x00,    /* ----- */
4140:      0x00,    /* ----- */
4141:      0x00,    /* ----- */
4142:      0x00,    /* ----- */
4143:
4144:      /* 0xe6 */
4145:      0x00,    /* ----- */
4146:      0x00,    /* ----- */
4147:      0x00,    /* ----- */
4148:      0x00,    /* ----- */
4149:      0x00,    /* ----- */
4150:      0x7e,    /* -#####- */
4151:      0xdb,    /* ##-##-## */
4152:      0x1b,    /* ---##-## */
4153:      0x7f,    /* -##### */
4154:      0xd8,    /* ##-##-- */

```

drivers/video/font-lat9-8x16.c

Page 63/70

```

4155:      0xdb,    /* ##-##-## */
4156:      0x7e,    /* -#####- */
4157:      0x00,    /* ----- */
4158:      0x00,    /* ----- */
4159:      0x00,    /* ----- */
4160:      0x00,    /* ----- */
4161:
4162:          /* 0xe7 */
4163:      0x00,    /* ----- */
4164:      0x00,    /* ----- */
4165:      0x00,    /* ----- */
4166:      0x00,    /* ----- */
4167:      0x00,    /* ----- */
4168:      0x7c,    /* -#####- */
4169:      0xc6,    /* ##---##- */
4170:      0xc0,    /* ##----- */
4171:      0xc0,    /* ##----- */
4172:      0xc0,    /* ##----- */
4173:      0xc6,    /* ##---##- */
4174:      0x7c,    /* -#####- */
4175:      0x18,    /* ---##--- */
4176:      0x6c,    /* -##-##- */
4177:      0x38,    /* --###--- */
4178:      0x00,    /* ----- */
4179:
4180:          /* 0xe8 */
4181:      0x00,    /* ----- */
4182:      0x60,    /* -##----- */
4183:      0x30,    /* --##----- */
4184:      0x18,    /* ---##--- */
4185:      0x00,    /* ----- */
4186:      0x7c,    /* -#####- */
4187:      0xc6,    /* ##---##- */
4188:      0xfe,    /* #####- */
4189:      0xc0,    /* ##----- */
4190:      0xc0,    /* ##----- */
4191:      0xc6,    /* ##---##- */
4192:      0x7c,    /* -#####- */
4193:      0x00,    /* ----- */
4194:      0x00,    /* ----- */
4195:      0x00,    /* ----- */
4196:      0x00,    /* ----- */
4197:
4198:          /* 0xe9 */
4199:      0x00,    /* ----- */
4200:      0x0c,    /* ----##- */
4201:      0x18,    /* ---##--- */
4202:      0x30,    /* --##----- */
4203:      0x00,    /* ----- */
4204:      0x7c,    /* -#####- */
4205:      0xc6,    /* ##---##- */
4206:      0xfe,    /* #####- */
4207:      0xc0,    /* ##----- */
4208:      0xc0,    /* ##----- */
4209:      0xc6,    /* ##---##- */
4210:      0x7c,    /* -#####- */
4211:      0x00,    /* ----- */
4212:      0x00,    /* ----- */
4213:      0x00,    /* ----- */
4214:      0x00,    /* ----- */
4215:
4216:          /* 0xea */
4217:      0x00,    /* ----- */
4218:      0x10,    /* ----#----- */
4219:      0x38,    /* --###--- */
4220:      0x6c,    /* -##-##- */
4221:      0x00,    /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 64/70

```

4222:      0x7c,    /* #####-- */
4223:      0xc6,    /* ##----##- */
4224:      0xfe,    /* #####-#- */
4225:      0xc0,    /* ##----- */
4226:      0xc0,    /* ##----- */
4227:      0xc6,    /* ##----##- */
4228:      0x7c,    /* #####-- */
4229:      0x00,    /* ----- */
4230:      0x00,    /* ----- */
4231:      0x00,    /* ----- */
4232:      0x00,    /* ----- */
4233:
4234:          /* 0xeb */
4235:      0x00,    /* ----- */
4236:      0x00,    /* ----- */
4237:      0x00,    /* ----- */
4238:      0x6c,    /* -##-##- */
4239:      0x00,    /* ----- */
4240:      0x7c,    /* #####-- */
4241:      0xc6,    /* ##----##- */
4242:      0xfe,    /* #####-#- */
4243:      0xc0,    /* ##----- */
4244:      0xc0,    /* ##----- */
4245:      0xc6,    /* ##----##- */
4246:      0x7c,    /* #####-- */
4247:      0x00,    /* ----- */
4248:      0x00,    /* ----- */
4249:      0x00,    /* ----- */
4250:      0x00,    /* ----- */
4251:
4252:          /* 0xec */
4253:      0x00,    /* ----- */
4254:      0x60,    /* -##----- */
4255:      0x30,    /* --##----- */
4256:      0x18,    /* ---##----- */
4257:      0x00,    /* ----- */
4258:      0x38,    /* ---###----- */
4259:      0x18,    /* ---##----- */
4260:      0x18,    /* ---##----- */
4261:      0x18,    /* ---##----- */
4262:      0x18,    /* ---##----- */
4263:      0x18,    /* ---##----- */
4264:      0x3c,    /* ---####----- */
4265:      0x00,    /* ----- */
4266:      0x00,    /* ----- */
4267:      0x00,    /* ----- */
4268:      0x00,    /* ----- */
4269:
4270:          /* 0xed */
4271:      0x00,    /* ----- */
4272:      0x0c,    /* ----##----- */
4273:      0x18,    /* ---##----- */
4274:      0x30,    /* --##----- */
4275:      0x00,    /* ----- */
4276:      0x38,    /* ---###----- */
4277:      0x18,    /* ---##----- */
4278:      0x18,    /* ---##----- */
4279:      0x18,    /* ---##----- */
4280:      0x18,    /* ---##----- */
4281:      0x18,    /* ---##----- */
4282:      0x3c,    /* ---####----- */
4283:      0x00,    /* ----- */
4284:      0x00,    /* ----- */
4285:      0x00,    /* ----- */
4286:      0x00,    /* ----- */
4287:
4288:          /* 0xee */

```

drivers/video/font-lat9-8x16.c

Page 65/70

```

4289:      0x00,    /* ----- */
4290:      0x18,    /* ---##--- */
4291:      0x3c,    /* --####-- */
4292:      0x66,    /* -##--##- */
4293:      0x00,    /* ----- */
4294:      0x38,    /* ---##--- */
4295:      0x18,    /* ---##--- */
4296:      0x18,    /* ---##--- */
4297:      0x18,    /* ---##--- */
4298:      0x18,    /* ---##--- */
4299:      0x18,    /* ---##--- */
4300:      0x3c,    /* --####-- */
4301:      0x00,    /* ----- */
4302:      0x00,    /* ----- */
4303:      0x00,    /* ----- */
4304:      0x00,    /* ----- */
4305:
4306:          /* 0xef      */
4307:      0x00,    /* ----- */
4308:      0x00,    /* ----- */
4309:      0x00,    /* ----- */
4310:      0x6c,    /* -##-##- */
4311:      0x00,    /* ----- */
4312:      0x38,    /* ---##--- */
4313:      0x18,    /* ---##--- */
4314:      0x18,    /* ---##--- */
4315:      0x18,    /* ---##--- */
4316:      0x18,    /* ---##--- */
4317:      0x18,    /* ---##--- */
4318:      0x3c,    /* --####-- */
4319:      0x00,    /* ----- */
4320:      0x00,    /* ----- */
4321:      0x00,    /* ----- */
4322:      0x00,    /* ----- */
4323:
4324:          /* 0xf0      */
4325:      0x00,    /* ----- */
4326:      0x78,    /* -####--- */
4327:      0x30,    /* --##---- */
4328:      0x78,    /* -####--- */
4329:      0x0c,    /* ----##-- */
4330:      0x7e,    /* -#####- */
4331:      0xc6,    /* ##----##- */
4332:      0xc6,    /* ##----##- */
4333:      0xc6,    /* ##----##- */
4334:      0xc6,    /* ##----##- */
4335:      0xc6,    /* ##----##- */
4336:      0x7c,    /* -#####- */
4337:      0x00,    /* ----- */
4338:      0x00,    /* ----- */
4339:      0x00,    /* ----- */
4340:      0x00,    /* ----- */
4341:
4342:          /* 0xf1      */
4343:      0x00,    /* ----- */
4344:      0x00,    /* ----- */
4345:      0x76,    /* -###-##- */
4346:      0xdc,    /* ##-###-- */
4347:      0x00,    /* ----- */
4348:      0xdc,    /* ##-###-- */
4349:      0x66,    /* -##--##- */
4350:      0x66,    /* -##--##- */
4351:      0x66,    /* -##--##- */
4352:      0x66,    /* -##--##- */
4353:      0x66,    /* -##--##- */
4354:      0x66,    /* -##--##- */
4355:      0x00,    /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 66/70

```

4356:      0x00, /* ----- */
4357:      0x00, /* ----- */
4358:      0x00, /* ----- */
4359:
4360:          /* 0xf2 */
4361:      0x00, /* ----- */
4362:      0x60, /* -##----- */
4363:      0x30, /* --##----- */
4364:      0x18, /* ---##---- */
4365:      0x00, /* ----- */
4366:      0x7c, /* -#####-- */
4367:      0xc6, /* ##---##- */
4368:      0xc6, /* ##---##- */
4369:      0xc6, /* ##---##- */
4370:      0xc6, /* ##---##- */
4371:      0xc6, /* ##---##- */
4372:      0x7c, /* -#####-- */
4373:      0x00, /* ----- */
4374:      0x00, /* ----- */
4375:      0x00, /* ----- */
4376:      0x00, /* ----- */
4377:
4378:          /* 0xf3 */
4379:      0x00, /* ----- */
4380:      0x0c, /* ----##-- */
4381:      0x18, /* ---##---- */
4382:      0x30, /* --##----- */
4383:      0x00, /* ----- */
4384:      0x7c, /* -#####-- */
4385:      0xc6, /* ##---##- */
4386:      0xc6, /* ##---##- */
4387:      0xc6, /* ##---##- */
4388:      0xc6, /* ##---##- */
4389:      0xc6, /* ##---##- */
4390:      0x7c, /* -#####-- */
4391:      0x00, /* ----- */
4392:      0x00, /* ----- */
4393:      0x00, /* ----- */
4394:      0x00, /* ----- */
4395:
4396:          /* 0xf4 */
4397:      0x00, /* ----- */
4398:      0x10, /* ---#----- */
4399:      0x38, /* --###----- */
4400:      0x6c, /* -##-##-- */
4401:      0x00, /* ----- */
4402:      0x7c, /* -#####-- */
4403:      0xc6, /* ##---##- */
4404:      0xc6, /* ##---##- */
4405:      0xc6, /* ##---##- */
4406:      0xc6, /* ##---##- */
4407:      0xc6, /* ##---##- */
4408:      0x7c, /* -#####-- */
4409:      0x00, /* ----- */
4410:      0x00, /* ----- */
4411:      0x00, /* ----- */
4412:      0x00, /* ----- */
4413:
4414:          /* 0xf5 */
4415:      0x00, /* ----- */
4416:      0x00, /* ----- */
4417:      0x76, /* -###-##- */
4418:      0xdc, /* ##-###-- */
4419:      0x00, /* ----- */
4420:      0x7c, /* -#####-- */
4421:      0xc6, /* ##---##- */
4422:      0xc6, /* ##---##- */

```


drivers/video/font-lat9-8x16.c

Page 67/70

```

4423:      0xc6, /* ##---##- */
4424:      0xc6, /* ##---##- */
4425:      0xc6, /* ##---##- */
4426:      0x7c, /* -#####- */
4427:      0x00, /* ----- */
4428:      0x00, /* ----- */
4429:      0x00, /* ----- */
4430:      0x00, /* ----- */
4431:
4432:          /* 0xf6 */
4433:      0x00, /* ----- */
4434:      0x00, /* ----- */
4435:      0x00, /* ----- */
4436:      0x6c, /* -##-##- */
4437:      0x00, /* ----- */
4438:      0x7c, /* -#####- */
4439:      0xc6, /* ##---##- */
4440:      0xc6, /* ##---##- */
4441:      0xc6, /* ##---##- */
4442:      0xc6, /* ##---##- */
4443:      0xc6, /* ##---##- */
4444:      0x7c, /* -#####- */
4445:      0x00, /* ----- */
4446:      0x00, /* ----- */
4447:      0x00, /* ----- */
4448:      0x00, /* ----- */
4449:
4450:          /* 0xf7 */
4451:      0x00, /* ----- */
4452:      0x00, /* ----- */
4453:      0x00, /* ----- */
4454:      0x00, /* ----- */
4455:      0x00, /* ----- */
4456:      0x18, /* ---##--- */
4457:      0x00, /* ----- */
4458:      0x7e, /* -#####- */
4459:      0x00, /* ----- */
4460:      0x18, /* ---##--- */
4461:      0x00, /* ----- */
4462:      0x00, /* ----- */
4463:      0x00, /* ----- */
4464:      0x00, /* ----- */
4465:      0x00, /* ----- */
4466:      0x00, /* ----- */
4467:
4468:          /* 0xf8 */
4469:      0x00, /* ----- */
4470:      0x00, /* ----- */
4471:      0x00, /* ----- */
4472:      0x00, /* ----- */
4473:      0x00, /* ----- */
4474:      0x7e, /* -#####- */
4475:      0xce, /* ##---##- */
4476:      0xde, /* ##-####- */
4477:      0xfe, /* #####- */
4478:      0xf6, /* #####-##- */
4479:      0xe6, /* ###---##- */
4480:      0xfc, /* #####- */
4481:      0x00, /* ----- */
4482:      0x00, /* ----- */
4483:      0x00, /* ----- */
4484:      0x00, /* ----- */
4485:
4486:          /* 0xf9 */
4487:      0x00, /* ----- */
4488:      0x60, /* -##----- */
4489:      0x30, /* --##----- */

```

drivers/video/font-lat9-8x16.c

Page 68/70

```

4490:      0x18, /* ----##---- */
4491:      0x00, /* ----- */
4492:      0xcc, /* ##--##-- */
4493:      0xcc, /* ##--##-- */
4494:      0xcc, /* ##--##-- */
4495:      0xcc, /* ##--##-- */
4496:      0xcc, /* ##--##-- */
4497:      0xcc, /* ##--##-- */
4498:      0x76, /* -###-##- */
4499:      0x00, /* ----- */
4500:      0x00, /* ----- */
4501:      0x00, /* ----- */
4502:      0x00, /* ----- */
4503:
4504:          /* 0xfa */
4505:      0x00, /* ----- */
4506:      0x18, /* ----##---- */
4507:      0x30, /* --##----- */
4508:      0x60, /* -##----- */
4509:      0x00, /* ----- */
4510:      0xcc, /* ##--##-- */
4511:      0xcc, /* ##--##-- */
4512:      0xcc, /* ##--##-- */
4513:      0xcc, /* ##--##-- */
4514:      0xcc, /* ##--##-- */
4515:      0xcc, /* ##--##-- */
4516:      0x76, /* -###-##- */
4517:      0x00, /* ----- */
4518:      0x00, /* ----- */
4519:      0x00, /* ----- */
4520:      0x00, /* ----- */
4521:
4522:          /* 0xfb */
4523:      0x00, /* ----- */
4524:      0x30, /* --##----- */
4525:      0x78, /* -###-##- */
4526:      0xcc, /* ##--##-- */
4527:      0x00, /* ----- */
4528:      0xcc, /* ##--##-- */
4529:      0xcc, /* ##--##-- */
4530:      0xcc, /* ##--##-- */
4531:      0xcc, /* ##--##-- */
4532:      0xcc, /* ##--##-- */
4533:      0xcc, /* ##--##-- */
4534:      0x76, /* -###-##- */
4535:      0x00, /* ----- */
4536:      0x00, /* ----- */
4537:      0x00, /* ----- */
4538:      0x00, /* ----- */
4539:
4540:          /* 0xfc */
4541:      0x00, /* ----- */
4542:      0x00, /* ----- */
4543:      0x00, /* ----- */
4544:      0xcc, /* ##--##-- */
4545:      0x00, /* ----- */
4546:      0xcc, /* ##--##-- */
4547:      0xcc, /* ##--##-- */
4548:      0xcc, /* ##--##-- */
4549:      0xcc, /* ##--##-- */
4550:      0xcc, /* ##--##-- */
4551:      0xcc, /* ##--##-- */
4552:      0x76, /* -###-##- */
4553:      0x00, /* ----- */
4554:      0x00, /* ----- */
4555:      0x00, /* ----- */
4556:      0x00, /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 69/70

```

4557:
4558:          /* 0xfd          */
4559:          0x00, /* ----- */
4560:          0x0c, /* ----##-- */
4561:          0x18, /* ---##--- */
4562:          0x30, /* --##---- */
4563:          0x00, /* ----- */
4564:          0xc6, /* ##---##- */
4565:          0xc6, /* ##---##- */
4566:          0xc6, /* ##---##- */
4567:          0xc6, /* ##---##- */
4568:          0xc6, /* ##---##- */
4569:          0xc6, /* ##---##- */
4570:          0x7e, /* -#####- */
4571:          0x06, /* -----##- */
4572:          0x0c, /* ----##-- */
4573:          0xf8, /* #####--- */
4574:          0x00, /* ----- */
4575:
4576:          /* 0xfe          */
4577:          0x00, /* ----- */
4578:          0x00, /* ----- */
4579:          0xf0, /* ####---- */
4580:          0x60, /* -##----- */
4581:          0x60, /* -##----- */
4582:          0x7c, /* -#####-- */
4583:          0x66, /* -##---##- */
4584:          0x66, /* -##---##- */
4585:          0x66, /* -##---##- */
4586:          0x66, /* -##---##- */
4587:          0x7c, /* -#####-- */
4588:          0x60, /* -##----- */
4589:          0x60, /* -##----- */
4590:          0xf0, /* ####---- */
4591:          0x00, /* ----- */
4592:          0x00, /* ----- */
4593:
4594:          /* 0xff          */
4595:          0x00, /* ----- */
4596:          0x00, /* ----- */
4597:          0x00, /* ----- */
4598:          0x6c, /* -##-##-- */
4599:          0x00, /* ----- */
4600:          0xc6, /* ##---##- */
4601:          0xc6, /* ##---##- */
4602:          0xc6, /* ##---##- */
4603:          0xc6, /* ##---##- */
4604:          0xc6, /* ##---##- */
4605:          0xc6, /* ##---##- */
4606:          0x7e, /* -#####- */
4607:          0x06, /* -----##- */
4608:          0x0c, /* ----##-- */
4609:          0xf8, /* #####--- */
4610:          0x00, /* ----- */
4611:
4612: };
4613:
4614: static unsigned char cursorshape_8x16[] = {
4615:          0x00, /* ----- */
4616:          0x00, /* ----- */
4617:          0x00, /* ----- */
4618:          0x00, /* ----- */
4619:          0x00, /* ----- */
4620:          0x00, /* ----- */
4621:          0x00, /* ----- */
4622:          0x00, /* ----- */
4623:          0x00, /* ----- */

```

drivers/video/font-lat9-8x16.c

Page 70/70

```
4624:         0x00,    /* ----- */
4625:         0x00,    /* ----- */
4626:         0x00,    /* ----- */
4627:         0x00,    /* ----- */
4628:         0x00,    /* ----- */
4629:         0xFF,    /* ##### */
4630:         0xFF,    /* ##### */
4631: };
4632:
4633: struct fbcon_font_desc font_lat9_8x16 = {
4634:     "VGA8x16",
4635:     8,
4636:     16,
4637:     fontdata_8x16,
4638:     cursorshape_8x16
4639: };
```

drivers/video/font-lat9-8x8.c

Page 1/39

```

1: #include <fiwix/font.h>
2:
3: static unsigned char fontdata_8x8[] = {
4:     /* 0x00 */
5:     0x7e, /* -#####- */
6:     0xc3, /* ##-----# */
7:     0x99, /* #--##--# */
8:     0xf3, /* #####--## */
9:     0xe7, /* ###--### */
10:    0xff, /* ######## */
11:    0xe7, /* ###--### */
12:    0x7e, /* -#####- */
13:
14:    /* 0x01 */
15:    0x00, /* ----- */
16:    0x76, /* -###-##- */
17:    0xdc, /* ##-##-#- */
18:    0x00, /* ----- */
19:    0x76, /* -###-##- */
20:    0xdc, /* ##-##-#- */
21:    0x00, /* ----- */
22:    0x00, /* ----- */
23:
24:    /* 0x02 */
25:    0x76, /* -###-##- */
26:    0xd8, /* ##-##-#- */
27:    0xd8, /* ##-##-#- */
28:    0xdc, /* ##-##-#- */
29:    0xd8, /* ##-##-#- */
30:    0xd8, /* ##-##-#- */
31:    0x76, /* -###-##- */
32:    0x00, /* ----- */
33:
34:    /* 0x03 */
35:    0x00, /* ----- */
36:    0x00, /* ----- */
37:    0x6e, /* -##-###- */
38:    0xd8, /* ##-##-#- */
39:    0xde, /* ##-####- */
40:    0xd8, /* ##-##-#- */
41:    0x6e, /* -##-###- */
42:    0x00, /* ----- */
43:
44:    /* 0x04 */
45:    0x10, /* ---#---- */
46:    0x38, /* --###--- */
47:    0x7c, /* -#####- */
48:    0xfe, /* ########- */
49:    0x7c, /* -#####- */
50:    0x38, /* --###--- */
51:    0x10, /* ---#---- */
52:    0x00, /* ----- */
53:
54:    /* 0x05 */
55:    0xa0, /* #-#----- */
56:    0xa0, /* #-#----- */
57:    0xe0, /* ###----- */
58:    0xae, /* #-#-###- */
59:    0xa4, /* #-#-##-#- */
60:    0x04, /* -----#-- */
61:    0x04, /* -----#-- */
62:    0x04, /* -----#-- */
63:
64:    /* 0x06 */
65:    0xe0, /* ###----- */
66:    0x80, /* #----- */
67:    0xc0, /* ##----- */

```

drivers/video/font-lat9-8x8.c

Page 2/39

```

68:      0x8e,    /* #---##- */
69:      0x88,    /* #---#--- */
70:      0x0c,    /* ----##-- */
71:      0x08,    /* ----#--- */
72:      0x08,    /* ----#--- */
73:
74:          /* 0x07 */
75:      0x60,    /* -##----- */
76:      0x80,    /* #----- */
77:      0x80,    /* #----- */
78:      0x8c,    /* #---##- */
79:      0x6a,    /* -##-##- */
80:      0x0c,    /* ----##-- */
81:      0x0a,    /* ----#-#- */
82:      0x0a,    /* ----#-#- */
83:
84:          /* 0x08 */
85:      0x80,    /* #----- */
86:      0x80,    /* #----- */
87:      0x80,    /* #----- */
88:      0x8e,    /* #---##- */
89:      0xe8,    /* ###-#--- */
90:      0x0c,    /* ----##-- */
91:      0x08,    /* ----#--- */
92:      0x08,    /* ----#--- */
93:
94:          /* 0x09 */
95:      0x22,    /* --#---#- */
96:      0x88,    /* #---#--- */
97:      0x22,    /* --#---#- */
98:      0x88,    /* #---#--- */
99:      0x22,    /* --#---#- */
100:     0x88,    /* #---#--- */
101:     0x22,    /* --#---#- */
102:     0x88,    /* #---#--- */
103:
104:          /* 0x0a */
105:     0x55,    /* -#-#-#-# */
106:     0xaa,    /* #-#-#-#- */
107:     0x55,    /* -#-#-#-# */
108:     0xaa,    /* #-#-#-#- */
109:     0x55,    /* -#-#-#-# */
110:     0xaa,    /* #-#-#-#- */
111:     0x55,    /* -#-#-#-# */
112:     0xaa,    /* #-#-#-#- */
113:
114:          /* 0x0b */
115:     0xee,    /* ###-###- */
116:     0xbb,    /* #-###-## */
117:     0xee,    /* ###-###- */
118:     0xbb,    /* #-###-## */
119:     0xee,    /* ###-###- */
120:     0xbb,    /* #-###-## */
121:     0xee,    /* ###-###- */
122:     0xbb,    /* #-###-## */
123:
124:          /* 0x0c */
125:     0xff,    /* ##### */
126:     0xff,    /* ##### */
127:     0xff,    /* ##### */
128:     0xff,    /* ##### */
129:     0xff,    /* ##### */
130:     0xff,    /* ##### */
131:     0xff,    /* ##### */
132:     0xff,    /* ##### */
133:
134:          /* 0x0d */

```

drivers/video/font-lat9-8x8.c

Page 3/39

```

135:      0x00,    /* ----- */
136:      0x00,    /* ----- */
137:      0x00,    /* ----- */
138:      0x00,    /* ----- */
139:      0xff,    /* ##### */
140:      0xff,    /* ##### */
141:      0xff,    /* ##### */
142:      0xff,    /* ##### */
143:
144:          /* 0x0e */
145:      0xff,    /* ##### */
146:      0xff,    /* ##### */
147:      0xff,    /* ##### */
148:      0xff,    /* ##### */
149:      0x00,    /* ----- */
150:      0x00,    /* ----- */
151:      0x00,    /* ----- */
152:      0x00,    /* ----- */
153:
154:          /* 0x0f */
155:      0xf0,    /* ###----- */
156:      0xf0,    /* ###----- */
157:      0xf0,    /* ###----- */
158:      0xf0,    /* ###----- */
159:      0xf0,    /* ###----- */
160:      0xf0,    /* ###----- */
161:      0xf0,    /* ###----- */
162:      0xf0,    /* ###----- */
163:
164:          /* 0x10 */
165:      0x0f,    /* ----#### */
166:      0x0f,    /* ----#### */
167:      0x0f,    /* ----#### */
168:      0x0f,    /* ----#### */
169:      0x0f,    /* ----#### */
170:      0x0f,    /* ----#### */
171:      0x0f,    /* ----#### */
172:      0x0f,    /* ----#### */
173:
174:          /* 0x11 */
175:      0x90,    /* #-#----- */
176:      0xd0,    /* #-#----- */
177:      0xf0,    /* ###----- */
178:      0xb4,    /* #-##-#-- */
179:      0x94,    /* #-#-#-#-- */
180:      0x04,    /* -----#-- */
181:      0x04,    /* -----#-- */
182:      0x07,    /* -----### */
183:
184:          /* 0x12 */
185:      0xa0,    /* #-#----- */
186:      0xa0,    /* #-#----- */
187:      0xa0,    /* #-#----- */
188:      0xae,    /* #-#-###- */
189:      0x44,    /* -#----#-- */
190:      0x04,    /* -----#-- */
191:      0x04,    /* -----#-- */
192:      0x04,    /* -----#-- */
193:
194:          /* 0x13 */
195:      0x18,    /* ----##---- */
196:      0x30,    /* --##----- */
197:      0x60,    /* -##----- */
198:      0x30,    /* --##----- */
199:      0x18,    /* ----##---- */
200:      0x00,    /* ----- */
201:      0xfc,    /* #####-- */

```

drivers/video/font-lat9-8x8.c

Page 4/39

```

202:      0x00, /* ----- */
203:
204:      /* 0x14 */
205:      0x60, /* -##----- */
206:      0x30, /* --##----- */
207:      0x18, /* ---##----- */
208:      0x30, /* --##----- */
209:      0x60, /* -##----- */
210:      0x00, /* ----- */
211:      0xfc, /* #####-- */
212:      0x00, /* ----- */
213:
214:      /* 0x15 */
215:      0x00, /* ----- */
216:      0x0c, /* ----##-- */
217:      0xfe, /* #####-- */
218:      0x18, /* ---##----- */
219:      0x30, /* --##----- */
220:      0xfe, /* #####-- */
221:      0x60, /* -##----- */
222:      0x00, /* ----- */
223:
224:      /* 0x16 */
225:      0x02, /* -----#- */
226:      0x0e, /* ----###- */
227:      0x3e, /* ---#####- */
228:      0xfe, /* #####-- */
229:      0x3e, /* ---#####- */
230:      0x0e, /* ----###- */
231:      0x02, /* -----#- */
232:      0x00, /* ----- */
233:
234:      /* 0x17 */
235:      0x80, /* #----- */
236:      0xe0, /* ###----- */
237:      0xf8, /* #####--- */
238:      0xfe, /* #####-- */
239:      0xf8, /* #####--- */
240:      0xe0, /* ###----- */
241:      0x80, /* #----- */
242:      0x00, /* ----- */
243:
244:      /* 0x18 */
245:      0x18, /* ---##----- */
246:      0x3c, /* --####-- */
247:      0x7e, /* -#####- */
248:      0x18, /* ---##----- */
249:      0x18, /* ---##----- */
250:      0x18, /* ---##----- */
251:      0x18, /* ---##----- */
252:      0x00, /* ----- */
253:
254:      /* 0x19 */
255:      0x18, /* ---##----- */
256:      0x18, /* ---##----- */
257:      0x18, /* ---##----- */
258:      0x18, /* ---##----- */
259:      0x7e, /* -#####- */
260:      0x3c, /* --####-- */
261:      0x18, /* ---##----- */
262:      0x00, /* ----- */
263:
264:      /* 0x1a */
265:      0x00, /* ----- */
266:      0x18, /* ---##----- */
267:      0x0c, /* ----##-- */
268:      0xfe, /* #####-- */

```


drivers/video/font-lat9-8x8.c

Page 5/39

```

269:      0x0c, /* ----##-- */
270:      0x18, /* ----##---- */
271:      0x00, /* ----- */
272:      0x00, /* ----- */
273:
274:          /* 0x1b */
275:      0x00, /* ----- */
276:      0x30, /* --##----- */
277:      0x60, /* -##----- */
278:      0xfe, /* #####- */
279:      0x60, /* -##----- */
280:      0x30, /* --##----- */
281:      0x00, /* ----- */
282:      0x00, /* ----- */
283:
284:          /* 0x1c */
285:      0x18, /* ----##---- */
286:      0x3c, /* --#####-- */
287:      0x7e, /* -#####- */
288:      0x18, /* ----##---- */
289:      0x18, /* ----##---- */
290:      0x7e, /* -#####- */
291:      0x3c, /* --#####-- */
292:      0x18, /* ----##---- */
293:
294:          /* 0x1d */
295:      0x00, /* ----- */
296:      0x24, /* --#--#-- */
297:      0x66, /* -##--##- */
298:      0xff, /* ##### */
299:      0x66, /* -##--##- */
300:      0x24, /* --#--#-- */
301:      0x00, /* ----- */
302:      0x00, /* ----- */
303:
304:          /* 0x1e */
305:      0x06, /* -----##- */
306:      0x06, /* -----##- */
307:      0x36, /* --##-##- */
308:      0x66, /* -##-##- */
309:      0xfe, /* #####- */
310:      0x60, /* -##----- */
311:      0x30, /* --##----- */
312:      0x00, /* ----- */
313:
314:          /* 0x1f (' ') */
315:      0x00, /* ----- */
316:      0xc0, /* #------ */
317:      0x7c, /* -#####-- */
318:      0x6e, /* -##-##-#- */
319:      0x6c, /* -##-##-#- */
320:      0x6c, /* -##-##-#- */
321:      0x6c, /* -##-##-#- */
322:      0x00, /* ----- */
323:
324:          /* 0x20 (' ') */
325:      0x00, /* ----- */
326:      0x00, /* ----- */
327:      0x00, /* ----- */
328:      0x00, /* ----- */
329:      0x00, /* ----- */
330:      0x00, /* ----- */
331:      0x00, /* ----- */
332:      0x00, /* ----- */
333:
334:          /* 0x21 ('!') */
335:      0x30, /* --##----- */

```

drivers/video/font-lat9-8x8.c

Page 6/39

```

336:      0x78,    /* ---- */
337:      0x78,    /* ---- */
338:      0x30,    /* ---#--- */
339:      0x30,    /* ---#--- */
340:      0x00,    /* ----- */
341:      0x30,    /* ---#--- */
342:      0x00,    /* ----- */
343:
344:          /* 0x22 ('"') */
345:      0x6c,    /* -##-##- */
346:      0x6c,    /* -##-##- */
347:      0x00,    /* ----- */
348:      0x00,    /* ----- */
349:      0x00,    /* ----- */
350:      0x00,    /* ----- */
351:      0x00,    /* ----- */
352:      0x00,    /* ----- */
353:
354:          /* 0x23 ('#') */
355:      0x6c,    /* -##-##- */
356:      0x6c,    /* -##-##- */
357:      0xfe,    /* #####- */
358:      0x6c,    /* -##-##- */
359:      0xfe,    /* #####- */
360:      0x6c,    /* -##-##- */
361:      0x6c,    /* -##-##- */
362:      0x00,    /* ----- */
363:
364:          /* 0x24 ('$') */
365:      0x30,    /* ---#--- */
366:      0x7c,    /* -#####- */
367:      0xc0,    /* ##----- */
368:      0x78,    /* -#####- */
369:      0x0c,    /* -----##- */
370:      0xf8,    /* #####- */
371:      0x30,    /* ---#--- */
372:      0x00,    /* ----- */
373:
374:          /* 0x25 ('%') */
375:      0x00,    /* ----- */
376:      0xc6,    /* ##---##- */
377:      0xcc,    /* ##---##- */
378:      0x18,    /* ----##--- */
379:      0x30,    /* ---#--- */
380:      0x66,    /* -##-##- */
381:      0xc6,    /* ##---##- */
382:      0x00,    /* ----- */
383:
384:          /* 0x26 ('&') */
385:      0x38,    /* ---###--- */
386:      0x6c,    /* -##-##- */
387:      0x38,    /* ---###--- */
388:      0x76,    /* -###-##- */
389:      0xdc,    /* ##-###- */
390:      0xcc,    /* ##---##- */
391:      0x76,    /* -###-##- */
392:      0x00,    /* ----- */
393:
394:          /* 0x27 ('''') */
395:      0x30,    /* ---#--- */
396:      0x30,    /* ---#--- */
397:      0x60,    /* -##----- */
398:      0x00,    /* ----- */
399:      0x00,    /* ----- */
400:      0x00,    /* ----- */
401:      0x00,    /* ----- */
402:      0x00,    /* ----- */

```

drivers/video/font-lat9-8x8.c

Page 7/39

```

403:
404:          /* 0x28 ('(') */
405:    0x18, /* ----##---- */
406:    0x30, /* --##----- */
407:    0x60, /* -##----- */
408:    0x60, /* -##----- */
409:    0x60, /* -##----- */
410:    0x30, /* --##----- */
411:    0x18, /* ----##---- */
412:    0x00, /* ----- */
413:
414:          /* 0x29 (')') */
415:    0x60, /* -##----- */
416:    0x30, /* --##----- */
417:    0x18, /* ----##---- */
418:    0x18, /* ----##---- */
419:    0x18, /* ----##---- */
420:    0x30, /* --##----- */
421:    0x60, /* -##----- */
422:    0x00, /* ----- */
423:
424:          /* 0x2a ('*') */
425:    0x00, /* ----- */
426:    0x66, /* -##--##- */
427:    0x3c, /* --####-- */
428:    0xf1, /* #####-- */
429:    0x3c, /* --####-- */
430:    0x66, /* -##--##- */
431:    0x00, /* ----- */
432:    0x00, /* ----- */
433:
434:          /* 0x2b ('+') */
435:    0x00, /* ----- */
436:    0x30, /* --##----- */
437:    0x30, /* --##----- */
438:    0xfc, /* #####-- */
439:    0x30, /* --##----- */
440:    0x30, /* --##----- */
441:    0x00, /* ----- */
442:    0x00, /* ----- */
443:
444:          /* 0x2c (',') */
445:    0x00, /* ----- */
446:    0x00, /* ----- */
447:    0x00, /* ----- */
448:    0x00, /* ----- */
449:    0x00, /* ----- */
450:    0x30, /* --##----- */
451:    0x30, /* --##----- */
452:    0x60, /* -##----- */
453:
454:          /* 0x2d ('-') */
455:    0x00, /* ----- */
456:    0x00, /* ----- */
457:    0x00, /* ----- */
458:    0xfc, /* #####-- */
459:    0x00, /* ----- */
460:    0x00, /* ----- */
461:    0x00, /* ----- */
462:    0x00, /* ----- */
463:
464:          /* 0x2e ('.') */
465:    0x00, /* ----- */
466:    0x00, /* ----- */
467:    0x00, /* ----- */
468:    0x00, /* ----- */
469:    0x00, /* ----- */

```

drivers/video/font-lat9-8x8.c

Page 8/39

```

470:      0x30, /* --##----- */
471:      0x30, /* --##----- */
472:      0x00, /* ----- */
473:
474:      /* 0x2f ('/' ) */
475:      0x00, /* ----- */
476:      0x06, /* -----##- */
477:      0x0c, /* -----##- */
478:      0x18, /* ---##---- */
479:      0x30, /* --##----- */
480:      0x60, /* -##----- */
481:      0xc0, /* ##----- */
482:      0x00, /* ----- */
483:
484:      /* 0x30 ('0' ) */
485:      0x7c, /* -#####- */
486:      0xc6, /* ##---##- */
487:      0xc6, /* ##---##- */
488:      0xd6, /* ##-##-##- */
489:      0xc6, /* ##---##- */
490:      0xc6, /* ##---##- */
491:      0x7c, /* -#####- */
492:      0x00, /* ----- */
493:
494:      /* 0x31 ('1' ) */
495:      0x30, /* --##----- */
496:      0x70, /* -###----- */
497:      0x30, /* --##----- */
498:      0x30, /* --##----- */
499:      0x30, /* --##----- */
500:      0x30, /* --##----- */
501:      0xfc, /* #####--- */
502:      0x00, /* ----- */
503:
504:      /* 0x32 ('2' ) */
505:      0x78, /* -#####- */
506:      0xcc, /* ##---##- */
507:      0x0c, /* -----##- */
508:      0x38, /* --###----- */
509:      0x60, /* -##----- */
510:      0xcc, /* ##---##- */
511:      0xfc, /* #####--- */
512:      0x00, /* ----- */
513:
514:      /* 0x33 ('3' ) */
515:      0x78, /* -#####- */
516:      0xcc, /* ##---##- */
517:      0x0c, /* -----##- */
518:      0x38, /* --###----- */
519:      0x0c, /* -----##- */
520:      0xcc, /* ##---##- */
521:      0x78, /* -#####- */
522:      0x00, /* ----- */
523:
524:      /* 0x34 ('4' ) */
525:      0x1c, /* ---###--- */
526:      0x3c, /* --####--- */
527:      0x6c, /* -##-##--- */
528:      0xcc, /* ##---##- */
529:      0xfe, /* #####--- */
530:      0x0c, /* -----##- */
531:      0x1e, /* ---###--- */
532:      0x00, /* ----- */
533:
534:      /* 0x35 ('5' ) */
535:      0xfc, /* #####--- */
536:      0xc0, /* ##----- */

```

drivers/video/font-lat9-8x8.c

Page 9/39

```

537:      0xf8, /* #####--- */
538:      0x0c, /* ----##-- */
539:      0x0c, /* ----##-- */
540:      0xcc, /* ##--##-- */
541:      0x78, /* -####--- */
542:      0x00, /* ----- */
543:
544:      /* 0x36 ('6') */
545:      0x38, /* --###--- */
546:      0x60, /* -##----- */
547:      0xc0, /* ##----- */
548:      0xf8, /* #####--- */
549:      0xcc, /* ##--##-- */
550:      0xcc, /* ##--##-- */
551:      0x78, /* -####--- */
552:      0x00, /* ----- */
553:
554:      /* 0x37 ('7') */
555:      0xfc, /* #####--- */
556:      0xcc, /* ##--##-- */
557:      0x0c, /* ----##-- */
558:      0x18, /* ---##--- */
559:      0x30, /* --##---- */
560:      0x30, /* --##---- */
561:      0x30, /* --##---- */
562:      0x00, /* ----- */
563:
564:      /* 0x38 ('8') */
565:      0x78, /* -####--- */
566:      0xcc, /* ##--##-- */
567:      0xcc, /* ##--##-- */
568:      0x78, /* -####--- */
569:      0xcc, /* ##--##-- */
570:      0xcc, /* ##--##-- */
571:      0x78, /* -####--- */
572:      0x00, /* ----- */
573:
574:      /* 0x39 ('9') */
575:      0x78, /* -####--- */
576:      0xcc, /* ##--##-- */
577:      0xcc, /* ##--##-- */
578:      0x7c, /* -#####--- */
579:      0x0c, /* ----##-- */
580:      0x18, /* ---##--- */
581:      0x70, /* -###----- */
582:      0x00, /* ----- */
583:
584:      /* 0x3a (':') */
585:      0x00, /* ----- */
586:      0x30, /* --##---- */
587:      0x30, /* --##---- */
588:      0x00, /* ----- */
589:      0x00, /* ----- */
590:      0x30, /* --##---- */
591:      0x30, /* --##---- */
592:      0x00, /* ----- */
593:
594:      /* 0x3b (';') */
595:      0x00, /* ----- */
596:      0x30, /* --##---- */
597:      0x30, /* --##---- */
598:      0x00, /* ----- */
599:      0x00, /* ----- */
600:      0x30, /* --##---- */
601:      0x30, /* --##---- */
602:      0x60, /* -##----- */
603:

```

drivers/video/font-lat9-8x8.c

Page 10/39

```

604:                /* 0x3c ('<') */
605:    0x18,          /* ----##---- */
606:    0x30,          /* --##----- */
607:    0x60,          /* -##----- */
608:    0xc0,          /* ##----- */
609:    0x60,          /* -##----- */
610:    0x30,          /* --##----- */
611:    0x18,          /* ----##---- */
612:    0x00,          /* ----- */
613:
614:                /* 0x3d ('=') */
615:    0x00,          /* ----- */
616:    0x00,          /* ----- */
617:    0xfc,          /* #####-- */
618:    0x00,          /* ----- */
619:    0x00,          /* ----- */
620:    0xfc,          /* #####-- */
621:    0x00,          /* ----- */
622:    0x00,          /* ----- */
623:
624:                /* 0x3e ('>') */
625:    0x60,          /* -##----- */
626:    0x30,          /* --##----- */
627:    0x18,          /* ---##---- */
628:    0x0c,          /* ----##-- */
629:    0x18,          /* ---##---- */
630:    0x30,          /* --##----- */
631:    0x60,          /* -##----- */
632:    0x00,          /* ----- */
633:
634:                /* 0x3f ('?') */
635:    0x78,          /* -####--- */
636:    0xcc,          /* ##--##-- */
637:    0x0c,          /* ----##-- */
638:    0x18,          /* ---##---- */
639:    0x30,          /* --##----- */
640:    0x00,          /* ----- */
641:    0x30,          /* --##----- */
642:    0x00,          /* ----- */
643:
644:                /* 0x40 ('@') */
645:    0x7c,          /* -####--- */
646:    0xc6,          /* ##---##- */
647:    0xde,          /* ##-####- */
648:    0xde,          /* ##-####- */
649:    0xde,          /* ##-####- */
650:    0xc0,          /* ##----- */
651:    0x78,          /* -####--- */
652:    0x00,          /* ----- */
653:
654:                /* 0x41 ('A') */
655:    0x30,          /* --##----- */
656:    0x78,          /* -####--- */
657:    0xcc,          /* ##--##-- */
658:    0xcc,          /* ##--##-- */
659:    0xfc,          /* #####-- */
660:    0xcc,          /* ##--##-- */
661:    0xcc,          /* ##--##-- */
662:    0x00,          /* ----- */
663:
664:                /* 0x42 ('B') */
665:    0xfc,          /* #####-- */
666:    0x66,          /* -##--##- */
667:    0x66,          /* -##--##- */
668:    0x7c,          /* -####--- */
669:    0x66,          /* -##--##- */
670:    0x66,          /* -##--##- */

```

drivers/video/font-lat9-8x8.c

Page 11/39

```

671:      0xfc, /* #####-- */
672:      0x00, /* ----- */
673:
674:      /* 0x43 ('C') */
675:      0x3c, /* ---####-- */
676:      0x66, /* -##--##- */
677:      0xc0, /* ##----- */
678:      0xc0, /* ##----- */
679:      0xc0, /* ##----- */
680:      0x66, /* -##--##- */
681:      0x3c, /* ---####-- */
682:      0x00, /* ----- */
683:
684:      /* 0x44 ('D') */
685:      0xf8, /* #####--- */
686:      0x6c, /* -##-##-- */
687:      0x66, /* -##--##- */
688:      0x66, /* -##--##- */
689:      0x66, /* -##--##- */
690:      0x6c, /* -##-##-- */
691:      0xf8, /* #####--- */
692:      0x00, /* ----- */
693:
694:      /* 0x45 ('E') */
695:      0xfe, /* #####-- */
696:      0x62, /* -##---#- */
697:      0x68, /* -##-#--- */
698:      0x78, /* -####--- */
699:      0x68, /* -##-#--- */
700:      0x62, /* -##---#- */
701:      0xfe, /* #####-- */
702:      0x00, /* ----- */
703:
704:      /* 0x46 ('F') */
705:      0xfe, /* #####-- */
706:      0x62, /* -##---#- */
707:      0x68, /* -##-#--- */
708:      0x78, /* -####--- */
709:      0x68, /* -##-#--- */
710:      0x60, /* -##----- */
711:      0xf0, /* ####----- */
712:      0x00, /* ----- */
713:
714:      /* 0x47 ('G') */
715:      0x3c, /* ---####-- */
716:      0x66, /* -##--##- */
717:      0xc0, /* ##----- */
718:      0xc0, /* ##----- */
719:      0xce, /* ##--####- */
720:      0x66, /* -##--##- */
721:      0x3e, /* --#####- */
722:      0x00, /* ----- */
723:
724:      /* 0x48 ('H') */
725:      0xcc, /* ##--##-- */
726:      0xcc, /* ##--##-- */
727:      0xcc, /* ##--##-- */
728:      0xfc, /* #####-- */
729:      0xcc, /* ##--##-- */
730:      0xcc, /* ##--##-- */
731:      0xcc, /* ##--##-- */
732:      0x00, /* ----- */
733:
734:      /* 0x49 ('I') */
735:      0x78, /* -####--- */
736:      0x30, /* --##----- */
737:      0x30, /* --##----- */

```

drivers/video/font-lat9-8x8.c

Page 12/39

```

738:      0x30,    /* --##----- */
739:      0x30,    /* --##----- */
740:      0x30,    /* --##----- */
741:      0x78,    /* -####----- */
742:      0x00,    /* ----- */
743:
744:          /* 0x4a ('J') */
745:      0x1e,    /* ----###- */
746:      0x0c,    /* ----##- */
747:      0x0c,    /* ----##- */
748:      0x0c,    /* ----##- */
749:      0xcc,    /* ##--##- */
750:      0xcc,    /* ##--##- */
751:      0x78,    /* -####----- */
752:      0x00,    /* ----- */
753:
754:          /* 0x4b ('K') */
755:      0xe6,    /* ###--##- */
756:      0x66,    /* -##--##- */
757:      0x6c,    /* -##--##- */
758:      0x78,    /* -####----- */
759:      0x6c,    /* -##--##- */
760:      0x66,    /* -##--##- */
761:      0xe6,    /* ###--##- */
762:      0x00,    /* ----- */
763:
764:          /* 0x4c ('L') */
765:      0xf0,    /* #####----- */
766:      0x60,    /* -##----- */
767:      0x60,    /* -##----- */
768:      0x60,    /* -##----- */
769:      0x62,    /* -##---#- */
770:      0x66,    /* -##--##- */
771:      0xfe,    /* #####---#- */
772:      0x00,    /* ----- */
773:
774:          /* 0x4d ('M') */
775:      0xc6,    /* ##---##- */
776:      0xee,    /* ###-###- */
777:      0xfe,    /* #####---#- */
778:      0xfe,    /* #####---#- */
779:      0xd6,    /* ##-##-##- */
780:      0xc6,    /* ##---##- */
781:      0xc6,    /* ##---##- */
782:      0x00,    /* ----- */
783:
784:          /* 0x4e ('N') */
785:      0xc6,    /* ##---##- */
786:      0xe6,    /* ###-##-#- */
787:      0xf6,    /* #####-##- */
788:      0xde,    /* ##-####-#- */
789:      0xce,    /* ##--###-#- */
790:      0xc6,    /* ##---##-#- */
791:      0xc6,    /* ##---##-#- */
792:      0x00,    /* ----- */
793:
794:          /* 0x4f ('O') */
795:      0x38,    /* --###----- */
796:      0x6c,    /* -##-##-#-#- */
797:      0xc6,    /* ##---##-#- */
798:      0xc6,    /* ##---##-#- */
799:      0xc6,    /* ##---##-#- */
800:      0x6c,    /* -##-##-#-#- */
801:      0x38,    /* --###----- */
802:      0x00,    /* ----- */
803:
804:          /* 0x50 ('P') */

```


drivers/video/font-lat9-8x8.c

Page 13/39

```

805:      0xfc, /* #####-- */
806:      0x66, /* -##---##- */
807:      0x66, /* -##---##- */
808:      0x7c, /* #####-- */
809:      0x60, /* -##----- */
810:      0x60, /* -##----- */
811:      0xf0, /* ###- ---- */
812:      0x00, /* ----- */
813:
814:      /* 0x51 ('Q') */
815:      0x78, /* -####---- */
816:      0xcc, /* ##--##-- */
817:      0xcc, /* ##--##-- */
818:      0xcc, /* ##--##-- */
819:      0xdc, /* ##-###-- */
820:      0x78, /* -####---- */
821:      0x1c, /* ----###-- */
822:      0x00, /* ----- */
823:
824:      /* 0x52 ('R') */
825:      0xfc, /* #####-- */
826:      0x66, /* -##---##- */
827:      0x66, /* -##---##- */
828:      0x7c, /* #####-- */
829:      0x6c, /* -##-##-- */
830:      0x66, /* -##---##- */
831:      0xe6, /* ###-##-- */
832:      0x00, /* ----- */
833:
834:      /* 0x53 ('S') */
835:      0x78, /* -####---- */
836:      0xcc, /* ##--##-- */
837:      0xe0, /* ###----- */
838:      0x70, /* -###----- */
839:      0x1c, /* ----###-- */
840:      0xcc, /* ##--##-- */
841:      0x78, /* -####---- */
842:      0x00, /* ----- */
843:
844:      /* 0x54 ('T') */
845:      0xfc, /* #####-- */
846:      0xb4, /* #-##-##-- */
847:      0x30, /* ---##----- */
848:      0x30, /* ---##----- */
849:      0x30, /* ---##----- */
850:      0x30, /* ---##----- */
851:      0x78, /* -####---- */
852:      0x00, /* ----- */
853:
854:      /* 0x55 ('U') */
855:      0xcc, /* ##--##-- */
856:      0xcc, /* ##--##-- */
857:      0xcc, /* ##--##-- */
858:      0xcc, /* ##--##-- */
859:      0xcc, /* ##--##-- */
860:      0xcc, /* ##--##-- */
861:      0x78, /* -####---- */
862:      0x00, /* ----- */
863:
864:      /* 0x56 ('V') */
865:      0xcc, /* ##--##-- */
866:      0xcc, /* ##--##-- */
867:      0xcc, /* ##--##-- */
868:      0xcc, /* ##--##-- */
869:      0xcc, /* ##--##-- */
870:      0x78, /* -####---- */
871:      0x30, /* ---##----- */

```

drivers/video/font-lat9-8x8.c

Page 14/39

```

872:      0x00,    /* ----- */
873:
874:      /* 0x57 ('W') */
875:      0xc6,    /* ##---##- */
876:      0xc6,    /* ##---##- */
877:      0xc6,    /* ##---##- */
878:      0xd6,    /* ##-##-##- */
879:      0xfe,    /* #####- */
880:      0xee,    /* ##-##-##- */
881:      0xc6,    /* ##---##- */
882:      0x00,    /* ----- */
883:
884:      /* 0x58 ('X') */
885:      0xc6,    /* ##---##- */
886:      0xc6,    /* ##---##- */
887:      0x6c,    /* -##-##- */
888:      0x38,    /* --###--- */
889:      0x6c,    /* -##-##- */
890:      0xc6,    /* ##---##- */
891:      0xc6,    /* ##---##- */
892:      0x00,    /* ----- */
893:
894:      /* 0x59 ('Y') */
895:      0xcc,    /* ##--##-- */
896:      0xcc,    /* ##--##-- */
897:      0xcc,    /* ##--##-- */
898:      0x78,    /* -####--- */
899:      0x30,    /* --##----- */
900:      0x30,    /* --##----- */
901:      0x78,    /* -####--- */
902:      0x00,    /* ----- */
903:
904:      /* 0x5a ('Z') */
905:      0xfe,    /* #####- */
906:      0xc6,    /* ##---##- */
907:      0x0c,    /* ----##-- */
908:      0x18,    /* ----##-- */
909:      0x30,    /* --##----- */
910:      0x66,    /* -##-##- */
911:      0xfe,    /* #####- */
912:      0x00,    /* ----- */
913:
914:      /* 0x5b ('[') */
915:      0x78,    /* -####--- */
916:      0x60,    /* -##----- */
917:      0x60,    /* -##----- */
918:      0x60,    /* -##----- */
919:      0x60,    /* -##----- */
920:      0x60,    /* -##----- */
921:      0x78,    /* -####--- */
922:      0x00,    /* ----- */
923:
924:      /* 0x5c ('\') */
925:      0x00,    /* ----- */
926:      0xc0,    /* ##----- */
927:      0x60,    /* -##----- */
928:      0x30,    /* --##----- */
929:      0x18,    /* ----##-- */
930:      0x0c,    /* ----##-- */
931:      0x06,    /* -----##- */
932:      0x00,    /* ----- */
933:
934:      /* 0x5d (']') */
935:      0x78,    /* -####--- */
936:      0x18,    /* ----##-- */
937:      0x18,    /* ----##-- */
938:      0x18,    /* ----##-- */

```

drivers/video/font-lat9-8x8.c

Page 15/39

```

939:      0x18, /* ----##---- */
940:      0x18, /* ----##---- */
941:      0x78, /* -####---- */
942:      0x00, /* ----- */
943:
944:      /* 0x5e ('^') */
945:      0x18, /* ----##---- */
946:      0x3c, /* --####-- */
947:      0x66, /* -##--##- */
948:      0x00, /* ----- */
949:      0x00, /* ----- */
950:      0x00, /* ----- */
951:      0x00, /* ----- */
952:      0x00, /* ----- */
953:
954:      /* 0x5f ('_') */
955:      0x00, /* ----- */
956:      0x00, /* ----- */
957:      0x00, /* ----- */
958:      0x00, /* ----- */
959:      0x00, /* ----- */
960:      0x00, /* ----- */
961:      0x00, /* ----- */
962:      0xff, /* ##### */
963:
964:      /* 0x60 ('`') */
965:      0x30, /* --##---- */
966:      0x30, /* --##---- */
967:      0x18, /* ----##---- */
968:      0x00, /* ----- */
969:      0x00, /* ----- */
970:      0x00, /* ----- */
971:      0x00, /* ----- */
972:      0x00, /* ----- */
973:
974:      /* 0x61 ('a') */
975:      0x00, /* ----- */
976:      0x00, /* ----- */
977:      0x78, /* -####---- */
978:      0x0c, /* ----##-- */
979:      0x7c, /* -#####-- */
980:      0xcc, /* ##--##-- */
981:      0x76, /* -##-##- */
982:      0x00, /* ----- */
983:
984:      /* 0x62 ('b') */
985:      0xe0, /* ###----- */
986:      0x60, /* -##----- */
987:      0x60, /* -##----- */
988:      0x7c, /* -#####-- */
989:      0x66, /* -##--##- */
990:      0x66, /* -##--##- */
991:      0xdc, /* ##-###-- */
992:      0x00, /* ----- */
993:
994:      /* 0x63 ('c') */
995:      0x00, /* ----- */
996:      0x00, /* ----- */
997:      0x78, /* -####---- */
998:      0xcc, /* ##--##-- */
999:      0xc0, /* ##----- */
1000:     0xcc, /* ##--##-- */
1001:     0x78, /* -####---- */
1002:     0x00, /* ----- */
1003:
1004:     /* 0x64 ('d') */
1005:     0x1c, /* ----##-- */

```

drivers/video/font-lat9-8x8.c

Page 16/39

```

1006:      0x0c,    /* ----##-- */
1007:      0x0c,    /* ----##-- */
1008:      0x7c,    /* -##### */
1009:      0xcc,    /* ##--##-- */
1010:      0xcc,    /* ##--##-- */
1011:      0x76,    /* -###-##- */
1012:      0x00,    /* ----- */
1013:
1014:          /* 0x65 ('e') */
1015:      0x00,    /* ----- */
1016:      0x00,    /* ----- */
1017:      0x78,    /* -####--- */
1018:      0xcc,    /* ##--##-- */
1019:      0xfc,    /* #####--- */
1020:      0xc0,    /* ##----- */
1021:      0x78,    /* -####--- */
1022:      0x00,    /* ----- */
1023:
1024:          /* 0x66 ('f') */
1025:      0x38,    /* --###--- */
1026:      0x6c,    /* -##-##-- */
1027:      0x60,    /* -##----- */
1028:      0xf0,    /* #####--- */
1029:      0x60,    /* -##----- */
1030:      0x60,    /* -##----- */
1031:      0xf0,    /* #####--- */
1032:      0x00,    /* ----- */
1033:
1034:          /* 0x67 ('g') */
1035:      0x00,    /* ----- */
1036:      0x00,    /* ----- */
1037:      0x76,    /* -###-##- */
1038:      0xcc,    /* ##--##-- */
1039:      0xcc,    /* ##--##-- */
1040:      0x7c,    /* -##### */
1041:      0x0c,    /* ----##-- */
1042:      0xf8,    /* #####--- */
1043:
1044:          /* 0x68 ('h') */
1045:      0xe0,    /* ###----- */
1046:      0x60,    /* -##----- */
1047:      0x6c,    /* -##-##-- */
1048:      0x76,    /* -###-##- */
1049:      0x66,    /* -##--##- */
1050:      0x66,    /* -##--##- */
1051:      0xe6,    /* ###--##- */
1052:      0x00,    /* ----- */
1053:
1054:          /* 0x69 ('i') */
1055:      0x30,    /* --##----- */
1056:      0x00,    /* ----- */
1057:      0x70,    /* -###----- */
1058:      0x30,    /* --##----- */
1059:      0x30,    /* --##----- */
1060:      0x30,    /* --##----- */
1061:      0x78,    /* -##### */
1062:      0x00,    /* ----- */
1063:
1064:          /* 0x6a ('j') */
1065:      0x0c,    /* ----##-- */
1066:      0x00,    /* ----- */
1067:      0x0c,    /* ----##-- */
1068:      0x0c,    /* ----##-- */
1069:      0x0c,    /* ----##-- */
1070:      0xcc,    /* ##--##-- */
1071:      0xcc,    /* ##--##-- */
1072:      0x78,    /* -####--- */

```

drivers/video/font-lat9-8x8.c

Page 17/39

```

1073:
1074:          /* 0x6b ('k') */
1075:    0xe0,  /* ###----- */
1076:    0x60,  /* -##----- */
1077:    0x66,  /* -##--##- */
1078:    0x6c,  /* -##-##-- */
1079:    0x78,  /* -####---- */
1080:    0x6c,  /* -##-##-- */
1081:    0xe6,  /* ###--##- */
1082:    0x00,  /* ----- */
1083:
1084:          /* 0x6c ('l') */
1085:    0x70,  /* -###----- */
1086:    0x30,  /* --##----- */
1087:    0x30,  /* --##----- */
1088:    0x30,  /* --##----- */
1089:    0x30,  /* --##----- */
1090:    0x30,  /* --##----- */
1091:    0x78,  /* -####---- */
1092:    0x00,  /* ----- */
1093:
1094:          /* 0x6d ('m') */
1095:    0x00,  /* ----- */
1096:    0x00,  /* ----- */
1097:    0xcc,  /* ##--##-- */
1098:    0xfe,  /* #####-- */
1099:    0xfe,  /* #####-- */
1100:    0xd6,  /* ##-##-##- */
1101:    0xc6,  /* ##--##- */
1102:    0x00,  /* ----- */
1103:
1104:          /* 0x6e ('n') */
1105:    0x00,  /* ----- */
1106:    0x00,  /* ----- */
1107:    0xf8,  /* #####-- */
1108:    0xcc,  /* ##--##-- */
1109:    0xcc,  /* ##--##-- */
1110:    0xcc,  /* ##--##-- */
1111:    0xcc,  /* ##--##-- */
1112:    0x00,  /* ----- */
1113:
1114:          /* 0x6f ('o') */
1115:    0x00,  /* ----- */
1116:    0x00,  /* ----- */
1117:    0x78,  /* -####---- */
1118:    0xcc,  /* ##--##-- */
1119:    0xcc,  /* ##--##-- */
1120:    0xcc,  /* ##--##-- */
1121:    0x78,  /* -####---- */
1122:    0x00,  /* ----- */
1123:
1124:          /* 0x70 ('p') */
1125:    0x00,  /* ----- */
1126:    0x00,  /* ----- */
1127:    0xdc,  /* ##-##-##- */
1128:    0x66,  /* -##--##- */
1129:    0x66,  /* -##--##- */
1130:    0x7c,  /* -#####-- */
1131:    0x60,  /* -##----- */
1132:    0xf0,  /* #####----- */
1133:
1134:          /* 0x71 ('q') */
1135:    0x00,  /* ----- */
1136:    0x00,  /* ----- */
1137:    0x76,  /* -##-##- */
1138:    0xcc,  /* ##--##-- */
1139:    0xcc,  /* ##--##-- */

```

drivers/video/font-lat9-8x8.c

Page 18/39

```

1140:      0x7c, /* #####-- */
1141:      0x0c, /* -----#-- */
1142:      0x1e, /* ---##### */
1143:
1144:      /* 0x72 ('r') */
1145:      0x00, /* ----- */
1146:      0x00, /* ----- */
1147:      0xdc, /* ##-###-- */
1148:      0x76, /* -##-##- */
1149:      0x66, /* -##-##- */
1150:      0x60, /* -##----- */
1151:      0xf0, /* #####---- */
1152:      0x00, /* ----- */
1153:
1154:      /* 0x73 ('s') */
1155:      0x00, /* ----- */
1156:      0x00, /* ----- */
1157:      0x7c, /* #####-- */
1158:      0xc0, /* ##----- */
1159:      0x78, /* -###---- */
1160:      0x0c, /* -----#-- */
1161:      0xf8, /* #####---- */
1162:      0x00, /* ----- */
1163:
1164:      /* 0x74 ('t') */
1165:      0x10, /* ---#----- */
1166:      0x30, /* --##----- */
1167:      0x7c, /* #####-- */
1168:      0x30, /* --##----- */
1169:      0x30, /* --##----- */
1170:      0x34, /* --##-#-- */
1171:      0x18, /* ---#----- */
1172:      0x00, /* ----- */
1173:
1174:      /* 0x75 ('u') */
1175:      0x00, /* ----- */
1176:      0x00, /* ----- */
1177:      0xcc, /* ##--##-- */
1178:      0xcc, /* ##--##-- */
1179:      0xcc, /* ##--##-- */
1180:      0xcc, /* ##--##-- */
1181:      0x76, /* -##-##- */
1182:      0x00, /* ----- */
1183:
1184:      /* 0x76 ('v') */
1185:      0x00, /* ----- */
1186:      0x00, /* ----- */
1187:      0xcc, /* ##--##-- */
1188:      0xcc, /* ##--##-- */
1189:      0xcc, /* ##--##-- */
1190:      0x78, /* #####-- */
1191:      0x30, /* --##----- */
1192:      0x00, /* ----- */
1193:
1194:      /* 0x77 ('w') */
1195:      0x00, /* ----- */
1196:      0x00, /* ----- */
1197:      0xc6, /* ##---##- */
1198:      0xd6, /* ##-#-##- */
1199:      0xfe, /* #####-#- */
1200:      0xfe, /* #####-#- */
1201:      0x6c, /* -##-##-- */
1202:      0x00, /* ----- */
1203:
1204:      /* 0x78 ('x') */
1205:      0x00, /* ----- */
1206:      0x00, /* ----- */

```

drivers/video/font-lat9-8x8.c

Page 19/39

```

1207:      0xc6,    /* ##---##- */
1208:      0x6c,    /* -##-##-- */
1209:      0x38,    /* --###--- */
1210:      0x6c,    /* -##-##-- */
1211:      0xc6,    /* ##---##- */
1212:      0x00,    /* ----- */
1213:
1214:          /* 0x79 ('y') */
1215:      0x00,    /* ----- */
1216:      0x00,    /* ----- */
1217:      0xcc,    /* ##---##- */
1218:      0xcc,    /* ##---##- */
1219:      0xcc,    /* ##---##- */
1220:      0x7c,    /* -#####- */
1221:      0x0c,    /* ----##-- */
1222:      0xf8,    /* #####--- */
1223:
1224:          /* 0x7a ('z') */
1225:      0x00,    /* ----- */
1226:      0x00,    /* ----- */
1227:      0xfc,    /* #####--- */
1228:      0x98,    /* #-##--- */
1229:      0x30,    /* --##--- */
1230:      0x64,    /* -##-##-- */
1231:      0xfc,    /* #####--- */
1232:      0x00,    /* ----- */
1233:
1234:          /* 0x7b ('{') */
1235:      0x1c,    /* ---###-- */
1236:      0x30,    /* --##--- */
1237:      0x30,    /* --##--- */
1238:      0xe0,    /* ###----- */
1239:      0x30,    /* --##--- */
1240:      0x30,    /* --##--- */
1241:      0x1c,    /* ---###-- */
1242:      0x00,    /* ----- */
1243:
1244:          /* 0x7c ('|') */
1245:      0x30,    /* --##--- */
1246:      0x30,    /* --##--- */
1247:      0x30,    /* --##--- */
1248:      0x30,    /* --##--- */
1249:      0x30,    /* --##--- */
1250:      0x30,    /* --##--- */
1251:      0x30,    /* --##--- */
1252:      0x00,    /* ----- */
1253:
1254:          /* 0x7d ('}') */
1255:      0xe0,    /* ###----- */
1256:      0x30,    /* --##--- */
1257:      0x30,    /* --##--- */
1258:      0x1c,    /* ---###-- */
1259:      0x30,    /* --##--- */
1260:      0x30,    /* --##--- */
1261:      0xe0,    /* ###----- */
1262:      0x00,    /* ----- */
1263:
1264:          /* 0x7e ('~') */
1265:      0x76,    /* -###-##- */
1266:      0xdc,    /* #-###--- */
1267:      0x00,    /* ----- */
1268:      0x00,    /* ----- */
1269:      0x00,    /* ----- */
1270:      0x00,    /* ----- */
1271:      0x00,    /* ----- */
1272:      0x00,    /* ----- */
1273:

```

drivers/video/font-lat9-8x8.c

Page 20/39

```

1274:                /* 0x7f          */
1275:    0xcc,          /* ##---##--- */
1276:    0x00,          /* ----- */
1277:    0xcc,          /* ##---##--- */
1278:    0xcc,          /* ##---##--- */
1279:    0x78,          /* -####--- */
1280:    0x30,          /* --##----- */
1281:    0x78,          /* -####--- */
1282:    0x00,          /* ----- */
1283:
1284:                /* 0x80          */
1285:    0x00,          /* ----- */
1286:    0xff,          /* ##### */
1287:    0x00,          /* ----- */
1288:    0x00,          /* ----- */
1289:    0x00,          /* ----- */
1290:    0x00,          /* ----- */
1291:    0x00,          /* ----- */
1292:    0x00,          /* ----- */
1293:
1294:                /* 0x81          */
1295:    0x18,          /* ---##--- */
1296:    0x18,          /* ---##--- */
1297:    0x18,          /* ---##--- */
1298:    0x18,          /* ---##--- */
1299:    0x00,          /* ----- */
1300:    0x00,          /* ----- */
1301:    0x00,          /* ----- */
1302:    0x00,          /* ----- */
1303:
1304:                /* 0x82          */
1305:    0x00,          /* ----- */
1306:    0x00,          /* ----- */
1307:    0x00,          /* ----- */
1308:    0x1f,          /* ---##### */
1309:    0x00,          /* ----- */
1310:    0x00,          /* ----- */
1311:    0x00,          /* ----- */
1312:    0x00,          /* ----- */
1313:
1314:                /* 0x83          */
1315:    0x18,          /* ---##--- */
1316:    0x18,          /* ---##--- */
1317:    0x18,          /* ---##--- */
1318:    0x1f,          /* ---##### */
1319:    0x00,          /* ----- */
1320:    0x00,          /* ----- */
1321:    0x00,          /* ----- */
1322:    0x00,          /* ----- */
1323:
1324:                /* 0x84          */
1325:    0x00,          /* ----- */
1326:    0x00,          /* ----- */
1327:    0x00,          /* ----- */
1328:    0x00,          /* ----- */
1329:    0x18,          /* ---##--- */
1330:    0x18,          /* ---##--- */
1331:    0x18,          /* ---##--- */
1332:    0x18,          /* ---##--- */
1333:
1334:                /* 0x85          */
1335:    0x18,          /* ---##--- */
1336:    0x18,          /* ---##--- */
1337:    0x18,          /* ---##--- */
1338:    0x18,          /* ---##--- */
1339:    0x18,          /* ---##--- */
1340:    0x18,          /* ---##--- */

```


drivers/video/font-lat9-8x8.c

Page 21/39

```

1341:      0x18, /* ----##---- */
1342:      0x18, /* ----##---- */
1343:
1344:          /* 0x86 */
1345:      0x00, /* ----- */
1346:      0x00, /* ----- */
1347:      0x00, /* ----- */
1348:      0x1f, /* ----##### */
1349:      0x18, /* ----##---- */
1350:      0x18, /* ----##---- */
1351:      0x18, /* ----##---- */
1352:      0x18, /* ----##---- */
1353:
1354:          /* 0x87 */
1355:      0x18, /* ----##---- */
1356:      0x18, /* ----##---- */
1357:      0x18, /* ----##---- */
1358:      0x1f, /* ----##### */
1359:      0x18, /* ----##---- */
1360:      0x18, /* ----##---- */
1361:      0x18, /* ----##---- */
1362:      0x18, /* ----##---- */
1363:
1364:          /* 0x88 */
1365:      0x00, /* ----- */
1366:      0x00, /* ----- */
1367:      0x00, /* ----- */
1368:      0xf8, /* #####---- */
1369:      0x00, /* ----- */
1370:      0x00, /* ----- */
1371:      0x00, /* ----- */
1372:      0x00, /* ----- */
1373:
1374:          /* 0x89 */
1375:      0x18, /* ----##---- */
1376:      0x18, /* ----##---- */
1377:      0x18, /* ----##---- */
1378:      0xf8, /* #####---- */
1379:      0x00, /* ----- */
1380:      0x00, /* ----- */
1381:      0x00, /* ----- */
1382:      0x00, /* ----- */
1383:
1384:          /* 0x8a */
1385:      0x00, /* ----- */
1386:      0x00, /* ----- */
1387:      0x00, /* ----- */
1388:      0xff, /* #####---- */
1389:      0x00, /* ----- */
1390:      0x00, /* ----- */
1391:      0x00, /* ----- */
1392:      0x00, /* ----- */
1393:
1394:          /* 0x8b */
1395:      0x18, /* ----##---- */
1396:      0x18, /* ----##---- */
1397:      0x18, /* ----##---- */
1398:      0xff, /* #####---- */
1399:      0x00, /* ----- */
1400:      0x00, /* ----- */
1401:      0x00, /* ----- */
1402:      0x00, /* ----- */
1403:
1404:          /* 0x8c */
1405:      0x00, /* ----- */
1406:      0x00, /* ----- */
1407:      0x00, /* ----- */

```

drivers/video/font-lat9-8x8.c

Page 22/39

```

1408:      0xf8, /* #####--- */
1409:      0x18, /* ----##--- */
1410:      0x18, /* ----##--- */
1411:      0x18, /* ----##--- */
1412:      0x18, /* ----##--- */
1413:
1414:          /* 0x8d */
1415:      0x18, /* ----##--- */
1416:      0x18, /* ----##--- */
1417:      0x18, /* ----##--- */
1418:      0xf8, /* #####--- */
1419:      0x18, /* ----##--- */
1420:      0x18, /* ----##--- */
1421:      0x18, /* ----##--- */
1422:      0x18, /* ----##--- */
1423:
1424:          /* 0x8e */
1425:      0x00, /* ----- */
1426:      0x00, /* ----- */
1427:      0x00, /* ----- */
1428:      0xff, /* ##### */
1429:      0x18, /* ----##--- */
1430:      0x18, /* ----##--- */
1431:      0x18, /* ----##--- */
1432:      0x18, /* ----##--- */
1433:
1434:          /* 0x8f */
1435:      0x18, /* ----##--- */
1436:      0x18, /* ----##--- */
1437:      0x18, /* ----##--- */
1438:      0xff, /* ##### */
1439:      0x18, /* ----##--- */
1440:      0x18, /* ----##--- */
1441:      0x18, /* ----##--- */
1442:      0x18, /* ----##--- */
1443:
1444:          /* 0x90 */
1445:      0x00, /* ----- */
1446:      0x00, /* ----- */
1447:      0x00, /* ----- */
1448:      0x00, /* ----- */
1449:      0x00, /* ----- */
1450:      0xff, /* ##### */
1451:      0x00, /* ----- */
1452:      0x00, /* ----- */
1453:
1454:          /* 0x91 */
1455:      0x6c, /* -##-##- */
1456:      0x6c, /* -##-##- */
1457:      0x6c, /* -##-##- */
1458:      0x6c, /* -##-##- */
1459:      0x7c, /* -#####- */
1460:      0x00, /* ----- */
1461:      0x00, /* ----- */
1462:      0x00, /* ----- */
1463:
1464:          /* 0x92 */
1465:      0x00, /* ----- */
1466:      0x00, /* ----- */
1467:      0x7f, /* -##### */
1468:      0x60, /* -##----- */
1469:      0x7f, /* -##### */
1470:      0x00, /* ----- */
1471:      0x00, /* ----- */
1472:      0x00, /* ----- */
1473:
1474:          /* 0x93 */

```

drivers/video/font-lat9-8x8.c

Page 23/39

```

1475:      0x6c, /* -##-##-- */
1476:      0x6c, /* -##-##-- */
1477:      0x6f, /* -##-#### */
1478:      0x60, /* -##----- */
1479:      0x7f, /* -##### */
1480:      0x00, /* ----- */
1481:      0x00, /* ----- */
1482:      0x00, /* ----- */
1483:
1484:          /* 0x94 */
1485:      0x00, /* ----- */
1486:      0x00, /* ----- */
1487:      0x7c, /* -####-- */
1488:      0x6c, /* -##-##-- */
1489:      0x6c, /* -##-##-- */
1490:      0x6c, /* -##-##-- */
1491:      0x6c, /* -##-##-- */
1492:      0x6c, /* -##-##-- */
1493:
1494:          /* 0x95 */
1495:      0x6c, /* -##-##-- */
1496:      0x6c, /* -##-##-- */
1497:      0x6c, /* -##-##-- */
1498:      0x6c, /* -##-##-- */
1499:      0x6c, /* -##-##-- */
1500:      0x6c, /* -##-##-- */
1501:      0x6c, /* -##-##-- */
1502:      0x6c, /* -##-##-- */
1503:
1504:          /* 0x96 */
1505:      0x00, /* ----- */
1506:      0x00, /* ----- */
1507:      0x7f, /* -##### */
1508:      0x60, /* -##----- */
1509:      0x6f, /* -##-#### */
1510:      0x6c, /* -##-##-- */
1511:      0x6c, /* -##-##-- */
1512:      0x6c, /* -##-##-- */
1513:
1514:          /* 0x97 */
1515:      0x6c, /* -##-##-- */
1516:      0x6c, /* -##-##-- */
1517:      0x6f, /* -##-#### */
1518:      0x60, /* -##----- */
1519:      0x6f, /* -##-#### */
1520:      0x6c, /* -##-##-- */
1521:      0x6c, /* -##-##-- */
1522:      0x6c, /* -##-##-- */
1523:
1524:          /* 0x98 */
1525:      0x00, /* ----- */
1526:      0x00, /* ----- */
1527:      0xfc, /* #####-- */
1528:      0x0c, /* ----##-- */
1529:      0xfc, /* #####-- */
1530:      0x00, /* ----- */
1531:      0x00, /* ----- */
1532:      0x00, /* ----- */
1533:
1534:          /* 0x99 */
1535:      0x6c, /* -##-##-- */
1536:      0x6c, /* -##-##-- */
1537:      0xec, /* ###-##-- */
1538:      0x0c, /* ----##-- */
1539:      0xfc, /* #####-- */
1540:      0x00, /* ----- */
1541:      0x00, /* ----- */

```

drivers/video/font-lat9-8x8.c

Page 24/39

```

1542:      0x00,    /* ----- */
1543:
1544:      /* 0x9a */
1545:      0x00,    /* ----- */
1546:      0x00,    /* ----- */
1547:      0xff,    /* ##### */
1548:      0x00,    /* ----- */
1549:      0xff,    /* ##### */
1550:      0x00,    /* ----- */
1551:      0x00,    /* ----- */
1552:      0x00,    /* ----- */
1553:
1554:      /* 0x9b */
1555:      0x6c,    /* -##-##- */
1556:      0x6c,    /* -##-##- */
1557:      0xef,    /* ###-#### */
1558:      0x00,    /* ----- */
1559:      0xff,    /* ##### */
1560:      0x00,    /* ----- */
1561:      0x00,    /* ----- */
1562:      0x00,    /* ----- */
1563:
1564:      /* 0x9c */
1565:      0x00,    /* ----- */
1566:      0x00,    /* ----- */
1567:      0xfc,    /* #####- */
1568:      0x0c,    /* ----##- */
1569:      0xec,    /* ###-##- */
1570:      0x6c,    /* -##-##- */
1571:      0x6c,    /* -##-##- */
1572:      0x6c,    /* -##-##- */
1573:
1574:      /* 0x9d */
1575:      0x6c,    /* -##-##- */
1576:      0x6c,    /* -##-##- */
1577:      0xec,    /* ###-##- */
1578:      0x0c,    /* ----##- */
1579:      0xec,    /* ###-##- */
1580:      0x6c,    /* -##-##- */
1581:      0x6c,    /* -##-##- */
1582:      0x6c,    /* -##-##- */
1583:
1584:      /* 0x9e */
1585:      0x00,    /* ----- */
1586:      0x00,    /* ----- */
1587:      0xff,    /* ##### */
1588:      0x00,    /* ----- */
1589:      0xef,    /* ###-#### */
1590:      0x6c,    /* -##-##- */
1591:      0x6c,    /* -##-##- */
1592:      0x6c,    /* -##-##- */
1593:
1594:      /* 0x9f */
1595:      0x6c,    /* -##-##- */
1596:      0x6c,    /* -##-##- */
1597:      0xef,    /* ###-#### */
1598:      0x00,    /* ----- */
1599:      0xef,    /* ###-#### */
1600:      0x6c,    /* -##-##- */
1601:      0x6c,    /* -##-##- */
1602:      0x6c,    /* -##-##- */
1603:
1604:      /* 0xa0 */
1605:      0x00,    /* ----- */
1606:      0x00,    /* ----- */
1607:      0x00,    /* ----- */
1608:      0x00,    /* ----- */

```

drivers/video/font-lat9-8x8.c

Page 25/39

```

1609:      0x00, /* ----- */
1610:      0x00, /* ----- */
1611:      0xc6, /* ##---##- */
1612:      0xfe, /* #####- */
1613:
1614:          /* 0xa1 */
1615:      0x00, /* ----- */
1616:      0x30, /* --##---- */
1617:      0x00, /* ----- */
1618:      0x30, /* --##---- */
1619:      0x30, /* --##---- */
1620:      0x78, /* -####--- */
1621:      0x78, /* -####--- */
1622:      0x30, /* --##---- */
1623:
1624:          /* 0xa2 */
1625:      0x30, /* --##---- */
1626:      0x78, /* -####--- */
1627:      0xcc, /* ##--##-- */
1628:      0xc0, /* ##----- */
1629:      0xcc, /* ##--##-- */
1630:      0x78, /* -####--- */
1631:      0x30, /* --##---- */
1632:      0x00, /* ----- */
1633:
1634:          /* 0xa3 */
1635:      0x38, /* ---###--- */
1636:      0x6c, /* -##-##-- */
1637:      0x64, /* -##--##- */
1638:      0xf0, /* #####--- */
1639:      0x60, /* -##----- */
1640:      0xe6, /* ###--##- */
1641:      0xfc, /* #####--- */
1642:      0x00, /* ----- */
1643:
1644:          /* 0xa4 */
1645:      0x38, /* ---###--- */
1646:      0x64, /* -##--##- */
1647:      0xf0, /* #####--- */
1648:      0x60, /* -##----- */
1649:      0xf0, /* #####--- */
1650:      0x64, /* -##--##- */
1651:      0x38, /* ---###--- */
1652:      0x00, /* ----- */
1653:
1654:          /* 0xa5 */
1655:      0xcc, /* ##--##-- */
1656:      0xcc, /* ##--##-- */
1657:      0x78, /* -####--- */
1658:      0xfc, /* #####--- */
1659:      0x30, /* --##---- */
1660:      0xfc, /* #####--- */
1661:      0x30, /* --##---- */
1662:      0x30, /* --##---- */
1663:
1664:          /* 0xa6 */
1665:      0x48, /* -#--#--- */
1666:      0x78, /* -####--- */
1667:      0x84, /* #----##- */
1668:      0x60, /* -##----- */
1669:      0x18, /* ----##--- */
1670:      0x84, /* #----##- */
1671:      0x78, /* -####--- */
1672:      0x00, /* ----- */
1673:
1674:          /* 0xa7 */
1675:      0x3e, /* ---#####- */

```

drivers/video/font-lat9-8x8.c

Page 26/39

```

1676:      0x61,    /* -##-----# */
1677:      0x3c,    /* ---####--- */
1678:      0x66,    /* -##--##- */
1679:      0x66,    /* -##--##- */
1680:      0x3c,    /* ---####--- */
1681:      0x86,    /* #-----##- */
1682:      0x7c,    /* -#####--- */
1683:
1684:          /* 0xa8 */
1685:      0x78,    /* -####--- */
1686:      0x00,    /* ----- */
1687:      0x7c,    /* -#####--- */
1688:      0xc0,    /* #------ */
1689:      0x78,    /* -####--- */
1690:      0x0c,    /* ----##-- */
1691:      0xf8,    /* #####--- */
1692:      0x00,    /* ----- */
1693:
1694:          /* 0xa9 */
1695:      0x7c,    /* -#####--- */
1696:      0x82,    /* #------#- */
1697:      0x9a,    /* #-##-##- */
1698:      0xa2,    /* #-#---##- */
1699:      0xa2,    /* #-#---##- */
1700:      0x9a,    /* #-##-##- */
1701:      0x82,    /* #------#- */
1702:      0x7c,    /* -#####--- */
1703:
1704:          /* 0xaa */
1705:      0x3c,    /* ---####--- */
1706:      0x6c,    /* -##-##-- */
1707:      0x3e,    /* ---#####- */
1708:      0x00,    /* ----- */
1709:      0x7e,    /* -#####- */
1710:      0x00,    /* ----- */
1711:      0x00,    /* ----- */
1712:      0x00,    /* ----- */
1713:
1714:          /* 0xab */
1715:      0x00,    /* ----- */
1716:      0x33,    /* --##--## */
1717:      0x66,    /* -##--##- */
1718:      0xcc,    /* ##--##-- */
1719:      0x66,    /* -##--##- */
1720:      0x33,    /* --##--## */
1721:      0x00,    /* ----- */
1722:      0x00,    /* ----- */
1723:
1724:          /* 0xac */
1725:      0x00,    /* ----- */
1726:      0x00,    /* ----- */
1727:      0x00,    /* ----- */
1728:      0xfc,    /* #####--- */
1729:      0x0c,    /* ----##-- */
1730:      0x0c,    /* ----##-- */
1731:      0x00,    /* ----- */
1732:      0x00,    /* ----- */
1733:
1734:          /* 0xad */
1735:      0x00,    /* ----- */
1736:      0x00,    /* ----- */
1737:      0x00,    /* ----- */
1738:      0x7c,    /* -#####--- */
1739:      0x00,    /* ----- */
1740:      0x00,    /* ----- */
1741:      0x00,    /* ----- */
1742:      0x00,    /* ----- */

```

drivers/video/font-lat9-8x8.c

Page 27/39

```

1743:
1744:          /* 0xae          */
1745:    0x7c, /* #####-- */
1746:    0x82, /* #-----# */
1747:    0xb2, /* #-##--# */
1748:    0xaa, /* #-##--# */
1749:    0xb2, /* #-##--# */
1750:    0xaa, /* #-##--# */
1751:    0x82, /* #-----# */
1752:    0x7c, /* #####-- */
1753:
1754:          /* 0xaf          */
1755:    0xff, /* ##### */
1756:    0x00, /* ----- */
1757:    0x00, /* ----- */
1758:    0x00, /* ----- */
1759:    0x00, /* ----- */
1760:    0x00, /* ----- */
1761:    0x00, /* ----- */
1762:    0x00, /* ----- */
1763:
1764:          /* 0xb0          */
1765:    0x70, /* -##----- */
1766:    0xd8, /* #-##----- */
1767:    0x70, /* -##----- */
1768:    0x00, /* ----- */
1769:    0x00, /* ----- */
1770:    0x00, /* ----- */
1771:    0x00, /* ----- */
1772:    0x00, /* ----- */
1773:
1774:          /* 0xb1          */
1775:    0x30, /* --##----- */
1776:    0x30, /* --##----- */
1777:    0xfc, /* #####-- */
1778:    0x30, /* --##----- */
1779:    0x30, /* --##----- */
1780:    0x00, /* ----- */
1781:    0xfc, /* #####-- */
1782:    0x00, /* ----- */
1783:
1784:          /* 0xb2          */
1785:    0x70, /* -##----- */
1786:    0xd8, /* #-##----- */
1787:    0x30, /* --##----- */
1788:    0x60, /* -##----- */
1789:    0xf8, /* #####-- */
1790:    0x00, /* ----- */
1791:    0x00, /* ----- */
1792:    0x00, /* ----- */
1793:
1794:          /* 0xb3          */
1795:    0x70, /* -##----- */
1796:    0xd8, /* #-##----- */
1797:    0x30, /* --##----- */
1798:    0xd8, /* #-##----- */
1799:    0x70, /* -##----- */
1800:    0x00, /* ----- */
1801:    0x00, /* ----- */
1802:    0x00, /* ----- */
1803:
1804:          /* 0xb4          */
1805:    0x6c, /* -##-##-- */
1806:    0xfe, /* #####-- */
1807:    0xcc, /* #-##--# */
1808:    0x18, /* ---##-- */
1809:    0x30, /* --##----- */

```

drivers/video/font-lat9-8x8.c

Page 28/39

```

1810:      0x66,      /* ---##---##- */
1811:      0xfe,      /* #####-#- */
1812:      0x00,      /* ----- */
1813:
1814:          /* 0xb5 */
1815:      0x00,      /* ----- */
1816:      0x00,      /* ----- */
1817:      0xcc,      /* ##---##--- */
1818:      0xcc,      /* ##---##--- */
1819:      0xcc,      /* ##---##--- */
1820:      0xcc,      /* ##---##--- */
1821:      0xf6,      /* ####-##- */
1822:      0xc0,      /* ##----- */
1823:
1824:          /* 0xb6 */
1825:      0x7f,      /* -##### */
1826:      0xdb,      /* ##-##-## */
1827:      0x7b,      /* -####-## */
1828:      0x3b,      /* --###-## */
1829:      0x1b,      /* ----##-## */
1830:      0x1b,      /* ----##-## */
1831:      0x1b,      /* ----##-## */
1832:      0x00,      /* ----- */
1833:
1834:          /* 0xb7 */
1835:      0x00,      /* ----- */
1836:      0x00,      /* ----- */
1837:      0x00,      /* ----- */
1838:      0x18,      /* ----##---- */
1839:      0x18,      /* ----##---- */
1840:      0x00,      /* ----- */
1841:      0x00,      /* ----- */
1842:      0x00,      /* ----- */
1843:
1844:          /* 0xb8 */
1845:      0x78,      /* -####---- */
1846:      0x00,      /* ----- */
1847:      0xfc,      /* #####-- */
1848:      0x98,      /* #--##---- */
1849:      0x30,      /* --##---- */
1850:      0x64,      /* -##--##- */
1851:      0xfc,      /* #####-- */
1852:      0x00,      /* ----- */
1853:
1854:          /* 0xb9 */
1855:      0x60,      /* -##----- */
1856:      0xe0,      /* ###----- */
1857:      0x60,      /* -##----- */
1858:      0x60,      /* -##----- */
1859:      0xf0,      /* ####----- */
1860:      0x00,      /* ----- */
1861:      0x00,      /* ----- */
1862:      0x00,      /* ----- */
1863:
1864:          /* 0xba */
1865:      0x38,      /* --###---- */
1866:      0x6c,      /* -##-##--- */
1867:      0x38,      /* --###---- */
1868:      0x00,      /* ----- */
1869:      0x7c,      /* -#####-- */
1870:      0x00,      /* ----- */
1871:      0x00,      /* ----- */
1872:      0x00,      /* ----- */
1873:
1874:          /* 0xbb */
1875:      0x00,      /* ----- */
1876:      0xcc,      /* ##---##--- */

```


drivers/video/font-lat9-8x8.c

Page 29/39

```

1877:      0x66,    /* ---##---##- */
1878:      0x33,    /* ---##---##- */
1879:      0x66,    /* ---##---##- */
1880:      0xcc,    /* ##---##--- */
1881:      0x00,    /* ----- */
1882:      0x00,    /* ----- */
1883:
1884:          /* 0xbc */
1885:      0x7e,    /* -#####- */
1886:      0xd8,    /* ##-##- --- */
1887:      0xd8,    /* ##-##- --- */
1888:      0xdc,    /* ##-##- --- */
1889:      0xd8,    /* ##-##- --- */
1890:      0xd8,    /* ##-##- --- */
1891:      0x7e,    /* -#####- */
1892:      0x00,    /* ----- */
1893:
1894:          /* 0xbd */
1895:      0x00,    /* ----- */
1896:      0x00,    /* ----- */
1897:      0x7e,    /* -#####- */
1898:      0xdb,    /* ##-##-##- */
1899:      0xde,    /* ##-#####- */
1900:      0xd8,    /* ##-##- --- */
1901:      0x7e,    /* -#####- */
1902:      0x00,    /* ----- */
1903:
1904:          /* 0xbe */
1905:      0xcc,    /* ##---##--- */
1906:      0x00,    /* ----- */
1907:      0xcc,    /* ##---##--- */
1908:      0x78,    /* -####- --- */
1909:      0x30,    /* --##- --- */
1910:      0x30,    /* --##- --- */
1911:      0x78,    /* -####- --- */
1912:      0x00,    /* ----- */
1913:
1914:          /* 0xbf */
1915:      0x00,    /* ----- */
1916:      0x18,    /* ---##- --- */
1917:      0x00,    /* ----- */
1918:      0x18,    /* ---##- --- */
1919:      0x30,    /* --##- --- */
1920:      0x60,    /* -##- --- */
1921:      0x66,    /* -##-##- */
1922:      0x3c,    /* --####- */
1923:
1924:          /* 0xc0 */
1925:      0x60,    /* -##- --- */
1926:      0x30,    /* --##- --- */
1927:      0x78,    /* -####- --- */
1928:      0xcc,    /* ##---##--- */
1929:      0xfc,    /* #####- --- */
1930:      0xcc,    /* ##---##--- */
1931:      0xcc,    /* ##---##--- */
1932:      0x00,    /* ----- */
1933:
1934:          /* 0xc1 */
1935:      0x18,    /* ---##- --- */
1936:      0x30,    /* --##- --- */
1937:      0x78,    /* -####- --- */
1938:      0xcc,    /* ##---##--- */
1939:      0xfc,    /* #####- --- */
1940:      0xcc,    /* ##---##--- */
1941:      0xcc,    /* ##---##--- */
1942:      0x00,    /* ----- */
1943:

```

drivers/video/font-lat9-8x8.c

Page 30/39

```

1944:                /* 0xc2          */
1945:                0x78, /* -###-###- */
1946:                0x84, /* #----#--- */
1947:                0x78, /* -###-###- */
1948:                0xcc, /* ##--##-- */
1949:                0xfc, /* #####--- */
1950:                0xcc, /* ##--##-- */
1951:                0xcc, /* ##--##-- */
1952:                0x00, /* ----- */
1953:
1954:                /* 0xc3          */
1955:                0x76, /* -###-###- */
1956:                0xdc, /* #-###--- */
1957:                0x78, /* -###-###- */
1958:                0xcc, /* ##--##-- */
1959:                0xfc, /* #####--- */
1960:                0xcc, /* ##--##-- */
1961:                0xcc, /* ##--##-- */
1962:                0x00, /* ----- */
1963:
1964:                /* 0xc4          */
1965:                0xcc, /* ##--##-- */
1966:                0x00, /* ----- */
1967:                0x78, /* -###-###- */
1968:                0xcc, /* ##--##-- */
1969:                0xfc, /* #####--- */
1970:                0xcc, /* ##--##-- */
1971:                0xcc, /* ##--##-- */
1972:                0x00, /* ----- */
1973:
1974:                /* 0xc5          */
1975:                0x30, /* --##----- */
1976:                0x48, /* -##-#----- */
1977:                0x78, /* -###-###- */
1978:                0xcc, /* ##--##-- */
1979:                0xfc, /* #####--- */
1980:                0xcc, /* ##--##-- */
1981:                0xcc, /* ##--##-- */
1982:                0x00, /* ----- */
1983:
1984:                /* 0xc6          */
1985:                0x3e, /* --#####- */
1986:                0x78, /* -###-###- */
1987:                0xd8, /* #-##-###- */
1988:                0xfc, /* #####--- */
1989:                0xd8, /* #-##-###- */
1990:                0xd8, /* #-##-###- */
1991:                0xde, /* #-#####- */
1992:                0x00, /* ----- */
1993:
1994:                /* 0xc7          */
1995:                0x3c, /* --#####- */
1996:                0x66, /* -##-##-###- */
1997:                0xc0, /* #------ */
1998:                0xc0, /* ##----- */
1999:                0x66, /* -##-##-###- */
2000:                0x3c, /* --#####- */
2001:                0x0c, /* ----##--- */
2002:                0x78, /* -###-###- */
2003:
2004:                /* 0xc8          */
2005:                0x60, /* -##-#----- */
2006:                0x30, /* --##-#----- */
2007:                0xfe, /* #####---#- */
2008:                0x62, /* -##-###-#- */
2009:                0x78, /* -###-###- */
2010:                0x62, /* -##-###-#- */

```

drivers/video/font-lat9-8x8.c

Page 31/39

```

2011:      0xfe, /* #####- */
2012:      0x00, /* ----- */
2013:
2014:          /* 0xc9 */
2015:      0x0c, /* ----##-- */
2016:      0x18, /* ---##--- */
2017:      0xfe, /* #####- */
2018:      0x62, /* -##---#- */
2019:      0x78, /* -####--- */
2020:      0x62, /* -##---#- */
2021:      0xfe, /* #####- */
2022:      0x00, /* ----- */
2023:
2024:          /* 0xca */
2025:      0x38, /* --###--- */
2026:      0x6c, /* -##-##-- */
2027:      0xfe, /* #####- */
2028:      0x62, /* -##---#- */
2029:      0x78, /* -####--- */
2030:      0x62, /* -##---#- */
2031:      0xfe, /* #####- */
2032:      0x00, /* ----- */
2033:
2034:          /* 0xcb */
2035:      0x6c, /* -##-##-- */
2036:      0x00, /* ----- */
2037:      0xfe, /* #####- */
2038:      0x62, /* -##---#- */
2039:      0x78, /* -####--- */
2040:      0x62, /* -##---#- */
2041:      0xfe, /* #####- */
2042:      0x00, /* ----- */
2043:
2044:          /* 0xcc */
2045:      0x60, /* -##----- */
2046:      0x30, /* --##----- */
2047:      0x78, /* -####--- */
2048:      0x30, /* --##----- */
2049:      0x30, /* --##----- */
2050:      0x30, /* --##----- */
2051:      0x78, /* -####--- */
2052:      0x00, /* ----- */
2053:
2054:          /* 0xcd */
2055:      0x18, /* ---##--- */
2056:      0x30, /* --##----- */
2057:      0x78, /* -####--- */
2058:      0x30, /* --##----- */
2059:      0x30, /* --##----- */
2060:      0x30, /* --##----- */
2061:      0x78, /* -####--- */
2062:      0x00, /* ----- */
2063:
2064:          /* 0xce */
2065:      0x78, /* -####--- */
2066:      0xcc, /* ##--##-- */
2067:      0x78, /* -####--- */
2068:      0x30, /* --##----- */
2069:      0x30, /* --##----- */
2070:      0x30, /* --##----- */
2071:      0x78, /* -####--- */
2072:      0x00, /* ----- */
2073:
2074:          /* 0xcf */
2075:      0xcc, /* ##--##-- */
2076:      0x00, /* ----- */
2077:      0x78, /* -####--- */

```

drivers/video/font-lat9-8x8.c

Page 32/39

```

2078:      0x30, /* --##----- */
2079:      0x30, /* --##----- */
2080:      0x30, /* --##----- */
2081:      0x78, /* -####----- */
2082:      0x00, /* ----- */
2083:
2084:          /* 0xd0 */
2085:      0xf8, /* #####----- */
2086:      0x6c, /* -##-##-- */
2087:      0x66, /* -##--##- */
2088:      0xf6, /* #####-##- */
2089:      0x66, /* -##-##- */
2090:      0x6c, /* -##-##-- */
2091:      0xf8, /* #####----- */
2092:      0x00, /* ----- */
2093:
2094:          /* 0xd1 */
2095:      0x76, /* -###-##- */
2096:      0xdc, /* #-###-- */
2097:      0xe6, /* ###--##- */
2098:      0xf6, /* #####-##- */
2099:      0xde, /* #-#####- */
2100:      0xce, /* ##--##- */
2101:      0xc6, /* #----##- */
2102:      0x00, /* ----- */
2103:
2104:          /* 0xd2 */
2105:      0x60, /* -##----- */
2106:      0x30, /* --##----- */
2107:      0x78, /* -####----- */
2108:      0xcc, /* #-##-- */
2109:      0xcc, /* #-##-- */
2110:      0xcc, /* #-##-- */
2111:      0x78, /* -####----- */
2112:      0x00, /* ----- */
2113:
2114:          /* 0xd3 */
2115:      0x18, /* ---##---- */
2116:      0x30, /* --##----- */
2117:      0x78, /* -####----- */
2118:      0xcc, /* #-##-- */
2119:      0xcc, /* #-##-- */
2120:      0xcc, /* #-##-- */
2121:      0x78, /* -####----- */
2122:      0x00, /* ----- */
2123:
2124:          /* 0xd4 */
2125:      0x78, /* -####----- */
2126:      0xcc, /* #-##-- */
2127:      0x78, /* -####----- */
2128:      0xcc, /* #-##-- */
2129:      0xcc, /* #-##-- */
2130:      0xcc, /* #-##-- */
2131:      0x78, /* -####----- */
2132:      0x00, /* ----- */
2133:
2134:          /* 0xd5 */
2135:      0x76, /* -###-##- */
2136:      0xdc, /* #-###-- */
2137:      0x78, /* -####----- */
2138:      0xcc, /* #-##-- */
2139:      0xcc, /* #-##-- */
2140:      0xcc, /* #-##-- */
2141:      0x78, /* -####----- */
2142:      0x00, /* ----- */
2143:
2144:          /* 0xd6 */

```

drivers/video/font-lat9-8x8.c

Page 33/39

```

2145:      0xcc, /* ##--##-- */
2146:      0x00, /* ----- */
2147:      0x78, /* -###- - */
2148:      0xcc, /* ##--##-- */
2149:      0xcc, /* ##--##-- */
2150:      0xcc, /* ##--##-- */
2151:      0x78, /* -###- - */
2152:      0x00, /* ----- */
2153:
2154:          /* 0xd7 */
2155:      0x00, /* ----- */
2156:      0x6c, /* -##-##- */
2157:      0x38, /* --##- - */
2158:      0x38, /* --##- - */
2159:      0x6c, /* -##-##- */
2160:      0x00, /* ----- */
2161:      0x00, /* ----- */
2162:      0x00, /* ----- */
2163:
2164:          /* 0xd8 */
2165:      0x3e, /* --####- */
2166:      0x6c, /* -##-##- */
2167:      0xde, /* ##-####- */
2168:      0xd6, /* ##-##-##- */
2169:      0xf6, /* ####-##- */
2170:      0x6c, /* -##-##- */
2171:      0xf8, /* #####- - */
2172:      0x00, /* ----- */
2173:
2174:          /* 0xd9 */
2175:      0x60, /* -##----- */
2176:      0x30, /* --##- - - */
2177:      0xcc, /* ##--##-- */
2178:      0xcc, /* ##--##-- */
2179:      0xcc, /* ##--##-- */
2180:      0xcc, /* ##--##-- */
2181:      0x78, /* -###- - - */
2182:      0x00, /* ----- */
2183:
2184:          /* 0xda */
2185:      0x18, /* ---##- - - */
2186:      0x30, /* --##- - - */
2187:      0xcc, /* ##--##-- */
2188:      0xcc, /* ##--##-- */
2189:      0xcc, /* ##--##-- */
2190:      0xcc, /* ##--##-- */
2191:      0x78, /* -###- - - */
2192:      0x00, /* ----- */
2193:
2194:          /* 0xdb */
2195:      0x78, /* -###- - - */
2196:      0xcc, /* ##--##-- */
2197:      0x00, /* ----- */
2198:      0xcc, /* ##--##-- */
2199:      0xcc, /* ##--##-- */
2200:      0xcc, /* ##--##-- */
2201:      0x78, /* -###- - - */
2202:      0x00, /* ----- */
2203:
2204:          /* 0xdc */
2205:      0xcc, /* ##--##-- */
2206:      0x00, /* ----- */
2207:      0xcc, /* ##--##-- */
2208:      0xcc, /* ##--##-- */
2209:      0xcc, /* ##--##-- */
2210:      0xcc, /* ##--##-- */
2211:      0x78, /* -###- - - */

```

drivers/video/font-lat9-8x8.c

Page 34/39

```

2212:      0x00,    /* ----- */
2213:
2214:      /* 0xdd */
2215:      0x18,    /* ---##--- */
2216:      0x30,    /* --##---- */
2217:      0xcc,    /* ##--##-- */
2218:      0xcc,    /* ##--##-- */
2219:      0x78,    /* -####--- */
2220:      0x30,    /* ---##---- */
2221:      0x78,    /* -####--- */
2222:      0x00,    /* ----- */
2223:
2224:      /* 0xde */
2225:      0xf0,    /* #####---- */
2226:      0x60,    /* -##----- */
2227:      0x7c,    /* -#####-- */
2228:      0x66,    /* -##--##- */
2229:      0x7c,    /* -#####-- */
2230:      0x60,    /* -##----- */
2231:      0xf0,    /* #####---- */
2232:      0x00,    /* ----- */
2233:
2234:      /* 0xdf */
2235:      0x7c,    /* -#####-- */
2236:      0xc6,    /* ##---##- */
2237:      0xc6,    /* ##---##- */
2238:      0xcc,    /* ##--##-- */
2239:      0xc6,    /* ##---##- */
2240:      0xd6,    /* ##-##-##- */
2241:      0xdc,    /* ##-###-- */
2242:      0x80,    /* #------ */
2243:
2244:      /* 0xe0 */
2245:      0x60,    /* -##----- */
2246:      0x30,    /* --##---- */
2247:      0x78,    /* -####--- */
2248:      0x0c,    /* ----##-- */
2249:      0x7c,    /* -#####-- */
2250:      0xcc,    /* ##--##-- */
2251:      0x76,    /* -###-##- */
2252:      0x00,    /* ----- */
2253:
2254:      /* 0xe1 */
2255:      0x18,    /* ---##--- */
2256:      0x30,    /* --##---- */
2257:      0x78,    /* -####--- */
2258:      0x0c,    /* ----##-- */
2259:      0x7c,    /* -#####-- */
2260:      0xcc,    /* ##--##-- */
2261:      0x76,    /* -###-##- */
2262:      0x00,    /* ----- */
2263:
2264:      /* 0xe2 */
2265:      0x78,    /* -####--- */
2266:      0x84,    /* #-----##- */
2267:      0x78,    /* -####--- */
2268:      0x0c,    /* ----##-- */
2269:      0x7c,    /* -#####-- */
2270:      0xcc,    /* ##--##-- */
2271:      0x76,    /* -###-##- */
2272:      0x00,    /* ----- */
2273:
2274:      /* 0xe3 */
2275:      0x76,    /* -###-##- */
2276:      0xdc,    /* ##-###-- */
2277:      0x78,    /* -####--- */
2278:      0x0c,    /* ----##--

```

drivers/video/font-lat9-8x8.c

Page 35/39

```

2279:      0x7c, /* #####-- */
2280:      0xcc, /* ##---##-- */
2281:      0x76, /* ---###-##- */
2282:      0x00, /* ----- */
2283:
2284:          /* 0xe4 */
2285:      0x6c, /* -##-##-- */
2286:      0x00, /* ----- */
2287:      0x78, /* -####--- */
2288:      0x0c, /* ----##-- */
2289:      0x7c, /* -#####-- */
2290:      0xcc, /* ##---##-- */
2291:      0x76, /* ---###-##- */
2292:      0x00, /* ----- */
2293:
2294:          /* 0xe5 */
2295:      0x38, /* --###--- */
2296:      0x6c, /* -##-##-- */
2297:      0x78, /* -####--- */
2298:      0x0c, /* ----##-- */
2299:      0x7c, /* -#####-- */
2300:      0xcc, /* ##---##-- */
2301:      0x76, /* ---###-##- */
2302:      0x00, /* ----- */
2303:
2304:          /* 0xe6 */
2305:      0x00, /* ----- */
2306:      0x00, /* ----- */
2307:      0x7e, /* -#####- */
2308:      0x1b, /* ---##-## */
2309:      0x7e, /* -#####- */
2310:      0xd8, /* ##-##--- */
2311:      0x7e, /* -#####- */
2312:      0x00, /* ----- */
2313:
2314:          /* 0xe7 */
2315:      0x00, /* ----- */
2316:      0x78, /* -####--- */
2317:      0xcc, /* ##---##-- */
2318:      0xc0, /* ##----- */
2319:      0xcc, /* ##---##-- */
2320:      0x78, /* -####--- */
2321:      0x18, /* ----##--- */
2322:      0x70, /* -###----- */
2323:
2324:          /* 0xe8 */
2325:      0x60, /* -##----- */
2326:      0x30, /* --##----- */
2327:      0x78, /* -####--- */
2328:      0xcc, /* ##---##-- */
2329:      0xfc, /* #####--- */
2330:      0xc0, /* ##----- */
2331:      0x78, /* -####--- */
2332:      0x00, /* ----- */
2333:
2334:          /* 0xe9 */
2335:      0x0c, /* ----##-- */
2336:      0x18, /* ----##--- */
2337:      0x78, /* -####--- */
2338:      0xcc, /* ##---##-- */
2339:      0xfc, /* #####--- */
2340:      0xc0, /* ##----- */
2341:      0x78, /* -####--- */
2342:      0x00, /* ----- */
2343:
2344:          /* 0xea */
2345:      0x78, /* -####--- */

```

drivers/video/font-lat9-8x8.c

Page 36/39

```

2346:      0x84,    /* #----#-- */
2347:      0x78,    /* -####--- */
2348:      0xcc,    /* ##--##-- */
2349:      0xfc,    /* #####-- */
2350:      0xc0,    /* ##----- */
2351:      0x78,    /* -####--- */
2352:      0x00,    /* ----- */
2353:
2354:          /* 0xeb          */
2355:      0xcc,    /* ##--##-- */
2356:      0x00,    /* ----- */
2357:      0x78,    /* -####--- */
2358:      0xcc,    /* ##--##-- */
2359:      0xfc,    /* #####-- */
2360:      0xc0,    /* ##----- */
2361:      0x78,    /* -####--- */
2362:      0x00,    /* ----- */
2363:
2364:          /* 0xec          */
2365:      0x60,    /* -##----- */
2366:      0x30,    /* --##----- */
2367:      0x00,    /* ----- */
2368:      0x70,    /* -###----- */
2369:      0x30,    /* --##----- */
2370:      0x30,    /* --##----- */
2371:      0x78,    /* -####--- */
2372:      0x00,    /* ----- */
2373:
2374:          /* 0xed          */
2375:      0x18,    /* ---##----- */
2376:      0x30,    /* --##----- */
2377:      0x00,    /* ----- */
2378:      0x70,    /* -###----- */
2379:      0x30,    /* --##----- */
2380:      0x30,    /* --##----- */
2381:      0x78,    /* -####--- */
2382:      0x00,    /* ----- */
2383:
2384:          /* 0xee          */
2385:      0x70,    /* -###----- */
2386:      0xd8,    /* ##-##----- */
2387:      0x00,    /* ----- */
2388:      0x70,    /* -###----- */
2389:      0x30,    /* --##----- */
2390:      0x30,    /* --##----- */
2391:      0x78,    /* -####--- */
2392:      0x00,    /* ----- */
2393:
2394:          /* 0xef          */
2395:      0x00,    /* ----- */
2396:      0xd8,    /* ##-##----- */
2397:      0x00,    /* ----- */
2398:      0x70,    /* -###----- */
2399:      0x30,    /* --##----- */
2400:      0x30,    /* --##----- */
2401:      0x78,    /* -####--- */
2402:      0x00,    /* ----- */
2403:
2404:          /* 0xf0          */
2405:      0x78,    /* -####--- */
2406:      0x70,    /* -###----- */
2407:      0x18,    /* ---##----- */
2408:      0x7c,    /* -#####-- */
2409:      0xcc,    /* ##--##-- */
2410:      0xcc,    /* ##--##-- */
2411:      0x78,    /* -####--- */
2412:      0x00,    /* ----- */

```


drivers/video/font-lat9-8x8.c

Page 37/39

```

2413:
2414:          /* 0xf1          */
2415:    0x76, /* ---###--- */
2416:    0xdc, /* ##-###-- */
2417:    0xf8, /* #####--- */
2418:    0xcc, /* ##--##-- */
2419:    0xcc, /* ##--##-- */
2420:    0xcc, /* ##--##-- */
2421:    0xcc, /* ##--##-- */
2422:    0x00, /* ----- */
2423:
2424:          /* 0xf2          */
2425:    0x60, /* -##----- */
2426:    0x30, /* --##----- */
2427:    0x00, /* ----- */
2428:    0x78, /* -####--- */
2429:    0xcc, /* ##--##-- */
2430:    0xcc, /* ##--##-- */
2431:    0x78, /* -####--- */
2432:    0x00, /* ----- */
2433:
2434:          /* 0xf3          */
2435:    0x18, /* ---##---- */
2436:    0x30, /* --##----- */
2437:    0x00, /* ----- */
2438:    0x78, /* -####--- */
2439:    0xcc, /* ##--##-- */
2440:    0xcc, /* ##--##-- */
2441:    0x78, /* -####--- */
2442:    0x00, /* ----- */
2443:
2444:          /* 0xf4          */
2445:    0x78, /* -####--- */
2446:    0xcc, /* ##--##-- */
2447:    0x00, /* ----- */
2448:    0x78, /* -####--- */
2449:    0xcc, /* ##--##-- */
2450:    0xcc, /* ##--##-- */
2451:    0x78, /* -####--- */
2452:    0x00, /* ----- */
2453:
2454:          /* 0xf5          */
2455:    0x76, /* ---###--- */
2456:    0xdc, /* ##-###-- */
2457:    0x00, /* ----- */
2458:    0x78, /* -####--- */
2459:    0xcc, /* ##--##-- */
2460:    0xcc, /* ##--##-- */
2461:    0x78, /* -####--- */
2462:    0x00, /* ----- */
2463:
2464:          /* 0xf6          */
2465:    0x00, /* ----- */
2466:    0xcc, /* ##--##-- */
2467:    0x00, /* ----- */
2468:    0x78, /* -####--- */
2469:    0xcc, /* ##--##-- */
2470:    0xcc, /* ##--##-- */
2471:    0x78, /* -####--- */
2472:    0x00, /* ----- */
2473:
2474:          /* 0xf7          */
2475:    0x00, /* ----- */
2476:    0x30, /* --##----- */
2477:    0x00, /* ----- */
2478:    0xfc, /* #####--- */
2479:    0x00, /* ----- */

```

drivers/video/font-lat9-8x8.c

Page 38/39

```

2480:      0x30, /* --##----- */
2481:      0x00, /* ----- */
2482:      0x00, /* ----- */
2483:
2484:          /* 0xf8 */
2485:      0x00, /* ----- */
2486:      0x00, /* ----- */
2487:      0x7c, /* #####-- */
2488:      0xdc, /* ##-##-#-- */
2489:      0xfc, /* #####-- */
2490:      0xec, /* ##-##-#-- */
2491:      0xf8, /* #####-- */
2492:      0x00, /* ----- */
2493:
2494:          /* 0xf9 */
2495:      0x60, /* -##----- */
2496:      0x30, /* --##----- */
2497:      0xcc, /* ##--##-#-- */
2498:      0xcc, /* ##--##-#-- */
2499:      0xcc, /* ##--##-#-- */
2500:      0xcc, /* ##--##-#-- */
2501:      0x76, /* -###-##-#-- */
2502:      0x00, /* ----- */
2503:
2504:          /* 0xfa */
2505:      0x18, /* ---##---- */
2506:      0x30, /* --##----- */
2507:      0xcc, /* ##--##-#-- */
2508:      0xcc, /* ##--##-#-- */
2509:      0xcc, /* ##--##-#-- */
2510:      0xcc, /* ##--##-#-- */
2511:      0x76, /* -###-##-#-- */
2512:      0x00, /* ----- */
2513:
2514:          /* 0xfb */
2515:      0x78, /* -####---- */
2516:      0xcc, /* ##--##-#-- */
2517:      0x00, /* ----- */
2518:      0xcc, /* ##--##-#-- */
2519:      0xcc, /* ##--##-#-- */
2520:      0xcc, /* ##--##-#-- */
2521:      0x76, /* -###-##-#-- */
2522:      0x00, /* ----- */
2523:
2524:          /* 0xfc */
2525:      0xcc, /* ##--##-#-- */
2526:      0x00, /* ----- */
2527:      0xcc, /* ##--##-#-- */
2528:      0xcc, /* ##--##-#-- */
2529:      0xcc, /* ##--##-#-- */
2530:      0xcc, /* ##--##-#-- */
2531:      0x76, /* -###-##-#-- */
2532:      0x00, /* ----- */
2533:
2534:          /* 0xfd */
2535:      0x18, /* ---##---- */
2536:      0x30, /* --##----- */
2537:      0xcc, /* ##--##-#-- */
2538:      0xcc, /* ##--##-#-- */
2539:      0xcc, /* ##--##-#-- */
2540:      0x7c, /* #####-- */
2541:      0x0c, /* ----##-#-- */
2542:      0xf8, /* #####-- */
2543:
2544:          /* 0xfe */
2545:      0xf0, /* #####---- */
2546:      0x60, /* -##----- */

```

drivers/video/font-lat9-8x8.c

Page 39/39

```

2547:         0x78,    /* -###- --- */
2548:         0x6c,    /* -##-##- --- */
2549:         0x6c,    /* -##-##- --- */
2550:         0x78,    /* -###- --- */
2551:         0x60,    /* -##- ---- */
2552:         0xf0,    /* ####- --- */
2553:
2554:             /* 0xff          */
2555:         0xcc,    /* ##-##- --- */
2556:         0x00,    /* ----- */
2557:         0xcc,    /* ##-##- --- */
2558:         0xcc,    /* ##-##- --- */
2559:         0xcc,    /* ##-##- --- */
2560:         0x7c,    /* -#####- --- */
2561:         0x0c,    /* ----##- --- */
2562:         0xf8,    /* #####- --- */
2563:
2564: };
2565:
2566: static unsigned char cursorshape_8x8[] = {
2567:     0x00,    /* ----- */
2568:     0x00,    /* ----- */
2569:     0x00,    /* ----- */
2570:     0x00,    /* ----- */
2571:     0x00,    /* ----- */
2572:     0x00,    /* ----- */
2573:     0xFF,    /* ##### */
2574:     0xFF,    /* ##### */
2575: };
2576:
2577: struct fbcon_font_desc font_lat9_8x8 = {
2578:     "VGA8x8",
2579:     8,
2580:     8,
2581:     fontdata_8x8,
2582:     cursorshape_8x8
2583: };

```

drivers/video/fonts.c

Page 1/1

```
1: /*
2:  * fiwix/drivers/char/fonts.c
3:  *
4:  * Copyright 2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/string.h>
9: #include <fiwix/font.h>
10: #include "font-lat9-8x8.c"
11: #include "font-lat9-8x10.c"
12: #include "font-lat9-8x12.c"
13: #include "font-lat9-8x14.c"
14: #include "font-lat9-8x16.c"
15:
16: #define NR_FONTS (sizeof(fbcon_fonts) / sizeof(*fbcon_fonts))
17:
18: static struct fbcon_font_desc *fbcon_fonts[] = {
19:     &font_lat9_8x8,
20:     &font_lat9_8x10,
21:     &font_lat9_8x12,
22:     &font_lat9_8x14,
23:     &font_lat9_8x16
24: };
25:
26: struct fbcon_font_desc * fbcon_find_font(int height)
27: {
28:     int n;
29:
30:     for(n = 0; n < NR_FONTS; n++) {
31:         if(fbcon_fonts[n]->height == height) {
32:             return fbcon_fonts[n];
33:         }
34:     }
35:
36:     return NULL;
37: }
```

drivers/video/Makefile

Page 1/1

```
1: # fiwix/drivers/video/Makefile
2: #
3: # Copyright 2021, Jordi Sanfeliu. All rights reserved.
4: # Distributed under the terms of the Fiwix License.
5: #
6:
7: .S.o:
8:     $(CC) -traditional -I$(INCLUDE) -c -o $@ $<
9: .c.o:
10:     $(CC) $(CFLAGS) -c -o $@ $<
11:
12: OBJS = video_init.o vgacon.o fbcon.o fonts.o
13:
14: video:  $(OBJS)
15:           $(LD) $(LDFLAGS) -r $(OBJS) -o video.o
16:
17: clean:
18:           rm -f *.o
19:
```

drivers/video/vgacon.c

Page 1/6

```

1: /*
2:  * fiwix/drivers/char/vgacon.c
3:  *
4:  * Copyright 2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/config.h>
10: #include <fiwix/segments.h>
11: #include <fiwix/vgacon.h>
12: #include <fiwix/console.h>
13: #include <fiwix/timer.h>
14: #include <fiwix/stdio.h>
15: #include <fiwix/string.h>
16:
17: /* ISO/IEC 8859-1:1998 (aka latin1, IBM819, CP819), same as in Linux */
18: static const char *iso8859 =
19:     "\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"
20:     "\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"
21:     " !\"#$%&'()*+,-./0123456789:;<=>?"
22:     "@ABCDEFGHIJKLMNPOQRSTUVWXYZ[\\]^_"
23:     "`abcdefghijklmnopqrstuvwxyz{|}~\0"
24:     "\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"
25:     "\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"
26:     "\377\255\233\234\376\235\174\025\376\376\246\256\252\055\376\376"
27:     "\370\361\375\376\376\346\024\371\376\376\247\257\254\253\376\250"
28:     "\376\376\376\376\216\217\222\200\376\220\376\376\376\376\376\376"
29:     "\376\245\376\376\376\376\231\376\350\376\376\376\232\376\376\341"
30:     "\205\240\203\376\204\206\221\207\212\202\210\211\215\241\214\213"
31:     "\376\244\225\242\223\376\224\366\355\227\243\226\201\376\376\230"
32: ;
33:
34: struct video_parms video;
35: static unsigned char screen_is_off = 0;
36:
37: void vgacon_put_char(struct vconsole *vc, unsigned char ch)
38: {
39:     short int *vidmem, *screen;
40:
41:     ch = iso8859[ch];
42:     screen = vc->screen;
43:
44:     if(!(vc->flags & CONSOLE_HAS_FOCUS)) {
45:         screen[(vc->y * vc->columns) + vc->x] = vc->color_attr | ch;
46:         return;
47:     }
48:
49:     vidmem = (short int *)vc->vidmem;
50:     vidmem[(vc->y * vc->columns) + vc->x] = vc->color_attr | ch;
51:     screen[(vc->y * vc->columns) + vc->x] = vc->color_attr | ch;
52:     vcbuf[(video.buf_y * vc->columns) + vc->x] = vc->color_attr | ch;
53: }
54:
55: void vgacon_insert_char(struct vconsole *vc)
56: {
57:     int n, offset;
58:     short int tmp, last_char, *vidmem, *screen;
59:
60:     vidmem = (short int *)vc->vidmem;
61:     screen = vc->screen;
62:     offset = (vc->y * vc->columns) + vc->x;
63:     n = vc->x;
64:     last_char = BLANK_MEM;
65:
66:     while(n++ < vc->columns) {
67:         if(vc->flags & CONSOLE_HAS_FOCUS) {

```

drivers/video/vgacon.c

Page 2/6

```

68:             memcpy_w(&tmp, vidmem + offset, 1);
69:             memset_w(vidmem + offset, last_char, 1);
70:         }
71:         memcpy_w(&tmp, screen + offset, 1);
72:         memset_w(screen + offset, last_char, 1);
73:         last_char = tmp;
74:         offset++;
75:     }
76: }
77:
78: void vgacon_delete_char(struct vconsole *vc)
79: {
80:     int offset, count;
81:     short int *vidmem, *screen;
82:
83:     vidmem = (short int *)vc->vidmem;
84:     screen = vc->screen;
85:     offset = (vc->y * vc->columns) + vc->x;
86:     count = vc->columns - vc->x;
87:
88:     if(vc->flags & CONSOLE_HAS_FOCUS) {
89:         memcpy_w(vidmem + offset, vidmem + offset + 1, count);
90:         memset_w(vidmem + offset + count, BLANK_MEM, 1);
91:     }
92:     memcpy_w(screen + offset, screen + offset + 1, count);
93:     memset_w(screen + offset + count, BLANK_MEM, 1);
94: }
95:
96: void vgacon_update_curpos(struct vconsole *vc)
97: {
98:     unsigned short int curpos;
99:
100:    if(vc->flags & CONSOLE_HAS_FOCUS) {
101:        curpos = (vc->y * vc->columns) + vc->x;
102:        output_b(video.port + CRT_INDEX, CRT_CURSOR_POS_HI);
103:        output_b(video.port + CRT_DATA, (curpos >> 8) & 0xFF);
104:        output_b(video.port + CRT_INDEX, CRT_CURSOR_POS_LO);
105:        output_b(video.port + CRT_DATA, (curpos & 0xFF));
106:    }
107: }
108:
109: void vgacon_show_cursor(struct vconsole *vc, int mode)
110: {
111:     int status;
112:
113:     switch(mode) {
114:         case COND:
115:             if(!(video.flags & VPF_CURSOR_ON)) {
116:                 break;
117:             }
118:             /* fall through */
119:         case ON:
120:             output_b(video.port + CRT_INDEX, CRT_CURSOR_STR);
121:             status = input_b(video.port + CRT_DATA);
122:             output_b(video.port + CRT_DATA, status & CURSOR_MASK);
123:             video.flags |= VPF_CURSOR_ON;
124:             break;
125:         case OFF:
126:             output_b(video.port + CRT_INDEX, CRT_CURSOR_STR);
127:             status = input_b(video.port + CRT_DATA);
128:             output_b(video.port + CRT_DATA, status | CURSOR_DISABLE
);
129:             video.flags &= ~VPF_CURSOR_ON;
130:             break;
131:     }
132: }
133:

```

drivers/video/vgacon.c

Page 3/6

```

134: void vgacon_get_curpos(struct vconsole *vc)
135: {
136:     unsigned short int curpos;
137:
138:     outport_b(video.port + CRT_INDEX, CRT_CURSOR_POS_HI);
139:     curpos = inport_b(video.port + CRT_DATA) << 8;
140:     outport_b(video.port + CRT_INDEX, CRT_CURSOR_POS_LO);
141:     curpos |= inport_b(video.port + CRT_DATA);
142:
143:     vc->x = curpos % vc->columns;
144:     vc->y = curpos / vc->columns;
145: }
146:
147: void vgacon_write_screen(struct vconsole *vc, int from, int count, short int col
or)
148: {
149:     short int *vidmem, *screen;
150:
151:     screen = vc->screen;
152:     if(!(vc->flags & CONSOLE_HAS_FOCUS)) {
153:         memset_w(screen + from, color, count);
154:         return;
155:     }
156:
157:     vidmem = (short int *)vc->vidmem;
158:     memset_w(vidmem + from, color, count);
159:     memset_w(screen + from, color, count);
160: }
161:
162: void vgacon_blank_screen(struct vconsole *vc)
163: {
164:     short int *vidmem;
165:
166:     if(vc->flags & CONSOLE_BLANKED) {
167:         return;
168:     }
169:
170:     if(vc->flags & CONSOLE_HAS_FOCUS) {
171:         vidmem = (short int *)vc->vidmem;
172:         memset_w(vidmem, BLANK_MEM, SCREEN_SIZE);
173:     }
174:     vc->flags |= CONSOLE_BLANKED;
175:     vgacon_show_cursor(vc, OFF);
176: }
177:
178: void vgacon_scroll_screen(struct vconsole *vc, int top, int mode)
179: {
180:     int n, offset, count;
181:     short int *vidmem, *screen;
182:
183:     vidmem = (short int *)vc->vidmem;
184:     screen = vc->screen;
185:
186:     if(!top) {
187:         top = vc->top;
188:     }
189:     switch(mode) {
190:         case SCROLL_UP:
191:             count = vc->columns * (vc->lines - top - 1);
192:             offset = top * vc->columns;
193:             top = (top + 1) * vc->columns;
194:             if(vc->flags & CONSOLE_HAS_FOCUS) {
195:                 memcpy_w(vidmem + offset, vidmem + top, count);
196:                 memset_w(vidmem + offset + count, BLANK_MEM, vc-
>columns);
197:             }
198:             memcpy_w(screen + offset, screen + top, count);

```


drivers/video/vgacon.c

Page 4/6

```

199:             memset_w(screen + offset + count, BLANK_MEM, vc->columns
);
200:             break;
201:         case SCROLL_DOWN:
202:             count = vc->columns;
203:             for(n = vc->lines - 1; n > top; n--) {
204:                 memcpy_w(screen + (vc->columns * n), screen + (v
c->columns * (n - 1)), count);
205:                 if(vc->flags & CONSOLE_HAS_FOCUS) {
206:                     memcpy_w(vidmem + (vc->columns * n), scr
een + (vc->columns * (n - 1)), count);
207:                 }
208:             }
209:             memset_w(screen + (top * vc->columns), BLANK_MEM, count)
;
210:             if(vc->flags & CONSOLE_HAS_FOCUS) {
211:                 memset_w(vidmem + (top * vc->columns), BLANK_MEM
, count);
212:             }
213:             break;
214:         }
215:         return;
216:     }
217:
218: void vgacon_restore_screen(struct vconsole *vc)
219: {
220:     short int *vidmem;
221:
222:     if(vc->flags & CONSOLE_HAS_FOCUS) {
223:         vidmem = (short int *)vc->vidmem;
224:         memcpy_w(vidmem, vc->screen, SCREEN_SIZE);
225:     }
226: }
227:
228: void vgacon_screen_on(struct vconsole *vc)
229: {
230:     unsigned long int flags;
231:     struct callout_req creq;
232:
233:     if(screen_is_off) {
234:         SAVE_FLAGS(flags); CLI();
235:         inport_b(INPUT_STAT1);
236:         inport_b(0x3BA);
237:         outport_b(ATTR_CONTROLLER, ATTR_CONTROLLER_PAS);
238:         RESTORE_FLAGS(flags);
239:     }
240:
241:     if(BLANK_INTERVAL) {
242:         creq.fn = vgacon_screen_off;
243:         creq.arg = 0;
244:         add_callout(&creq, BLANK_INTERVAL);
245:     }
246: }
247:
248: void vgacon_screen_off(unsigned int arg)
249: {
250:     unsigned long int flags;
251:
252:     screen_is_off = 1;
253:     SAVE_FLAGS(flags); CLI();
254:     inport_b(INPUT_STAT1);
255:     inport_b(0x3BA);
256:     outport_b(ATTR_CONTROLLER, 0);
257:     RESTORE_FLAGS(flags);
258: }
259:
260: void vgacon_buf_scroll(struct vconsole *vc, int mode)

```

drivers/video/vgacon.c

Page 5/6

```

261: {
262:     short int *vidmem;
263:
264:     if(video.buf_y <= SCREEN_LINES) {
265:         return;
266:     }
267:
268:     vidmem = (short int *)vc->vidmem;
269:     if(mode == SCROLL_UP) {
270:         if(video.buf_top < 0) {
271:             return;
272:         }
273:         if(!video.buf_top) {
274:             video.buf_top = (video.buf_y - SCREEN_LINES + 1) * SCREE
N_COLS;
275:         }
276:         video.buf_top -= (SCREEN_LINES / 2) * SCREEN_COLS;
277:         if(video.buf_top < 0) {
278:             video.buf_top = 0;
279:         }
280:         memcpy_w(vidmem, vcbuf + video.buf_top, SCREEN_SIZE);
281:         if(!video.buf_top) {
282:             video.buf_top = -1;
283:         }
284:         vgacon_show_cursor(vc, OFF);
285:         return;
286:     }
287:     if(mode == SCROLL_DOWN) {
288:         if(!video.buf_top) {
289:             return;
290:         }
291:         if(video.buf_top == video.buf_y * SCREEN_COLS) {
292:             return;
293:         }
294:         if(video.buf_top < 0) {
295:             video.buf_top = 0;
296:         }
297:         video.buf_top += (SCREEN_LINES / 2) * SCREEN_COLS;
298:         if(video.buf_top >= (video.buf_y - SCREEN_LINES + 1) * SCREEN_CO
LS) {
299:             vgacon_restore_screen(vc);
300:             video.buf_top = 0;
301:             vgacon_show_cursor(vc, ON);
302:             vgacon_update_curpos(vc);
303:             return;
304:         }
305:         memcpy_w(vidmem, vcbuf + video.buf_top, SCREEN_SIZE);
306:         return;
307:     }
308: }
309:
310: void vgacon_cursor_blink(unsigned int arg)
311: {
312:     /* not used */
313: }
314:
315: void vgacon_init(void)
316: {
317:     short int *bios_data;
318:
319:     /* get the VGA type from the BIOS equipment information */
320:     bios_data = (short int *) (KERNEL_BASE_ADDR + 0x410);
321:     if((*bios_data & 0x30) == 0x30) {
322:         /* monochrome = 0x30 */
323:         video.address = (void *) MONO_ADDR;
324:         video.port = MONO_6845_ADDR;
325:         strcpy((char *) video.signature, "VGA monochrome 80x25");

```

drivers/video/vgacon.c

Page 6/6

```
326:         } else {
327:             /* color = 0x00 || 0x20 */
328:             video.address = (void *)COLOR_ADDR;
329:             video.port = COLOR_6845_ADDR;
330:             strcpy((char *)video.signature, "VGA color 80x25");
331:         }
332:
333:         /* some parameters already set in multiboot.c */
334:
335:         video.put_char = vgacon_put_char;
336:         video.insert_char = vgacon_insert_char;
337:         video.delete_char = vgacon_delete_char;
338:         video.update_curpos = vgacon_update_curpos;
339:         video.show_cursor = vgacon_show_cursor;
340:         video.get_curpos = vgacon_get_curpos;
341:         video.write_screen = vgacon_write_screen;
342:         video.blank_screen = vgacon_blank_screen;
343:         video.scroll_screen = vgacon_scroll_screen;
344:         video.restore_screen = vgacon_restore_screen;
345:         video.screen_on = vgacon_screen_on;
346:         video.buf_scroll = vgacon_buf_scroll;
347:         video.cursor_blink = vgacon_cursor_blink;
348:
349:         memcpy_w(vcbuf, video.address, SCREEN_SIZE * 2);
350:     }
```

drivers/video/video_init.c

Page 1/1

```
1: /*
2:  * fiwix/drivers/char/video_init.c
3:  *
4:  * Copyright 2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/config.h>
10: #include <fiwix/vgacon.h>
11: #include <fiwix/fb.h>
12: #include <fiwix/fbcon.h>
13: #include <fiwix/console.h>
14: #include <fiwix/stdio.h>
15: #include <fiwix/string.h>
16:
17: struct video_parms video;
18:
19: void video_init(void)
20: {
21:     memset_b(vcbuf, 0, (video.columns * video.lines * SCREENS_LOG * 2 * size
of(short int)));
22:
23:     if(video.flags & VPF_VGA) {
24:         vgacon_init();
25:     }
26:     if(video.flags & VPF_VESA_FB) {
27:         fb_init();
28:         fbcon_init();
29:     }
30: }
```

fs/buffer.c

Page 1/8

```

1: /*
2:  * fiwix/fs/buffer.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: /*
9:  * buffer.c implements a cache using the LRU (Least Recently Used) algorithm,
10:  * with a free list as a doubly circular linked list and a chained hash table
11:  * with doubly linked lists.
12:  *
13:  * hash table
14:  * +-----+ +-----+ +-----+ +-----+
15:  * | index | |prev|data|next| |prev|data|next| |prev|data|next|
16:  * |  0   --> | / |   |   |   | <--- |   |   |   | <--- |   |   | / |
17:  * +-----+ +-----+ +-----+ +-----+
18:  * +-----+ +-----+ +-----+ +-----+
19:  * | index | |prev|data|next| |prev|data|next| |prev|data|next|
20:  * |  1   --> | / |   |   |   | <--- |   |   |   | <--- |   | / |
21:  * +-----+ +-----+ +-----+ +-----+
22:  *                               (buffer)       (buffer)       (buffer)
23:  *      ...
24:  */
25:
26: #include <fiwix/asm.h>
27: #include <fiwix/kernel.h>
28: #include <fiwix/sleep.h>
29: #include <fiwix/sched.h>
30: #include <fiwix/buffer.h>
31: #include <fiwix/devices.h>
32: #include <fiwix/fs.h>
33: #include <fiwix/mm.h>
34: #include <fiwix/errno.h>
35: #include <fiwix/stdio.h>
36: #include <fiwix/string.h>
37: #include <fiwix/stat.h>
38:
39: #define BUFFER_HASH(dev, block) (((__dev_t)(dev) ^ (__blk_t)(block)) % (NR_BUF_H
ASH))
40: #define NR_BUFFERS      (buffer_table_size / sizeof(struct buffer))
41: #define NR_BUF_HASH    (buffer_hash_table_size / sizeof(unsigned int))
42:
43: struct buffer *buffer_table;          /* buffer pool */
44: struct buffer *buffer_head;          /* buffer pool head */
45: struct buffer *buffer_dirty_head;
46: struct buffer **buffer_hash_table;
47:
48: static struct resource sync_resource = { NULL, NULL };
49:
50: static void insert_to_hash(struct buffer *buf)
51: {
52:     struct buffer **h;
53:     int i;
54:
55:     i = BUFFER_HASH(buf->dev, buf->block);
56:     h = &buffer_hash_table[i];
57:
58:     if(!*h) {
59:         *h = buf;
60:         (*h)->prev_hash = (*h)->next_hash = NULL;
61:     } else {
62:         buf->prev_hash = NULL;
63:         buf->next_hash = *h;
64:         (*h)->prev_hash = buf;
65:         *h = buf;
66:     }

```

fs/buffer.c

Page 2/8

```

67: }
68:
69: static void remove_from_hash(struct buffer *buf)
70: {
71:     struct buffer **h;
72:     int i;
73:
74:     i = BUFFER_HASH(buf->dev, buf->block);
75:     h = &buffer_hash_table[i];
76:
77:     while(*h) {
78:         if(*h == buf) {
79:             if((*h)->next_hash) {
80:                 (*h)->next_hash->prev_hash = (*h)->prev_hash;
81:             }
82:             if((*h)->prev_hash) {
83:                 (*h)->prev_hash->next_hash = (*h)->next_hash;
84:             }
85:             if(h == &buffer_hash_table[i]) {
86:                 *h = (*h)->next_hash;
87:             }
88:             break;
89:         }
90:         h = &(*h)->next_hash;
91:     }
92: }
93:
94: static void insert_on_dirty_list(struct buffer *buf)
95: {
96:     if(buf->prev_dirty || buf->next_dirty) {
97:         return;
98:     }
99:
100:    if(buffer_dirty_head) {
101:        buf->next_dirty = buffer_dirty_head;
102:        buffer_dirty_head->prev_dirty = buf;
103:    }
104:    buffer_dirty_head = buf;
105:    kstat.dirty += (PAGE_SIZE / 1024);
106: }
107:
108: static void remove_from_dirty_list(struct buffer *buf)
109: {
110:     if(buf->next_dirty) {
111:         buf->next_dirty->prev_dirty = buf->prev_dirty;
112:     }
113:     if(buf->prev_dirty) {
114:         buf->prev_dirty->next_dirty = buf->next_dirty;
115:     }
116:     if(buf == buffer_dirty_head) {
117:         buffer_dirty_head = buf->next_dirty;
118:     }
119:     buf->prev_dirty = buf->next_dirty = NULL;
120:     buf->flags &= ~BUFFER_DIRTY;
121:     kstat.dirty -= (PAGE_SIZE / 1024);
122: }
123:
124: static void insert_on_free_list(struct buffer *buf)
125: {
126:     if(!buffer_head) {
127:         buf->prev_free = buf->next_free = buf;
128:         buffer_head = buf;
129:     } else {
130:         buf->next_free = buffer_head;
131:         buf->prev_free = buffer_head->prev_free;
132:         buffer_head->prev_free->next_free = buf;
133:         buffer_head->prev_free = buf;

```

fs/buffer.c

Page 3/8

```
134:
135:         /*
136:         * If is marked as not valid then the buffer is
137:         * placed at the beginning of the free list.
138:         */
139:         if(!(buf->flags & BUFFER_VALID)) {
140:             buffer_head = buf;
141:         }
142:     }
143: }
144:
145: static void remove_from_free_list(struct buffer *buf)
146: {
147:     if(!buffer_head) {
148:         return;
149:     }
150:
151:     buf->prev_free->next_free = buf->next_free;
152:     buf->next_free->prev_free = buf->prev_free;
153:     if(buf == buffer_head) {
154:         buffer_head = buf->next_free;
155:     }
156:
157:     if(buffer_head == buffer_head->next_free) {
158:         buffer_head = NULL;
159:     }
160: }
161:
162: static void buffer_wait(struct buffer *buf)
163: {
164:     unsigned long int flags;
165:
166:     for(;;) {
167:         SAVE_FLAGS(flags); CLI();
168:         if(buf->flags & BUFFER_LOCKED) {
169:             RESTORE_FLAGS(flags);
170:             sleep(&buffer_wait, PROC_UNINTERRUPTIBLE);
171:         } else {
172:             break;
173:         }
174:     }
175:     buf->flags |= BUFFER_LOCKED;
176:     RESTORE_FLAGS(flags);
177: }
178:
179: static struct buffer * get_free_buffer(void)
180: {
181:     unsigned long int flags;
182:     struct buffer *buf;
183:
184:     /* no more buffers on free list */
185:     if(!buffer_head) {
186:         return NULL;
187:     }
188:
189:     for(;;) {
190:         SAVE_FLAGS(flags); CLI();
191:         buf = buffer_head;
192:         if(buf->flags & BUFFER_LOCKED) {
193:             RESTORE_FLAGS(flags);
194:             sleep(&buffer_wait, PROC_UNINTERRUPTIBLE);
195:         } else {
196:             break;
197:         }
198:     }
199:
200:     remove_from_free_list(buf);
```

fs/buffer.c

Page 4/8

```

201:         buf->flags |= BUFFER_LOCKED;
202:
203:         RESTORE_FLAGS(flags);
204:         return buf;
205: }
206:
207: static void sync_one_buffer(struct buffer *buf)
208: {
209:     struct device *d;
210:     int errno;
211:
212:     if(!(d = get_device(BLK_DEV, buf->dev))) {
213:         printk("WARNING: %s(): block device %d,%d not registered!\n", __
FUNCTION__, MAJOR(buf->dev), MINOR(buf->dev));
214:         return;
215:     }
216:
217:     /* this shouldn't happen */
218:     if(!buf->data) {
219:         printk("WARNING: %s(): buffer (dev=%x, block=%d, size=%d) don't
has data!\n", __FUNCTION__, buf->dev, buf->block, buf->size);
220:         return;
221:     }
222:
223:     if(d->fsop && d->fsop->write_block) {
224:         errno = d->fsop->write_block(buf->dev, buf->block, buf->data, bu
f->size);
225:         if(errno < 0) {
226:             if(errno == -EROFS) {
227:                 printk("WARNING: %s(): write protection on devic
e %d,%d.\n", __FUNCTION__, MAJOR(buf->dev), MINOR(buf->dev), buf->block);
228:             } else {
229:                 printk("WARNING: %s(): I/O error on device %d,%d
.\n", __FUNCTION__, MAJOR(buf->dev), MINOR(buf->dev), buf->block);
230:             }
231:             return;
232:         }
233:         remove_from_dirty_list(buf);
234:     } else {
235:         printk("WARNING: %s(): device %d,%d does not have the write_bloc
k() method!\n", __FUNCTION__, MAJOR(buf->dev), MINOR(buf->dev));
236:     }
237: }
238:
239: static struct buffer * search_buffer_hash(__dev_t dev, __blk_t block, int size)
240: {
241:     struct buffer *buf;
242:     int i;
243:
244:     i = BUFFER_HASH(dev, block);
245:     buf = buffer_hash_table[i];
246:
247:     while(buf) {
248:         if(buf->dev == dev && buf->block == block && buf->size == size)
{
249:             return buf;
250:         }
251:         buf = buf->next_hash;
252:     }
253:
254:     return NULL;
255: }
256:
257: static struct buffer * getblk(__dev_t dev, __blk_t block, int size)
258: {
259:     unsigned long int flags;
260:     struct buffer *buf;

```


fs/buffer.c

Page 5/8

```

261:
262:     for(;;) {
263:         if((buf = search_buffer_hash(dev, block, size)) {
264:             SAVE_FLAGS(flags); CLI();
265:             if(buf->flags & BUFFER_LOCKED) {
266:                 RESTORE_FLAGS(flags);
267:                 sleep(&buffer_wait, PROC_UNINTERRUPTIBLE);
268:                 continue;
269:             }
270:             buf->flags |= BUFFER_LOCKED;
271:             remove_from_free_list(buf);
272:             RESTORE_FLAGS(flags);
273:             return buf;
274:         }
275:
276:         if(!(buf = get_free_buffer())) {
277:             printk("WARNING: %s(): no more buffers on free list!\n",
__FUNCTION__);
278:             sleep(&get_free_buffer, PROC_UNINTERRUPTIBLE);
279:             continue;
280:         }
281:
282:         if(buf->flags & BUFFER_DIRTY) {
283:             sync_one_buffer(buf);
284:         } else {
285:             if(!buf->data) {
286:                 if(!(buf->data = (char *)kmalloc())) {
287:                     brelse(buf);
288:                     printk("%s(): returning NULL\n", __FUNCT
ION__);
289:                     return NULL;
290:                 }
291:                 kstat.buffers += (PAGE_SIZE / 1024);
292:             }
293:         }
294:
295:         SAVE_FLAGS(flags); CLI();
296:         remove_from_hash(buf); /* remove it from old hash */
297:         buf->dev = dev;
298:         buf->block = block;
299:         buf->size = size;
300:         insert_to_hash(buf);
301:         buf->flags &= ~BUFFER_VALID;
302:         RESTORE_FLAGS(flags);
303:         return buf;
304:     }
305: }
306:
307: struct buffer * bread(__dev_t dev, __blk_t block, int size)
308: {
309:     struct buffer *buf;
310:     struct device *d;
311:
312:     if(!(d = get_device(BLK_DEV, dev))) {
313:         printk("WARNING: %s(): device major %d not found!\n", __FUNCTION
__, MAJOR(dev));
314:         return NULL;
315:     }
316:
317:     if((buf = getblk(dev, block, size)) {
318:         if(buf->flags & BUFFER_VALID) {
319:             return buf;
320:         }
321:         if(d->fsop && d->fsop->read_block) {
322:             if(d->fsop->read_block(dev, block, buf->data, size) >= 0
) {
323:                 buf->flags |= BUFFER_VALID;

```

fs/buffer.c

Page 6/8

```

324:         }
325:     }
326:     if(buf->flags & BUFFER_VALID) {
327:         return buf;
328:     }
329:     brelse(buf);
330: }
331:
332: printk("WARNING: %s(): returning NULL!\n", __FUNCTION__);
333: return NULL;
334: }
335:
336: void bwrite(struct buffer *buf)
337: {
338:     buf->flags |= (BUFFER_DIRTY | BUFFER_VALID);
339:     brelse(buf);
340: }
341:
342: void brelse(struct buffer *buf)
343: {
344:     unsigned long int flags;
345:
346:     SAVE_FLAGS(flags); CLI();
347:
348:     if(buf->flags & BUFFER_DIRTY) {
349:         insert_on_dirty_list(buf);
350:     }
351:
352:     insert_on_free_list(buf);
353:     buf->flags &= ~BUFFER_LOCKED;
354:
355:     RESTORE_FLAGS(flags);
356:
357:     wakeup(&get_free_buffer);
358:     wakeup(&buffer_wait);
359: }
360:
361: void sync_buffers(__dev_t dev)
362: {
363:     struct buffer *buf, *next;
364:
365:     buf = buffer_dirty_head;
366:
367:     lock_resource(&sync_resource);
368:     while(buf) {
369:         next = buf->next_dirty;
370:         if(!dev || buf->dev == dev) {
371:             buffer_wait(buf);
372:             sync_one_buffer(buf);
373:             buf->flags &= ~BUFFER_LOCKED;
374:             wakeup(&buffer_wait);
375:         }
376:         buf = next;
377:     }
378:     unlock_resource(&sync_resource);
379: }
380:
381: void invalidate_buffers(__dev_t dev)
382: {
383:     unsigned long int flags;
384:     unsigned int n;
385:     struct buffer *buf;
386:
387:     buf = &buffer_table[0];
388:     SAVE_FLAGS(flags); CLI();
389:
390:     for(n = 0; n < NR_BUFFERS; n++) {

```

fs/buffer.c

Page 7/8

```

391:         if (!(buf->flags & BUFFER_LOCKED) && buf->dev == dev) {
392:             buffer_wait(buf);
393:             remove_from_hash(buf);
394:             buf->flags &= ~(BUFFER_VALID | BUFFER_LOCKED);
395:             wakeup(&buffer_wait);
396:         }
397:         buf++;
398:     }
399:
400:     RESTORE_FLAGS(flags);
401:     /* FIXME: invalidate_pages(dev); */
402: }
403:
404: /*
405:  * When kernel runs out of pages, kswapd is awoken and it calls this function
406:  * which goes through the buffer cache, freeing up to NR_BUF_RECLAIM buffers.
407:  */
408: int reclaim_buffers(void)
409: {
410:     struct buffer *buf, *first;
411:     int reclaimed;
412:
413:     reclaimed = 0;
414:     first = NULL;
415:
416:     for(;;) {
417:         if (!(buf = get_free_buffer())) {
418:             printk("WARNING: %s(): no more buffers on free list!\n",
FUNCTION__);
419:             sleep(&get_free_buffer, PROC_UNINTERRUPTIBLE);
420:             continue;
421:         }
422:
423:         if(buf->flags & BUFFER_DIRTY) {
424:             sync_one_buffer(buf);
425:         }
426:
427:         /* this ensures the buffer will go to the tail */
428:         buf->flags |= BUFFER_VALID;
429:
430:         if(first) {
431:             if(first == buf) {
432:                 brelse(buf);
433:                 break;
434:             }
435:         } else {
436:             first = buf;
437:         }
438:         if(buf->data) {
439:             kfree((unsigned int)buf->data);
440:             buf->data = NULL;
441:             remove_from_hash(buf);
442:             kstat.buffers -= (PAGE_SIZE / 1024);
443:             reclaimed++;
444:             if(reclaimed == NR_BUF_RECLAIM) {
445:                 brelse(buf);
446:                 break;
447:             }
448:         }
449:         brelse(buf);
450:     }
451:
452:     wakeup(&buffer_wait);
453:
454:     /*
455:      * If the total number of buffers reclaimed was less or equal to
456:      * NR_BUF_RECLAIM, then wakeup any process waiting for a new page

```

fs/buffer.c

Page 8/8

```
457:         * because release_page() won't do it.
458:         */
459:         if(reclaimed && reclaimed <= NR_BUF_RECLAIM) {
460:             wakeup(&get_free_page);
461:         }
462:
463:         return reclaimed;
464:     }
465:
466: void buffer_init(void)
467: {
468:     struct buffer *buf;
469:     unsigned int n;
470:
471:     memset_b(buffer_table, NULL, buffer_table_size);
472:     memset_b(buffer_hash_table, NULL, buffer_hash_table_size);
473:
474:     for(n = 0; n < NR_BUFFERS; n++) {
475:         buf = &buffer_table[n];
476:         insert_on_free_list(buf);
477:     }
478: }
```

fs/devices.c

Page 1/6

```

1: /*
2:  * fiwix/fs/devices.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/errno.h>
10: #include <fiwix/buffer.h>
11: #include <fiwix/devices.h>
12: #include <fiwix/fs.h>
13: #include <fiwix/mm.h>
14: #include <fiwix/process.h>
15: #include <fiwix/stdio.h>
16: #include <fiwix/string.h>
17:
18: struct device *chr_device_table[NR_CHRDEV];
19: struct device *blk_device_table[NR_BLKDEV];
20:
21: struct fs_operations def_chr_fsop = {
22:     0,
23:     0,
24:
25:     chr_dev_open,
26:     NULL,                /* close */
27:     NULL,                /* read */
28:     NULL,                /* write */
29:     NULL,                /* ioctl */
30:     NULL,                /* lseek */
31:     NULL,                /* readdir */
32:     NULL,                /* mmap */
33:     NULL,                /* select */
34:
35:     NULL,                /* readlink */
36:     NULL,                /* followlink */
37:     NULL,                /* bmap */
38:     NULL,                /* lockup */
39:     NULL,                /* rmdir */
40:     NULL,                /* link */
41:     NULL,                /* unlink */
42:     NULL,                /* symlink */
43:     NULL,                /* mkdir */
44:     NULL,                /* mknod */
45:     NULL,                /* truncate */
46:     NULL,                /* create */
47:     NULL,                /* rename */
48:
49:     NULL,                /* read_block */
50:     NULL,                /* write_block */
51:
52:     NULL,                /* read_inode */
53:     NULL,                /* write_inode */
54:     NULL,                /* ialloc */
55:     NULL,                /* ifree */
56:     NULL,                /* stats */
57:     NULL,                /* read_superblock */
58:     NULL,                /* remount_fs */
59:     NULL,                /* write_superblock */
60:     NULL,                /* release_superblock */
61: };
62:
63: struct fs_operations def_blk_fsop = {
64:     0,
65:     0,
66:
67:     blk_dev_open,

```

fs/devices.c

Page 2/6

```

68:      blk_dev_close,
69:      blk_dev_read,
70:      blk_dev_write,
71:      blk_dev_ioctl,
72:      blk_dev_lseek,
73:      NULL,                /* readdir */
74:      NULL,                /* mmap */
75:      NULL,                /* select */
76:
77:      NULL,                /* readlink */
78:      NULL,                /* followlink */
79:      NULL,                /* bmap */
80:      NULL,                /* lockup */
81:      NULL,                /* rmdir */
82:      NULL,                /* link */
83:      NULL,                /* unlink */
84:      NULL,                /* symlink */
85:      NULL,                /* mkdir */
86:      NULL,                /* mknod */
87:      NULL,                /* truncate */
88:      NULL,                /* create */
89:      NULL,                /* rename */
90:
91:      NULL,                /* read_block */
92:      NULL,                /* write_block */
93:
94:      NULL,                /* read_inode */
95:      NULL,                /* write_inode */
96:      NULL,                /* ialloc */
97:      NULL,                /* ifree */
98:      NULL,                /* stats */
99:      NULL,                /* read_superblock */
100:     NULL,                /* remount_fs */
101:     NULL,                /* write_superblock */
102:     NULL,                /* release_superblock */
103: };
104:
105: int register_device(int type, struct device *new_d)
106: {
107:     struct device **d;
108:     int n, minors;
109:
110:     switch(type) {
111:         case CHR_DEV:
112:             if(new_d->major >= NR_CHRDEV) {
113:                 printk("%s(): character device major %d is greater than NR_CHRDEV (%d).\n", __FUNCTION__, new_d->major, NR_CHRDEV);
114:                 return 1;
115:             }
116:             d = &chr_device_table[new_d->major];
117:             break;
118:         case BLK_DEV:
119:             if(new_d->major >= NR_BLKDEV) {
120:                 printk("%s(): block device major %d is greater than NR_BLKDEV (%d).\n", __FUNCTION__, new_d->major, NR_BLKDEV);
121:                 return 1;
122:             }
123:             d = &blk_device_table[new_d->major];
124:             break;
125:         default:
126:             printk("WARNING: %s(): invalid device type %d.\n", __FUNCTION__, type);
127:             return 1;
128:             break;
129:     }
130:
131:     /* make sure there are minors defined */

```

fs/devices.c

Page 3/6

```

132:         for(n = 0, minors = 0; n < 8; n++) {
133:             minors += new_d->minors[n];
134:         }
135:         if(!minors) {
136:             printk("WARNING: %s(): device major %d with no defined minors.\n",
", __FUNCTION__, new_d->major);
137:             return 1;
138:         }
139:
140:         if(*d) {
141:             if(&(*d)->minors == &new_d->minors || (&(*d)->next && &(*d)->next->minors == &new_d->minors)) {
142:                 printk("WARNING: %s(): duplicated device major %d.\n",
__FUNCTION__, new_d->major);
143:                 return 1;
144:             }
145:             do {
146:                 d = &(*d)->next;
147:             } while(*d);
148:         }
149:         *d = new_d;
150:
151:         return 0;
152: }
153:
154: struct device * get_device(int type, __dev_t dev)
155: {
156:     char *name;
157:     unsigned char major;
158:     struct device *d;
159:
160:     major = MAJOR(dev);
161:
162:     switch(type) {
163:         case CHR_DEV:
164:             if(major >= NR_CHRDEV) {
165:                 printk("%s(): character device major %d is great
er than NR_CHRDEV (%d).\n", __FUNCTION__, major, NR_CHRDEV);
166:                 return NULL;
167:             }
168:             d = chr_device_table[major];
169:             name = "character";
170:             break;
171:         case BLK_DEV:
172:             if(major >= NR_BLKDEV) {
173:                 printk("%s(): block device major %d is greater t
han NR_BLKDEV (%d).\n", __FUNCTION__, major, NR_BLKDEV);
174:                 return NULL;
175:             }
176:             d = blk_device_table[major];
177:             name = "block";
178:             break;
179:         default:
180:             printk("WARNING: %s(): invalid device type %d.\n", __FUN
CTION__, type);
181:             return NULL;
182:         }
183:
184:         while(d) {
185:             if(d->major == major) {
186:                 if(TEST_MINOR(d->minors, MINOR(dev))) {
187:                     return d;
188:                 }
189:                 d = d->next;
190:                 continue;
191:             }
192:             break;

```

fs/devices.c

Page 4/6

```

193:         }
194:
195:         printk("WARNING: %s(): %s device %d,%d not found.\n", __FUNCTION__, name
, major, MINOR(dev));
196:         return NULL;
197:     }
198:
199: int chr_dev_open(struct inode *i, struct fd *fd_table)
200: {
201:     struct device *d;
202:
203:     if((d = get_device(CHR_DEV, i->rdev)) {
204:         i->fsop = d->fsop;
205:         if(i->fsop && i->fsop->open) {
206:             return i->fsop->open(i, fd_table);
207:         }
208:         return -EINVAL;
209:     }
210:
211:     return -ENXIO;
212: }
213:
214: int blk_dev_open(struct inode *i, struct fd *fd_table)
215: {
216:     struct device *d;
217:
218:     if((d = get_device(BLK_DEV, i->rdev)) {
219:         if(d->fsop && d->fsop->open) {
220:             return d->fsop->open(i, fd_table);
221:         }
222:         return -EINVAL;
223:     }
224:
225:     return -ENXIO;
226: }
227:
228: int blk_dev_close(struct inode *i, struct fd *fd_table)
229: {
230:     struct device *d;
231:
232:     if((d = get_device(BLK_DEV, i->rdev)) {
233:         if(d->fsop && d->fsop->close) {
234:             return d->fsop->close(i, fd_table);
235:         }
236:         printk("WARNING: %s(): block device %d,%d does not have the clos
e() method.\n", __FUNCTION__, MAJOR(i->rdev), MINOR(i->rdev));
237:         return -EINVAL;
238:     }
239:
240:     return -ENXIO;
241: }
242:
243: int blk_dev_read(struct inode *i, struct fd *fd_table, char *buffer, __size_t co
unt)
244: {
245:     __blk_t block;
246:     __off_t total_read;
247:     unsigned long long int device_size;
248:     int blksize;
249:     unsigned int boffset, bytes;
250:     struct buffer *buf;
251:     struct device *d;
252:
253:     if(!(d = get_device(BLK_DEV, i->rdev)) {
254:         return -ENXIO;
255:     }
256:

```


fs/devices.c

Page 5/6

```

257:         blksize = d->blksize ? d->blksize : BLKSIZE_1K;
258:         total_read = 0;
259:         if(!d->device_data) {
260:             printk("%s(): don't know the size of the block device %d,%d.\n",
__FUNCTION__, MAJOR(i->rdev), MINOR(i->rdev));
261:             return -EIO;
262:         }
263:
264:         device_size = ((unsigned int *)d->device_data)[MINOR(i->rdev)];
265:         device_size *= 1024LLU;
266:
267:         count = (fd_table->offset + count > device_size) ? device_size - fd_table->offset : count;
268:         if(!count || fd_table->offset > device_size) {
269:             return 0;
270:         }
271:         while(count) {
272:             boffset = fd_table->offset % blksize;
273:             block = (fd_table->offset / blksize);
274:             if(!(buf = bread(i->rdev, block, blksize))) {
275:                 return -EIO;
276:             }
277:             bytes = blksize - boffset;
278:             bytes = MIN(bytes, count);
279:             memcpy_b(buffer + total_read, buf->data + boffset, bytes);
280:             total_read += bytes;
281:             count -= bytes;
282:             fd_table->offset += bytes;
283:             brelse(buf);
284:         }
285:         return total_read;
286:     }
287:
288: int blk_dev_write(struct inode *i, struct fd *fd_table, const char *buffer, __size_t count)
289: {
290:     __blk_t block;
291:     __off_t total_written;
292:     unsigned long long int device_size;
293:     int blksize;
294:     unsigned int boffset, bytes;
295:     struct buffer *buf;
296:     struct device *d;
297:
298:     if(!(d = get_device(BLK_DEV, i->rdev))) {
299:         return -ENXIO;
300:     }
301:
302:     blksize = d->blksize ? d->blksize : BLKSIZE_1K;
303:     total_written = 0;
304:     if(!d->device_data) {
305:         printk("%s(): don't know the size of the block device %d,%d.\n",
__FUNCTION__, MAJOR(i->rdev), MINOR(i->rdev));
306:         return -EIO;
307:     }
308:
309:     device_size = ((unsigned int *)d->device_data)[MINOR(i->rdev)];
310:     device_size *= 1024LLU;
311:
312:     count = (fd_table->offset + count > device_size) ? device_size - fd_table->offset : count;
313:     if(!count || fd_table->offset > device_size) {
314:         return -ENOSPC;
315:     }
316:     while(count) {
317:         boffset = fd_table->offset % blksize;
318:         block = (fd_table->offset / blksize);

```

fs/devices.c

Page 6/6

```

319:         if (!(buf = bread(i->rdev, block, blksize))) {
320:             return -EIO;
321:         }
322:         bytes = blksize - boffset;
323:         bytes = MIN(bytes, count);
324:         memcpy_b(buf->data + boffset, buffer + total_written, bytes);
325:         total_written += bytes;
326:         count -= bytes;
327:         fd_table->offset += bytes;
328:         bwrite(buf);
329:     }
330:     return total_written;
331: }
332:
333: int blk_dev_ioctl(struct inode *i, int cmd, unsigned long int arg)
334: {
335:     struct device *d;
336:
337:     if ((d = get_device(BLK_DEV, i->rdev)) {
338:         if (d->fsop && d->fsop->ioctl) {
339:             return d->fsop->ioctl(i, cmd, arg);
340:         }
341:         printk("WARNING: %s(): block device %d,%d does not have the ioc
l() method.\n", __FUNCTION__, MAJOR(i->rdev), MINOR(i->rdev));
342:         return -EINVAL;
343:     }
344:
345:     return -ENXIO;
346: }
347:
348: int blk_dev_lseek(struct inode *i, __off_t offset)
349: {
350:     struct device *d;
351:
352:     if ((d = get_device(BLK_DEV, i->rdev)) {
353:         if (d->fsop && d->fsop->lseek) {
354:             return d->fsop->lseek(i, offset);
355:         }
356:     }
357:
358:     return offset;
359: }
360:
361: void dev_init(void)
362: {
363:     memset_b(chr_device_table, NULL, sizeof(chr_device_table));
364:     memset_b(blk_device_table, NULL, sizeof(blk_device_table));
365: }

```

```

1: /*
2:  * fiwix/fs/elf.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/asm.h>
10: #include <fiwix/types.h>
11: #include <fiwix/buffer.h>
12: #include <fiwix/fs.h>
13: #include <fiwix/i386elf.h>
14: #include <fiwix/mm.h>
15: #include <fiwix/mman.h>
16: #include <fiwix/fs.h>
17: #include <fiwix/fcntl.h>
18: #include <fiwix/process.h>
19: #include <fiwix/errno.h>
20: #include <fiwix/stdio.h>
21: #include <fiwix/string.h>
22:
23: #define AT_ITEMS          12          /* ELF Auxiliary Vectors */
24:
25: static int check_elf(struct elf32_hdr *elf32_h)
26: {
27:     if(elf32_h->e_ident[EI_MAG0] != ELFMAG0 ||
28:        elf32_h->e_ident[EI_MAG1] != ELFMAG1 ||
29:        elf32_h->e_ident[EI_MAG2] != ELFMAG2 ||
30:        elf32_h->e_ident[EI_MAG3] != ELFMAG3 ||
31:        (elf32_h->e_type != ET_EXEC && elf32_h->e_type != ET_DYN) ||
32:        elf32_h->e_machine != EM_386) {
33:         return -EINVAL;
34:     }
35:     return 0;
36: }
37:
38: /*
39:  * Setup the initial process stack (System V ABI for i386)
40:  * -----
41:  * 0xBFFFFFFF
42:  *   +-----+
43:  *   | envp[] str |
44:  *   +-----+
45:  *   | argv[] str |
46:  *   +-----+
47:  *   | NULL      |
48:  *   +-----+
49:  *   | ELF Aux.Vect. |
50:  *   +-----+
51:  *   | NULL      |
52:  *   +-----+
53:  *   | envp[] ptr |
54:  *   +-----+
55:  *   | NULL      |
56:  *   +-----+
57:  *   | argv[] ptr |
58:  *   +-----+
59:  *   | argc      |
60:  *   +-----+
61:  *   | stack pointer | grows toward lower addresses
62:  *   +-----+ ||
63:  *   | ..... | \
64:  *   | ..... |
65:  *   | ..... |
66:  *   | ..... | /\
67:  *   +-----+ ||

```

elf_create_stack() setups this section

fs/elf.c

Page 2/10

```

68: *      | brk (heap) | grows toward higher addresses
69: *      +-----+
70: *      | .bss section |
71: *      +-----+
72: *      | .data section |
73: *      +-----+
74: *      | .text section |
75: *      +-----+
76: * 0x08048000
77: */
78: static void elf_create_stack(struct binargs *barg, unsigned int *sp, unsigned in
t str_ptr, int at_base, struct elf32_hdr *elf32_h, unsigned int phdr_addr)
79: {
80:     unsigned int n, addr;
81:     char *str;
82:
83:     /* copy strings */
84:     for(n = 0; n < ARG_MAX; n++) {
85:         if(barg->page[n]) {
86:             addr = KERNEL_BASE_ADDR - ((ARG_MAX - n) * PAGE_SIZE);
87:             memcpy_b((void *)addr, (void *)barg->page[n], PAGE_SIZE)
;
88:         }
89:     }
90:
91: #ifdef __DEBUG__
92:     printk("sp = 0x%08x\n", sp);
93: #endif /*__DEBUG__ */
94:
95:     /* copy the value of 'argc' into the stack */
96:     current->argc = barg->argc;
97:     *sp = barg->argc;
98: #ifdef __DEBUG__
99:     printk("at 0x%08x -> argc\n", sp);
100: #endif /*__DEBUG__ */
101:     sp++;
102:
103:     /* copy as many pointers to strings as 'argv' */
104:     current->argv = (char **)sp;
105:     for(n = 0; n < barg->argc; n++) {
106:         *sp = str_ptr;
107:         str = (char *)str_ptr;
108: #ifdef __DEBUG__
109:         printk("at 0x%08x -> str_ptr(%d) = 0x%08x (+ %d)\n", sp, n, str_
ptr, strlen(str) + 1);
110: #endif /*__DEBUG__ */
111:         sp++;
112:         str_ptr += strlen(str) + 1;
113:     }
114:
115:     /* the last element of 'argv[]' must be a NULL-pointer */
116:     *sp = NULL;
117: #ifdef __DEBUG__
118:     printk("at 0x%08x -> ----- = 0x%08x\n", sp, 0);
119: #endif /*__DEBUG__ */
120:     sp++;
121:
122:     /* copy as many pointers to strings as 'envc' */
123:     current->envc = barg->envc;
124:     current->envp = (char **)sp;
125:     for(n = 0; n < barg->envc; n++) {
126:         *sp = str_ptr;
127:         str = (char *)str_ptr;
128: #ifdef __DEBUG__
129:         printk("at 0x%08x -> str_ptr(%d) = 0x%08x (+ %d)\n", sp, n, str_
ptr, strlen(str) + 1);
130: #endif /*__DEBUG__ */

```

```

131:             sp++;
132:             str_ptr += strlen(str) + 1;
133:         }
134:
135:         /* the last element of 'envp[]' must be a NULL-pointer */
136:         *sp = NULL;
137: #ifdef __DEBUG__
138:         printk("at 0x%08x -> ----- = 0x%08x\n", sp, 0);
139: #endif /*__DEBUG__ */
140:         sp++;
141:
142:
143:         /* copy the Auxiliar Table Items (dlinfo_items) */
144:         if(at_base) {
145:             memset_l((void *)sp, AT_PHDR, 1);
146: #ifdef __DEBUG__
147:             printk("at 0x%08x -> AT_PHDR = %d", sp, *sp);
148: #endif /*__DEBUG__ */
149:             sp++;
150:
151:             memcpy_l((void *)sp, &phdr_addr, 1);
152: #ifdef __DEBUG__
153:             printk("\t\tAT_PHDR = 0x%08x\n", *sp);
154: #endif /*__DEBUG__ */
155:             sp++;
156:
157:             memset_l((void *)sp, AT_PHEMT, 1);
158: #ifdef __DEBUG__
159:             printk("at 0x%08x -> AT_PHEMT = %d", sp, *sp);
160: #endif /*__DEBUG__ */
161:             sp++;
162:
163:             memset_l((void *)sp, sizeof(struct elf32_phdr), 1);
164: #ifdef __DEBUG__
165:             printk("\t\tAT_PHEMT = %d\n", *sp);
166: #endif /*__DEBUG__ */
167:             sp++;
168:
169:             memset_l((void *)sp, AT_PHNUM, 1);
170: #ifdef __DEBUG__
171:             printk("at 0x%08x -> AT_PHNUM = %d", sp, *sp);
172: #endif /*__DEBUG__ */
173:             sp++;
174:
175:             memset_l((void *)sp, 0, 1);
176:             memcpy_w((void *)sp, &elf32_h->e_phnum, 1);
177: #ifdef __DEBUG__
178:             printk("\t\tAT_PHNUM = %d\n", *sp);
179: #endif /*__DEBUG__ */
180:             sp++;
181:
182:             memset_l((void *)sp, AT_PAGESZ, 1);
183: #ifdef __DEBUG__
184:             printk("at 0x%08x -> AT_PGFSIZE = %d", sp, *sp);
185: #endif /*__DEBUG__ */
186:             sp++;
187:
188:             memset_l((void *)sp, PAGE_SIZE, 1);
189: #ifdef __DEBUG__
190:             printk("\t\tAT_PGFSIZE = %d\n", *sp);
191: #endif /*__DEBUG__ */
192:             sp++;
193:
194:             memset_l((void *)sp, AT_BASE, 1);
195: #ifdef __DEBUG__
196:             printk("at 0x%08x -> AT_BASE = %d", sp, *sp);
197: #endif /*__DEBUG__ */

```

```

198:             sp++;
199:
200:             memset_l((void *)sp, at_base, 1);
201: #ifdef __DEBUG__
202:             printk("\t\tAT_BASE = 0x%08x\n", sp);
203: #endif /*__DEBUG__ */
204:             sp++;
205:
206:             memset_l((void *)sp, AT_FLAGS, 1);
207: #ifdef __DEBUG__
208:             printk("at 0x%08x -> AT_FLAGS = %d", sp, *sp);
209: #endif /*__DEBUG__ */
210:             sp++;
211:
212:             memset_l((void *)sp, NULL, 1);
213: #ifdef __DEBUG__
214:             printk("\t\tAT_FLAGS = %d\n", *sp);
215: #endif /*__DEBUG__ */
216:             sp++;
217:
218:             memset_l((void *)sp, AT_ENTRY, 1);
219: #ifdef __DEBUG__
220:             printk("at 0x%08x -> AT_ENTRY = %d ", sp, *sp);
221: #endif /*__DEBUG__ */
222:             sp++;
223:
224:             memcpy_l((void *)sp, &elf32_h->e_entry, 1);
225: #ifdef __DEBUG__
226:             printk("\t\tAT_ENTRY = 0x%08x\n", *sp);
227: #endif /*__DEBUG__ */
228:             sp++;
229:
230:             memset_l((void *)sp, AT_UID, 1);
231: #ifdef __DEBUG__
232:             printk("at 0x%08x -> AT_UID = %d", sp, *sp);
233: #endif /*__DEBUG__ */
234:             sp++;
235:
236:             memcpy_l((void *)sp, &current->uid, 1);
237: #ifdef __DEBUG__
238:             printk("\t\tAT_UID = %d\n", *sp);
239: #endif /*__DEBUG__ */
240:             sp++;
241:
242:             memset_l((void *)sp, AT_EUID, 1);
243: #ifdef __DEBUG__
244:             printk("at 0x%08x -> AT_EUID = %d", sp, *sp);
245: #endif /*__DEBUG__ */
246:             sp++;
247:
248:             memcpy_l((void *)sp, &current->euid, 1);
249: #ifdef __DEBUG__
250:             printk("\t\tAT_EUID = %d\n", *sp);
251: #endif /*__DEBUG__ */
252:             sp++;
253:
254:             memset_l((void *)sp, AT_GID, 1);
255: #ifdef __DEBUG__
256:             printk("at 0x%08x -> AT_GID = %d", sp, *sp);
257: #endif /*__DEBUG__ */
258:             sp++;
259:
260:             memcpy_l((void *)sp, &current->gid, 1);
261: #ifdef __DEBUG__
262:             printk("\t\tAT_GID = %d\n", *sp);
263: #endif /*__DEBUG__ */
264:             sp++;

```

```

265:
266:             memset_l((void *)sp, AT_EGID, 1);
267: #ifdef  __DEBUG__
268:             printk("at 0x%08x -> AT_EGID = %d", sp, *sp);
269: #endif /*__DEBUG__ */
270:             sp++;
271:
272:             memcpy_l((void *)sp, &current->egid, 1);
273: #ifdef  __DEBUG__
274:             printk("\t\tAT_EGID = %d\n", *sp);
275: #endif /*__DEBUG__ */
276:             sp++;
277:     }
278:
279:     memset_l((void *)sp, AT_NULL, 1);
280: #ifdef  __DEBUG__
281:     printk("at 0x%08x -> AT_NULL = %d", sp, *sp);
282: #endif /*__DEBUG__ */
283:     sp++;
284:
285:     *sp = NULL;
286: #ifdef  __DEBUG__
287:     printk("\t\tAT_NULL = %d\n", *sp);
288: #endif /*__DEBUG__ */
289:
290: #ifdef  __DEBUG__
291:     for(n = 0; n < barg->argc; n++) {
292:         printk("at 0x%08x -> argv[%d] = '%s'\n", current->argv[n], n, cu
rrent->argv[n]);
293:     }
294:     for(n = 0; n < barg->envc; n++) {
295:         printk("at 0x%08x -> envp[%d] = '%s'\n", current->envp[n], n, cu
rrent->envp[n]);
296:     }
297: #endif /*__DEBUG__ */
298: }
299:
300: static int elf_load_interpreter(struct inode *ii)
301: {
302:     int n, errno;
303:     struct buffer *buf;
304:     struct elf32_hdr *elf32_h;
305:     struct elf32_phdr *elf32_ph, *last_ptload;
306:     __blk_t block;
307:     unsigned int start, end, length;
308:     unsigned int prot;
309:     char *data;
310:     char type;
311:
312:     if((block = bmap(ii, 0, FOR_READING)) < 0) {
313:         return block;
314:     }
315:     if(!(buf = bread(ii->dev, block, ii->sb->s_blocksize))) {
316:         return -EIO;
317:     }
318:
319:     /*
320:      * The contents of the buffer is copied and then freed immediately to
321:      * make sure that it won't conflict while zeroing the BSS fractional
322:      * page, in case that the same block is requested during the page fault.
323:      */
324:     if(!(data = (void *)kmalloc())) {
325:         brelse(buf);
326:         return -ENOMEM;
327:     }
328:     memcpy_b(data, buf->data, ii->sb->s_blocksize);
329:     brelse(buf);

```

```

330:
331:     elf32_h = (struct elf32_hdr *)data;
332:     if(check_elf(elf32_h)) {
333:         kfree((unsigned int)data);
334:         return -ELIBBAD;
335:     }
336:
337:     last_ptload = NULL;
338:     for(n = 0; n < elf32_h->e_phnum; n++) {
339:         elf32_ph = (struct elf32_phdr *) (data + elf32_h->e_phoff + (size
of(struct elf32_phdr) * n));
340:         if(elf32_ph->p_type == PT_LOAD) {
341:             #ifdef __DEBUG__
342:                 printk("p_offset = 0x%08x\n", elf32_ph->p_offset);
343:                 printk("p_vaddr = 0x%08x\n", elf32_ph->p_vaddr);
344:                 printk("p_paddr = 0x%08x\n", elf32_ph->p_paddr);
345:                 printk("p_filesz = 0x%08x\n", elf32_ph->p_filesz);
346:                 printk("p_memsz = 0x%08x\n\n", elf32_ph->p_memsz);
347:             #endif /*__DEBUG__*/
348:             start = (elf32_ph->p_vaddr & PAGE_MASK) + MMAP_START;
349:             length = (elf32_ph->p_vaddr & ~PAGE_MASK) + elf32_ph->p_
filesz;
350:             type = P_DATA;
351:             prot = 0;
352:             if(elf32_ph->p_flags & PF_R) {
353:                 prot = PROT_READ;
354:             }
355:             if(elf32_ph->p_flags & PF_W) {
356:                 prot |= PROT_WRITE;
357:             }
358:             if(elf32_ph->p_flags & PF_X) {
359:                 prot |= PROT_EXEC;
360:                 type = P_TEXT;
361:             }
362:             errno = do_mmap(ii, start, length, prot, MAP_PRIVATE | M
AP_FIXED, elf32_ph->p_offset & PAGE_MASK, type, O_RDONLY);
363:             if(errno < 0 && errno > -PAGE_SIZE) {
364:                 kfree((unsigned int)data);
365:                 send_sig(current, SIGSEGV);
366:                 return -ENOEXEC;
367:             }
368:             last_ptload = elf32_ph;
369:         }
370:     }
371:
372:     if(!last_ptload) {
373:         printk("WARNING: 'last_ptload' is NULL!\n");
374:     }
375:     elf32_ph = last_ptload;
376:
377:     /* zero-fill the fractional page of the DATA section */
378:     end = PAGE_ALIGN(elf32_ph->p_vaddr + elf32_ph->p_filesz) + MMAP_START;
379:     start = (elf32_ph->p_vaddr + elf32_ph->p_filesz) + MMAP_START;
380:     length = end - start;
381:
382:     /* this will generate a page fault which will load the page in */
383:     memset_b((void *)start, NULL, length);
384:
385:     /* setup the BSS section */
386:     start = (elf32_ph->p_vaddr + elf32_ph->p_filesz) + MMAP_START;
387:     start = PAGE_ALIGN(start);
388:     end = (elf32_ph->p_vaddr + elf32_ph->p_memsz) + MMAP_START;
389:     end = PAGE_ALIGN(end);
390:     length = end - start;
391:     errno = do_mmap(NULL, start, length, PROT_READ | PROT_WRITE, MAP_PRIVATE
| MAP_FIXED, 0, P_BSS, 0);
392:     if(errno < 0 && errno > -PAGE_SIZE) {

```



```

393:         kfree((unsigned int) data);
394:         send_sig(current, SIGSEGV);
395:         return -ENOEXEC;
396:     }
397:     kfree((unsigned int) data);
398:     return elf32_h->e_entry + MMAP_START;
399: }
400:
401: int elf_load(struct inode *i, struct binargs *barg, struct sigcontext *sc, char
*data)
402: {
403:     int n, errno;
404:     struct elf32_hdr *elf32_h;
405:     struct elf32_phdr *elf32_ph, *last_ptload;
406:     struct inode *ii;
407:     unsigned int start, end, length;
408:     unsigned int prot;
409:     char *interpreter;
410:     int at_base, phdr_addr;
411:     char type;
412:     unsigned int ae_ptr_len, ae_str_len;
413:     unsigned int sp, str;
414:
415:     elf32_h = (struct elf32_hdr *) data;
416:     if(check_elf(elf32_h)) {
417:         if(current->pid == INIT) {
418:             PANIC("%s has an unrecognized binary format.\n", INIT_PR
OGRAM);
419:         }
420:         return -ENOEXEC;
421:     }
422:
423:     if(elf32_h->e_phnum < 1) {
424:         printk("%s(): no program headers.");
425:         return -ENOEXEC;
426:     }
427:
428:     /* check if an interpreter is required */
429:     interpreter = NULL;
430:     ii = NULL;
431:     phdr_addr = at_base = 0;
432:     for(n = 0; n < elf32_h->e_phnum; n++) {
433:         elf32_ph = (struct elf32_phdr *) (data + elf32_h->e_phoff + (size
of(struct elf32_phdr) * n));
434:         if(elf32_ph->p_type == PT_INTERP) {
435:             at_base = MMAP_START;
436:             interpreter = data + elf32_ph->p_offset;
437:             if(namei(interpreter, &ii, NULL, FOLLOW_LINKS)) {
438:                 printk("%s(): can't find interpreter '%s'.\n", _
_FUNCTION__, interpreter);
439:                 send_sig(current, SIGSEGV);
440:                 return -ELIBACC;
441:             }
442: #ifdef __DEBUG__
443:             printk("p_offset = 0x%08x\n", elf32_ph->p_offset);
444:             printk("p_vaddr = 0x%08x\n", elf32_ph->p_vaddr);
445:             printk("p_paddr = 0x%08x\n", elf32_ph->p_paddr);
446:             printk("p_filesz = 0x%08x\n", elf32_ph->p_filesz);
447:             printk("p_memsz = 0x%08x\n", elf32_ph->p_memsz);
448:             printk("using interpreter '%s'\n", interpreter);
449: #endif /*__DEBUG__*/
450:         }
451:     }
452:
453:     /*
454:      * calculate the final size of 'ae_ptr_len' based on:
455:      * - argc = 4 bytes (unsigned int)

```

fs/elf.c

Page 8/10

```

456:         * - barg.argc = (num. of pointers to strings + 1 NULL) x 4 bytes (unsi
gned int)
457:         * - barg.envc = (num. of pointers to strings + 1 NULL) x 4 bytes (unsi
gned int)
458:         */
459:         ae_ptr_len = (1 + (barg->argc + 1) + (barg->envc + 1)) * sizeof(unsigned
int));
460:         ae_str_len = barg->argv_len + barg->envp_len;
461:         if(ae_ptr_len + ae_str_len > (ARG_MAX * PAGE_SIZE)) {
462:             printk("WARNING: %s(): argument list (%d) exceeds ARG_MAX (%d)!\n
n", __FUNCTION__, ae_ptr_len + ae_str_len, ARG_MAX * PAGE_SIZE);
463:             return -E2BIG;
464:         }
465:
466: #ifdef __DEBUG__
467:     printk("argc=%d (argv_len=%d) envc=%d (envp_len=%d) ae_ptr_len=%d ae_st
r_len=%d\n", barg->argc, barg->argv_len, barg->envc, barg->envp_len, ae_ptr_len, ae_str
_len);
468: #endif /*__DEBUG__ */
469:
470:
471:     /* point of no return */
472:
473:     release_binary();
474:     current->rss = 0;
475:
476:     current->entry_address = elf32_h->e_entry;
477:     if(interpreter) {
478:         errno = elf_load_interpreter(ii);
479:         if(errno < 0) {
480:             printk("%s(): unable to load the interpreter '%s'.\n", _
__FUNCTION__, interpreter);
481:             iput(ii);
482:             send_sig(current, SIGKILL);
483:             return errno;
484:         }
485:         current->entry_address = errno;
486:         iput(ii);
487:     }
488:
489:     last_ptload = NULL;
490:     for(n = 0; n < elf32_h->e_phnum; n++) {
491:         elf32_ph = (struct elf32_phdr *) (data + elf32_h->e_phoff + (size
of(struct elf32_phdr) * n));
492:         if(elf32_ph->p_type == PT_PHDR) {
493:             phdr_addr = elf32_ph->p_vaddr;
494:         }
495:         if(elf32_ph->p_type == PT_LOAD) {
496:             start = elf32_ph->p_vaddr & PAGE_MASK;
497:             length = (elf32_ph->p_vaddr & ~PAGE_MASK) + elf32_ph->p_
filesz;
498:             type = P_DATA;
499:             prot = 0;
500:             if(elf32_ph->p_flags & PF_R) {
501:                 prot = PROT_READ;
502:             }
503:             if(elf32_ph->p_flags & PF_W) {
504:                 prot |= PROT_WRITE;
505:             }
506:             if(elf32_ph->p_flags & PF_X) {
507:                 prot |= PROT_EXEC;
508:                 type = P_TEXT;
509:             }
510:             errno = do_mmap(i, start, length, prot, MAP_PRIVATE | MA
P_FIXED, elf32_ph->p_offset & PAGE_MASK, type, O_RDONLY);
511:             if(errno < 0 && errno > -PAGE_SIZE) {
512:                 send_sig(current, SIGSEGV);

```

fs/elf.c

Page 9/10

```

513:                 return -ENOEXEC;
514:             }
515:             last_ptload = elf32_ph;
516:         }
517:     }
518:
519:     elf32_ph = last_ptload;
520:
521:     /* zero-fill the fractional page of the DATA section */
522:     end = PAGE_ALIGN(elf32_ph->p_vaddr + elf32_ph->p_filesz);
523:     start = elf32_ph->p_vaddr + elf32_ph->p_filesz;
524:     length = end - start;
525:
526:     /* this will generate a page fault which will load the page in */
527:     memset_b((void *)start, NULL, length);
528:
529:     /* setup the BSS section */
530:     start = elf32_ph->p_vaddr + elf32_ph->p_filesz;
531:     start = PAGE_ALIGN(start);
532:     end = elf32_ph->p_vaddr + elf32_ph->p_memsz;
533:     end = PAGE_ALIGN(end);
534:     length = end - start;
535:     errno = do_mmap(NULL, start, length, PROT_READ | PROT_WRITE, MAP_PRIVATE
| MAP_FIXED, 0, P_BSS, 0);
536:     if(errno < 0 && errno > -PAGE_SIZE) {
537:         send_sig(current, SIGSEGV);
538:         return -ENOEXEC;
539:     }
540:     current->brk_lower = start;
541:
542:     /* setup the HEAP section */
543:     start = elf32_ph->p_vaddr + elf32_ph->p_memsz;
544:     start = PAGE_ALIGN(start);
545:     length = PAGE_SIZE;
546:     errno = do_mmap(NULL, start, length, PROT_READ | PROT_WRITE, MAP_PRIVATE
| MAP_FIXED, 0, P_HEAP, 0);
547:     if(errno < 0 && errno > -PAGE_SIZE) {
548:         send_sig(current, SIGSEGV);
549:         return -ENOEXEC;
550:     }
551:     current->brk = start;
552:
553:     /* setup the STACK section */
554:     sp = KERNEL_BASE_ADDR - 4;      /* formerly 0xBFFFFFFC */
555:     sp -= ae_str_len;
556:     str = sp;      /* this is the address of the first string (argv[0]) */
557:     sp &= ~3;
558:     sp -= at_base ? (AT_ITEMS * 2) * sizeof(unsigned int) : 2 * sizeof(unsig
ned int);
559:     sp -= ae_ptr_len;
560:     length = KERNEL_BASE_ADDR - (sp & PAGE_MASK);
561:     errno = do_mmap(NULL, sp & PAGE_MASK, length, PROT_READ | PROT_WRITE | P
ROT_EXEC, MAP_PRIVATE | MAP_FIXED, 0, P_STACK, 0);
562:     if(errno < 0 && errno > -PAGE_SIZE) {
563:         send_sig(current, SIGSEGV);
564:         return -ENOEXEC;
565:     }
566:
567:     elf_create_stack(barg, (unsigned int *)sp, str, at_base, elf32_h, phdr_a
ddr);
568:
569:     /* set %esp to point to 'argc' */
570:     sc->oldesp = sp;
571:     sc->eflags = 0x202;      /* FIXME: linux 2.2 = 0x292 */
572:     sc->eip = current->entry_address;
573:     sc->err = 0;
574:     sc->eax = 0;

```

fs/elf.c

Page 10/10

```
575:         sc->ecx = 0;
576:         sc->edx = 0;
577:         sc->ebx = 0;
578:         sc->ebp = 0;
579:         sc->esi = 0;
580:         sc->edi = 0;
581:         return 0;
582: }
```

fs/fd.c

Page 1/1

```
1: /*
2:  * fiwix/fs/fd.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/errno.h>
9: #include <fiwix/types.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/sleep.h>
12: #include <fiwix/stdio.h>
13: #include <fiwix/string.h>
14:
15: struct fd *fd_table;
16:
17: static struct resource fd_resource = { NULL, NULL };
18:
19: int get_new_fd(struct inode *i)
20: {
21:     unsigned int n;
22:
23:     lock_resource(&fd_resource);
24:
25:     for(n = 1; n < NR_OPENS; n++) {
26:         if(fd_table[n].count == 0) {
27:             memset_b(&fd_table[n], NULL, sizeof(struct fd));
28:             fd_table[n].inode = i;
29:             fd_table[n].count = 1;
30:             unlock_resource(&fd_resource);
31:             return n;
32:         }
33:     }
34:
35:     unlock_resource(&fd_resource);
36:
37:     return -ENFILE;
38: }
39:
40: void release_fd(unsigned int fd)
41: {
42:     lock_resource(&fd_resource);
43:     fd_table[fd].count = 0;
44:     unlock_resource(&fd_resource);
45: }
46:
47: void fd_init(void)
48: {
49:     memset_b(fd_table, NULL, fd_table_size);
50: }
```

fs/filesystems.c

Page 1/2

```

1: /*
2:  * fiwix/fs/filesystems.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/errno.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/filesystems.h>
12: #include <fiwix/fs_pipe.h>
13: #include <fiwix/fs_iso9660.h>
14: #include <fiwix/fs_proc.h>
15: #include <fiwix/stdio.h>
16: #include <fiwix/string.h>
17:
18: int register_filesystem(const char *name, struct fs_operations *fsop)
19: {
20:     int n;
21:     __dev_t dev;
22:
23:     for(n = 0; n < NR_FILESYSTEMS; n++) {
24:         if(filesystems_table[n].name) {
25:             if(strcmp(filesystems_table[n].name, name) == 0) {
26:                 printk("WARNING: %s(): filesystem '%s' already r
registered!\n", __FUNCTION__, name);
27:                 return 1;
28:             }
29:         }
30:         if(!filesystems_table[n].name) {
31:             filesystems_table[n].name = name;
32:             filesystems_table[n].fsop = fsop;
33:             if((fsop->flags & FSOP_KERN_MOUNT)) {
34:                 dev = fsop->fsdev;
35:                 return kern_mount(dev, &filesystems_table[n]);
36:             }
37:             return 0;
38:         }
39:     }
40:     printk("WARNING: %s(): filesystems table is full!\n", __FUNCTION__);
41:     return 1;
42: }
43:
44: struct filesystems * get_filesystem(const char *name)
45: {
46:     int n;
47:
48:     if(!name) {
49:         return NULL;
50:     }
51:     for(n = 0; n < NR_FILESYSTEMS; n++) {
52:         if(!filesystems_table[n].name) {
53:             continue;
54:         }
55:         if(strcmp(filesystems_table[n].name, name) == 0) {
56:             return &filesystems_table[n];
57:         }
58:     }
59:     return NULL;
60: }
61:
62: void fs_init(void)
63: {
64:     memset_b(filesystems_table, NULL, sizeof(filesystems_table));
65:
66:     if(minix_init()) {

```

fs/filesystems.c

Page 2/2

```
67:                printk("%s(): unable to register 'minix' filesystem.\n", __FUNCT
ION__);
68:                }
69:                if(ext2_init()) {
70:                    printk("%s(): unable to register 'ext2' filesystem.\n", __FUNCTI
ON__);
71:                }
72:                if(pipefs_init()) {
73:                    printk("%s(): unable to register 'pipefs' filesystem.\n", __FUNC
TION__);
74:                }
75:                if(iso9660_init()) {
76:                    printk("%s(): unable to register 'iso9660' filesystem.\n", __FUN
CTION__);
77:                }
78:                if(procfs_init()) {
79:                    printk("%s(): unable to register 'procfs' filesystem.\n", __FUNC
TION__);
80:                }
81: }
```

fs/inode.c

```

1: /*
2:  * fiwix/fs/inode.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: /*
9:  * inode.c implements a cache with a free list as a doubly circular linked
10:  * list and a chained hash table with doubly linked lists.
11:  *
12:  * hash table
13:  * +-----+ +-----+ +-----+ +-----+
14:  * | index | |prev|data|next| |prev|data|next| |prev|data|next|
15:  * |   0   --> | / |   |   |   | <--- |   |   |   | <--- |   |   |   |
16:  * +-----+ +-----+ +-----+ +-----+
17:  * +-----+ +-----+ +-----+ +-----+
18:  * | index | |prev|data|next| |prev|data|next| |prev|data|next|
19:  * |   1   --> | / |   |   |   | <--- |   |   |   | <--- |   |   |   |
20:  * +-----+ +-----+ +-----+ +-----+
21:  *                               (inode)       (inode)       (inode)
22:  *      ...
23:  */
24:
25: #include <fiwix/asm.h>
26: #include <fiwix/kernel.h>
27: #include <fiwix/sleep.h>
28: #include <fiwix/sched.h>
29: #include <fiwix/fs.h>
30: #include <fiwix/filesystems.h>
31: #include <fiwix/stat.h>
32: #include <fiwix/errno.h>
33: #include <fiwix/mm.h>
34: #include <fiwix/stdio.h>
35: #include <fiwix/string.h>
36:
37: #define INODE_HASH(dev, inode) (((__dev_t)(dev) ^ (__ino_t)(inode)) % (NR_INO_H
ASH))
38: #define NR_INODES      (inode_table_size / sizeof(struct inode))
39: #define NR_INO_HASH    (inode_hash_table_size / sizeof(unsigned int))
40:
41: struct inode *inode_table;          /* inode pool */
42: struct inode *inode_head;          /* inode pool head */
43: struct inode **inode_hash_table;
44:
45: static struct resource sync_resource = { NULL, NULL };
46:
47: static void insert_to_hash(struct inode *i)
48: {
49:     struct inode **h;
50:     int n;
51:
52:     n = INODE_HASH(i->dev, i->inode);
53:     h = &inode_hash_table[n];
54:
55:     if(!*h) {
56:         *h = i;
57:         (*h)->prev_hash = (*h)->next_hash = NULL;
58:     } else {
59:         i->prev_hash = NULL;
60:         i->next_hash = *h;
61:         (*h)->prev_hash = i;
62:         *h = i;
63:     }
64: }
65:
66: static void remove_from_hash(struct inode *i)

```



```

67: {
68:     struct inode **h;
69:     int n;
70:
71:     if(!i->inode) {
72:         return;
73:     }
74:
75:     n = INODE_HASH(i->dev, i->inode);
76:     h = &inode_hash_table[n];
77:
78:     while(*h) {
79:         if(*h == i) {
80:             if((*h)->next_hash) {
81:                 (*h)->next_hash->prev_hash = (*h)->prev_hash;
82:             }
83:             if((*h)->prev_hash) {
84:                 (*h)->prev_hash->next_hash = (*h)->next_hash;
85:             }
86:             if(h == &inode_hash_table[n]) {
87:                 *h = (*h)->next_hash;
88:             }
89:             break;
90:         }
91:         h = &(*h)->next_hash;
92:     }
93: }
94:
95: static void insert_on_free_list(struct inode *i)
96: {
97:     if(!inode_head) {
98:         i->prev_free = i->next_free = i;
99:         inode_head = i;
100:    } else {
101:        i->next_free = inode_head;
102:        i->prev_free = inode_head->prev_free;
103:        inode_head->prev_free->next_free = i;
104:        inode_head->prev_free = i;
105:    }
106:
107:    kstat.free_inodes++;
108: }
109:
110: static void remove_from_free_list(struct inode *i)
111: {
112:     if(!kstat.free_inodes) {
113:         return;
114:     }
115:
116:     i->prev_free->next_free = i->next_free;
117:     i->next_free->prev_free = i->prev_free;
118:     kstat.free_inodes--;
119:     if(i == inode_head) {
120:         inode_head = i->next_free;
121:     }
122:
123:     if(!kstat.free_inodes) {
124:         inode_head = NULL;
125:     }
126: }
127:
128: static struct inode * get_free_inode(void)
129: {
130:     unsigned long int flags;
131:     struct inode *i;
132:
133:     /* no more inodes on free list */

```

```

134:         if(!kstat.free_inodes) {
135:             return NULL;
136:         }
137:
138:     SAVE_FLAGS(flags); CLI();
139:
140:     i = inode_head;
141:     remove_from_free_list(i);
142:     remove_from_hash(i);
143:     i->i_mode = 0;
144:     i->i_uid = 0;
145:     i->i_size = 0;
146:     i->i_atime = 0;
147:     i->i_ctime = 0;
148:     i->i_mtime = 0;
149:     i->i_gid = 0;
150:     i->i_nlink = 0;
151:     i->i_blocks = 0;
152:     i->i_flags = 0;
153:     i->locked = 0;
154:     i->dirty = 0;
155:     i->mount_point = NULL;
156:     i->dev = 0;
157:     i->inode = 0;
158:     i->count = 0;
159:     i->rdev = 0;
160:     i->fsop = NULL;
161:     i->sb = NULL;
162:     memset_b(&i->u, NULL, sizeof(i->u));
163:
164:     RESTORE_FLAGS(flags);
165:     return i;
166: }
167:
168: static int read_inode(struct inode *i)
169: {
170:     int errno;
171:
172:     inode_lock(i);
173:     if(i->sb && i->sb->fsop && i->sb->fsop->read_inode) {
174:         errno = i->sb->fsop->read_inode(i);
175:         inode_unlock(i);
176:         return errno;
177:     }
178:     inode_unlock(i);
179:     return -EINVAL;
180: }
181:
182: static int write_inode(struct inode *i)
183: {
184:     int errno;
185:
186:     inode_lock(i);
187:     if(i->sb && i->sb->fsop && i->sb->fsop->write_inode) {
188:         errno = i->sb->fsop->write_inode(i);
189:     } else {
190:         /* PIPE_DEV inodes can't be flushed on disk */
191:         i->dirty = 0;
192:         errno = 0;
193:     }
194:     inode_unlock(i);
195:
196:     return errno;
197: }
198:
199: static struct inode * search_inode_hash(__dev_t dev, __ino_t inode)
200: {

```

fs/inode.c

Page 4/7

```

201:     struct inode *i;
202:     int n;
203:
204:     n = INODE_HASH(dev, inode);
205:     i = inode_hash_table[n];
206:
207:     while(i) {
208:         if(i->dev == dev && i->inode == inode) {
209:             return i;
210:         }
211:         i = i->next_hash;
212:     }
213:
214:     return NULL;
215: }
216:
217: void inode_lock(struct inode *i)
218: {
219:     unsigned long int flags;
220:
221:     for(;;) {
222:         SAVE_FLAGS(flags); CLI();
223:         if(i->locked) {
224:             RESTORE_FLAGS(flags);
225:             sleep(i, PROC_UNINTERRUPTIBLE);
226:         } else {
227:             break;
228:         }
229:     }
230:     i->locked = 1;
231:     RESTORE_FLAGS(flags);
232: }
233:
234: void inode_unlock(struct inode *i)
235: {
236:     unsigned long int flags;
237:
238:     SAVE_FLAGS(flags); CLI();
239:     i->locked = 0;
240:     wakeup(i);
241:     RESTORE_FLAGS(flags);
242: }
243:
244: struct inode * ialloc(struct superblock *sb, int mode)
245: {
246:     int errno;
247:     struct inode *i;
248:
249:     if((i = get_free_inode())) {
250:         i->sb = sb;
251:         i->rdev = sb->dev;
252:         if(i->sb && i->sb->fsop && i->sb->fsop->ialloc) {
253:             errno = i->sb->fsop->ialloc(i, mode);
254:         } else {
255:             printk("WARNING: this filesystem does not have the iallo
c() method!\n");
256:             i->count = 1;
257:             i->sb = NULL;
258:             iput(i);
259:             return NULL;
260:         }
261:         if(errno) {
262:             i->count = 1;
263:             i->sb = NULL;
264:             iput(i);
265:             return NULL;
266:         }

```

fs/inode.c

Page 5/7

```

267:         i->dev = sb->dev;
268:         insert_to_hash(i);
269:         return i;
270:     }
271:     printk("WARNING: %s(): no more inodes on free list!\n", __FUNCTION__);
272:     return NULL;
273: }
274:
275: struct inode * iget(struct superblock *sb, __ino_t inode)
276: {
277:     unsigned long int flags;
278:     struct inode *i;
279:
280:     if(!inode) {
281:         return NULL;
282:     }
283:
284:     for(;;) {
285:         if((i = search_inode_hash(sb->dev, inode)) {
286:             inode_lock(i);
287:
288:             if(i->mount_point) {
289:                 inode_unlock(i);
290:                 i = i->mount_point;
291:                 inode_lock(i);
292:             }
293:             if(!i->count) {
294:                 remove_from_free_list(i);
295:             }
296:             i->count++;
297:             inode_unlock(i);
298:             return i;
299:         }
300:
301:         if(!(i = get_free_inode())) {
302:             printk("WARNING: %s(): no more inodes on free list! (%d)
.\n", __FUNCTION__, kstat.free_inodes);
303:             return NULL;
304:         }
305:
306:         SAVE_FLAGS(flags); CLI();
307:         i->dev = i->rdev = sb->dev;
308:         i->inode = inode;
309:         i->sb = sb;
310:         i->count = 1;
311:         RESTORE_FLAGS(flags);
312:         if(read_inode(i)) {
313:             iput(i);
314:             return NULL;
315:         }
316:         insert_to_hash(i);
317:         return i;
318:     }
319: }
320:
321: int bmap(struct inode *i, __off_t offset, int mode)
322: {
323:     if(i->fsop && i->fsop->bmap) {
324:         return i->fsop->bmap(i, offset, mode);
325:     }
326:     return -EPERM;
327: }
328:
329: int check_fs_busy(__dev_t dev, struct inode *root)
330: {
331:     struct inode *i;
332:     unsigned int n;

```

fs/inode.c

Page 6/7

```

333:
334:     i = &inode_table[0];
335:     for(n = 0; n < NR_INODES; n++, i = &inode_table[n]) {
336:         if(i->dev == dev && i->count) {
337:             if(i == root && i->count == 1) {
338:                 continue;
339:             }
340:             /* FIXME: to be removed */
341:             printk("WARNING: root %d with count %d (on dev %d,%d)\n"
, root->inode, root->count, MAJOR(i->dev), MINOR(i->dev));
342:             printk("WARNING: inode %d with count %d (on dev %d,%d)\n
", i->inode, i->count, MAJOR(i->dev), MINOR(i->dev));
343:             return 1;
344:         }
345:     }
346:     return 0;
347: }
348:
349: void iput(struct inode *i)
350: {
351:     unsigned long int flags;
352:
353:     /* this solves the problem with rmdir("/") and iput(dir) which is NULL */
/
354:     if(!i) {
355:         return;
356:     }
357:
358:     if(!i->count) {
359:         printk("WARNING: %s(): trying to free an already freed inode (%d
)! \n", __FUNCTION__, i->inode);
360:         return;
361:     }
362:
363:     if(--i->count > 0) {
364:         return;
365:     }
366:
367:     if(!i->i_nlink) {
368:         if(i->sb && i->sb->fsop && i->sb->fsop->ifree) {
369:             inode_lock(i);
370:             i->sb->fsop->ifree(i);
371:             inode_unlock(i);
372:         }
373:         remove_from_hash(i);
374:     }
375:     if(i->dirty) {
376:         if(write_inode(i)) {
377:             printk("WARNING: %s(): can't write inode %d (%d,%d), wil
l remain as dirty.\n", __FUNCTION__, i->inode, MAJOR(i->dev), MINOR(i->dev));
378:             return;
379:         }
380:     }
381:
382:     SAVE_FLAGS(flags); CLI();
383:     insert_on_free_list(i);
384:     RESTORE_FLAGS(flags);
385: }
386:
387: void sync_inodes(__dev_t dev)
388: {
389:     struct inode *i;
390:     int n;
391:
392:     i = &inode_table[0];
393:
394:     lock_resource(&sync_resource);

```

fs/inode.c

Page 7/7

```
395:         for(n = 0; n < NR_INODES; n++) {
396:             if(i->dirty) {
397:                 if(!dev || i->dev == dev) {
398:                     if(write_inode(i)) {
399:                         printk("WARNING: %s(): can't write inode
%d (%d,%d), will remain as dirty.\n", __FUNCTION__, i->inode, MAJOR(i->dev), MINOR(i->
dev));
400:                     }
401:                 }
402:             }
403:             i++;
404:         }
405:         unlock_resource(&sync_resource);
406:     }
407:
408: void invalidate_inodes(__dev_t dev)
409: {
410:     unsigned long int flags;
411:     unsigned int n;
412:     struct inode *i;
413:
414:     i = &inode_table[0];
415:     SAVE_FLAGS(flags); CLI();
416:
417:     for(n = 0; n < NR_INODES; n++) {
418:         if(i->dev == dev) {
419:             inode_lock(i);
420:             remove_from_hash(i);
421:             inode_unlock(i);
422:         }
423:         i++;
424:     }
425:
426:     RESTORE_FLAGS(flags);
427: }
428:
429: void inode_init(void)
430: {
431:     struct inode *i;
432:     unsigned int n;
433:
434:     memset_b(inode_table, NULL, inode_table_size);
435:     memset_b(inode_hash_table, NULL, inode_hash_table_size);
436:     for(n = 0; n < NR_INODES; n++) {
437:         i = &inode_table[n];
438:         i->count = 1;
439:         insert_on_free_list(i);
440:     }
441: }
```

fs/locks.c

Page 1/4

```

1: /*
2:  * fiwix/fs/locks.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/errno.h>
9: #include <fiwix/types.h>
10: #include <fiwix/locks.h>
11: #include <fiwix/fs.h>
12: #include <fiwix/sleep.h>
13: #include <fiwix/sched.h>
14: #include <fiwix/stdio.h>
15: #include <fiwix/string.h>
16:
17: static struct resource flock_resource = { NULL, NULL };
18:
19: static struct flock_file * get_new_flock(struct inode *i)
20: {
21:     int n;
22:     struct flock_file *ff;
23:
24:     lock_resource(&flock_resource);
25:
26:     for(n = 0; n < NR_FLOCKS; n++) {
27:         ff = &flock_file_table[n];
28:         if(!ff->inode) {
29:             ff->inode = i; /* mark it as busy */
30:             unlock_resource(&flock_resource);
31:             return ff;
32:         }
33:     }
34:
35:     printk("WARNING: %s(): no more free slots in flock file table.\n");
36:     unlock_resource(&flock_resource);
37:     return NULL;
38: }
39:
40: static void release_flock(struct flock_file *ff)
41: {
42:     memset_b(ff, 0, sizeof(struct flock_file));
43: }
44:
45: static struct flock_file * get_flock_file(struct inode *i, int op, struct proc *
p)
46: {
47:     int n;
48:     struct flock_file *ff;
49:
50:     lock_resource(&flock_resource);
51:
52:     ff = NULL;
53:     for(n = 0; n < NR_FLOCKS; n++) {
54:         ff = &flock_file_table[n];
55:         if(ff->inode != i) {
56:             continue;
57:         }
58:         if(p && p != ff->proc) {
59:             continue;
60:         }
61:         break;
62:     }
63:     unlock_resource(&flock_resource);
64:     return ff;
65: }
66:

```

fs/locks.c

Page 2/4

```

67: int posix_lock(int ufd, int cmd, struct flock *fl)
68: {
69:     int n;
70:     struct flock_file *ff;
71:     struct inode *i;
72:     unsigned char type;
73:
74:     lock_resource(&flock_resource);
75:     i = fd_table[current->fd[ufd]].inode;
76:     for(n = 0; n < NR_FLOCKS; n++) {
77:         ff = &flock_file_table[n];
78:         if(ff->inode != i) {
79:             continue;
80:         }
81:         break;
82:     }
83:     unlock_resource(&flock_resource);
84:     if(cmd == F_GETLK) {
85:         if(ff->inode == i) {
86:             fl->l_type = (ff->type & LOCK_SH) ? F_RDLCK : F_WRLCK;
87:             fl->l_whence = SEEK_SET;
88:             fl->l_start = 0;
89:             fl->l_len = 0;
90:             fl->l_pid = ff->proc->pid;
91:         } else {
92:             fl->l_type = F_UNLCK;
93:         }
94:     }
95:
96:     switch(fl->l_type) {
97:         case F_RDLCK:
98:             type = LOCK_SH;
99:             break;
100:        case F_WRLCK:
101:            type = LOCK_EX;
102:            break;
103:        case F_UNLCK:
104:            type = LOCK_UN;
105:            break;
106:        default:
107:            return -EINVAL;
108:    }
109:    if(cmd == F_SETLK) {
110:        return flock_inode(i, type);
111:    }
112:    if(cmd == F_SETLKW) {
113:        return flock_inode(i, type | LOCK_NB);
114:    }
115:    return 0;
116: }
117:
118: void flock_release_inode(struct inode *i)
119: {
120:     int n;
121:     struct flock_file *ff;
122:
123:     lock_resource(&flock_resource);
124:     for(n = 0; n < NR_FLOCKS; n++) {
125:         ff = &flock_file_table[n];
126:         if(ff->inode != i) {
127:             continue;
128:         }
129:         if(ff->proc != current) {
130:             continue;
131:         }
132:         wakeup(ff);
133:         release_flock(ff);

```


fs/locks.c

Page 3/4

```

134:         }
135:         unlock_resource(&flock_resource);
136:     }
137:
138: int flock_inode(struct inode *i, int op)
139: {
140:     int n;
141:     struct flock_file *ff, *new;
142:
143:     if(op & LOCK_UN) {
144:         if((ff = get_flock_file(i, op, current))) {
145:             wakeup(ff);
146:             release_flock(ff);
147:         }
148:         return 0;
149:     }
150:
151: loop:
152:     lock_resource(&flock_resource);
153:     new = NULL;
154:     for(n = 0; n < NR_FLOCKS; n++) {
155:         ff = &flock_file_table[n];
156:         if(ff->inode != i) {
157:             continue;
158:         }
159:         if(op & LOCK_SH) {
160:             if(ff->type & LOCK_EX) {
161:                 if(ff->proc == current) {
162:                     new = ff;
163:                     wakeup(ff);
164:                     break;
165:                 }
166:                 unlock_resource(&flock_resource);
167:                 if(op & LOCK_NB) {
168:                     return -EWOULDBLOCK;
169:                 }
170:                 if(sleep(ff, PROC_INTERRUPTIBLE)) {
171:                     return -EINTR;
172:                 }
173:                 goto loop;
174:             }
175:         }
176:         if(op & LOCK_EX) {
177:             if(ff->proc == current) {
178:                 new = ff;
179:                 continue;
180:             }
181:             unlock_resource(&flock_resource);
182:             if(op & LOCK_NB) {
183:                 return -EWOULDBLOCK;
184:             }
185:             if(sleep(ff, PROC_INTERRUPTIBLE)) {
186:                 return -EINTR;
187:             }
188:             goto loop;
189:         }
190:     }
191:     unlock_resource(&flock_resource);
192:
193:     if(!new) {
194:         if(!(new = get_new_flock(i))) {
195:             return -ENOLCK;
196:         }
197:     }
198:     new->inode = i;
199:     new->type = op;
200:     new->proc = current;

```

fs/locks.c

Page 4/4

```
201:
202:     return 0;
203: }
204:
205: void flock_init(void)
206: {
207:     memset_b(flock_file_table, NULL, sizeof(flock_file_table));
208: }
```

fs/Makefile

Page 1/1

```
1: # fiwix/fs/Makefile
2: #
3: # Copyright 2018, Jordi Sanfeliu. All rights reserved.
4: # Distributed under the terms of the Fiwix License.
5: #
6:
7: .S.o:
8:     $(CC) -traditional -I$(INCLUDE) -c -o $@ $<
9: .c.o:
10:     $(CC) $(CFLAGS) -c -o $@ $<
11:
12: FSDIRS = minix ext2 pipefs iso9660 procfs
13: FILESYSTEMS = minix/minix.o ext2/ext2.o pipefs/pipefs.o iso9660/iso9660.o \
14:     procfs/procfs.o
15: OBJS = filesystems.o devices.o buffer.o fd.o locks.o super.o inode.o \
16:     namei.o elf.o script.o
17:
18: fs:     $(OBJS)
19:     @for n in $(FSDIRS) ; do (cd $$n ; $(MAKE)) ; done
20:     $(LD) $(LDFLAGS) -r $(FILESYSTEMS) $(OBJS) -o fs.o
21:
22: clean:
23:     @for n in $(FSDIRS) ; do (cd $$n ; $(MAKE) clean) ; done
24:     rm -f *.o
25:
```

fs/namei.c

Page 1/3

```

1: /*
2:  * fiwix/fs/namei.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/sleep.h>
10: #include <fiwix/sched.h>
11: #include <fiwix/fs.h>
12: #include <fiwix/filesystems.h>
13: #include <fiwix/stat.h>
14: #include <fiwix/mm.h>
15: #include <fiwix/mman.h>
16: #include <fiwix/errno.h>
17: #include <fiwix/stdio.h>
18: #include <fiwix/string.h>
19:
20: static int namei_lookup(char *name, struct inode *dir, struct inode **i_res)
21: {
22:     if(dir->fsop && dir->fsop->lookup) {
23:         return dir->fsop->lookup(name, dir, i_res);
24:     }
25:     return -EACCES;
26: }
27:
28: static int do_namei(char *path, struct inode *dir, struct inode **i_res, struct
inode **d_res, int follow_links)
29: {
30:     char *name, *ptr_name;
31:     struct inode *i;
32:     struct superblock *sb;
33:
34:     int errno;
35:
36:     *i_res = dir;
37:     for(;;) {
38:         while(*path == '/') {
39:             path++;
40:         }
41:         if(*path == NULL) {
42:             return 0;
43:         }
44:
45:         /* extracts the next component of the path */
46:         if(! (name = (char *)kmalloc())) {
47:             return -ENOMEM;
48:         }
49:         ptr_name = name;
50:         while(*path != NULL && *path != '/') {
51:             if(ptr_name > name + NAME_MAX - 1) {
52:                 break;
53:             }
54:             *ptr_name++ = *path++;
55:         }
56:         *ptr_name = NULL;
57:
58:         /*
59:          * If the inode is the root of a file system, then return the
60:          * inode on which the file system was mounted.
61:          */
62:         if(name[0] == '.' && name[1] == '.' && name[2] == NULL) {
63:             if(dir == dir->sb->root) {
64:                 sb = dir->sb;
65:                 iput(dir);
66:                 dir = sb->dir;

```

```

67:                                dir->count++;
68:                                }
69:                                }
70:
71:                                if((errno = check_permission(TO_EXEC, dir)) {
72:                                    break;
73:                                }
74:
75:                                dir->count++;
76:                                if((errno = namei_lookup(name, dir, &i)) {
77:                                    break;
78:                                }
79:
80:                                kfree((unsigned int) name);
81:                                if(*path == '/') {
82:                                    if(!S_ISDIR(i->i_mode) && !S_ISLNK(i->i_mode)) {
83:                                        iput(dir);
84:                                        iput(i);
85:                                        return -ENOTDIR;
86:                                    }
87:                                    if(S_ISLNK(i->i_mode)) {
88:                                        if(i->fsop && i->fsop->followlink) {
89:                                            if((errno = i->fsop->followlink(dir, i,
&i)) {
90:                                                iput(dir);
91:                                                return errno;
92:                                            }
93:                                        }
94:                                    }
95:                                } else {
96:                                    if((i->fsop && i->fsop->followlink) && follow_links) {
97:                                        if((errno = i->fsop->followlink(dir, i, &i)) {
98:                                            iput(dir);
99:                                            return errno;
100:                                        }
101:                                    }
102:                                }
103:
104:                                if(d_res) {
105:                                    if(*d_res) {
106:                                        iput(*d_res);
107:                                    }
108:                                    *d_res = dir;
109:                                } else {
110:                                    iput(dir);
111:                                }
112:                                dir = i;
113:                                *i_res = i;
114:                            }
115:
116:                                kfree((unsigned int) name);
117:                                if(d_res) {
118:                                    if(*d_res) {
119:                                        iput(*d_res);
120:                                    }
121:                                /*
122:                                * If that was the last component of the path,
123:                                * then return the directory.
124:                                */
125:                                if(*path == NULL) {
126:                                    *d_res = dir;
127:                                    dir->count++;
128:                                } else {
129:                                    /* that's an non-existent directory */
130:                                    *d_res = NULL;
131:                                    errno = -ENOTDIR;
132:                                }

```

fs/namei.c

Page 3/3

```
133:         iput(dir);
134:         *i_res = NULL;
135:     } else {
136:         iput(dir);
137:     }
138:
139:     return errno;
140: }
141:
142: int parse_namei(char *path, struct inode *base_dir, struct inode **i_res, struct
inode **d_res, int follow_links)
143: {
144:     struct inode *dir;
145:     int errno;
146:
147:     if(!path) {
148:         return -EFAULT;
149:     }
150:     if(*path == NULL) {
151:         return -ENOENT;
152:     }
153:
154:     if(!(dir = base_dir)) {
155:         dir = current->pwd;
156:     }
157:
158:     /* it is definitely an absolute path */
159:     if(path[0] == '/') {
160:         dir = current->root;
161:     }
162:     dir->count++;
163:     errno = do_namei(path, dir, i_res, d_res, follow_links);
164:     return errno;
165: }
166:
167: /*
168:  * namei() returns:
169:  * i_res -> the inode of the last component of the path, or NULL.
170:  * d_res -> the inode of the directory where i_res resides, or NULL.
171:  */
172: int namei(char *path, struct inode **i_res, struct inode **d_res, int follow_lin
ks)
173: {
174:     *i_res = NULL;
175:     if(d_res) {
176:         *d_res = NULL;
177:     }
178:     return parse_namei(path, NULL, i_res, d_res, follow_links);
179: }
```

fs/script.c

Page 1/2

```

1:  /*
2:  *  fiwix/fs/script.c
3:  *
4:  *  Copyright 2019, Jordi Sanfeliu. All rights reserved.
5:  *  Distributed under the terms of the Fiwix License.
6:  */
7:
8:  #include <fiwix/limits.h>
9:  #include <fiwix/process.h>
10: #include <fiwix/errno.h>
11: #include <fiwix/stdio.h>
12: #include <fiwix/string.h>
13:
14: int script_load(char *interpreter, char *args, char *data)
15: {
16:     char *p;
17:     int n, noargs;
18:
19:     /* has shebang? */
20:     if(data[0] != '#' && data[1] != '!') {
21:         return -ENOEXEC;
22:     }
23:
24:     /* discard possible blanks before the interpreter name */
25:     for(n = 2; n < NAME_MAX; n++) {
26:         if(data[n] != ' ' && data[n] != '\t') {
27:             break;
28:         }
29:     }
30:
31:     /* get the interpreter name */
32:     p = interpreter;
33:     noargs = 0;
34:     while(n < NAME_MAX) {
35:         if(data[n] == '\n' || data[n] == NULL) {
36:             noargs = 1;
37:             break;
38:         }
39:         if(data[n] == ' ' || data[n] == '\t') {
40:             break;
41:         }
42:         *p = data[n];
43:         n++;
44:         p++;
45:     }
46:
47:     if(!interpreter) {
48:         return -ENOEXEC;
49:     }
50:
51:
52:     /* get the interpreter arguments */
53:     if(!noargs) {
54:         p = args;
55:         /* discard possible blanks before the arguments */
56:         while(n < NAME_MAX) {
57:             if(data[n] != ' ' && data[n] != '\t') {
58:                 break;
59:             }
60:             n++;
61:         }
62:         while(n < NAME_MAX) {
63:             if(data[n] == '\n' || data[n] == NULL) {
64:                 break;
65:             }
66:             *p = data[n];
67:             n++;

```

fs/script.c

Page 2/2

```
68:                                     p++;
69:                                     }
70:     }
71:
72:     return 0;
73: }
```


fs/super.c

Page 1/4

```

1: /*
2:  * fiwix/fs/super.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/kernel.h>
10: #include <fiwix/types.h>
11: #include <fiwix/errno.h>
12: #include <fiwix/fs.h>
13: #include <fiwix/stat.h>
14: #include <fiwix/filesystems.h>
15: #include <fiwix/sleep.h>
16: #include <fiwix/sched.h>
17: #include <fiwix/stdio.h>
18: #include <fiwix/string.h>
19:
20: struct mount *mount_table;
21: static struct resource sync_resource = { NULL, NULL };
22:
23: void superblock_lock(struct superblock *sb)
24: {
25:     unsigned long int flags;
26:
27:     for(;;) {
28:         SAVE_FLAGS(flags); CLI();
29:         if(sb->locked) {
30:             sb->wanted = 1;
31:             RESTORE_FLAGS(flags);
32:             sleep(&superblock_lock, PROC_UNINTERRUPTIBLE);
33:         } else {
34:             break;
35:         }
36:     }
37:     sb->locked = 1;
38:     RESTORE_FLAGS(flags);
39: }
40:
41: void superblock_unlock(struct superblock *sb)
42: {
43:     unsigned long int flags;
44:
45:     SAVE_FLAGS(flags); CLI();
46:     sb->locked = 0;
47:     if(sb->wanted) {
48:         sb->wanted = 0;
49:         wakeup(&superblock_lock);
50:     }
51:     RESTORE_FLAGS(flags);
52: }
53:
54: struct mount * get_free_mount_point(__dev_t dev)
55: {
56:     unsigned long int flags;
57:     int n;
58:
59:     if(!dev) {
60:         printk("%s(): invalid device %d,%d.\n", __FUNCTION__, MAJOR(dev)
, MINOR(dev));
61:         return NULL;
62:     }
63:
64:     for(n = 0; n < NR_MOUNT_POINTS; n++) {
65:         if(mount_table[n].dev == dev) {
66:             printk("%s(): device %d,%d already mounted.\n", __FUNCTI

```

fs/super.c

Page 2/4

```

ON__, MAJOR(dev), MINOR(dev));
67:         return NULL;
68:     }
69: }
70:
71:     SAVE_FLAGS(flags); CLI();
72:     for(n = 0; n < NR_MOUNT_POINTS; n++) {
73:         if(!mount_table[n].used) {
74:             /* 'dev' is saved here now for get_superblock() (which
75:              * in turn is called by read_inode(), which in turn is
76:              * called by iget(), which in turn is called by
77:              * read_superblock) to be able to find the device.
78:              */
79:             mount_table[n].dev = dev;
80:             mount_table[n].used = 1;
81:             RESTORE_FLAGS(flags);
82:             return &mount_table[n];
83:         }
84:     }
85:     RESTORE_FLAGS(flags);
86:
87:     printk("WARNING: %s(): mount-point table is full.\n", __FUNCTION__);
88:     return NULL;
89: }
90:
91: void release_mount_point(struct mount *mt)
92: {
93:     memset_b(mt, NULL, sizeof(struct mount));
94: }
95:
96: struct mount * get_mount_point(struct inode *i_target)
97: {
98:     int n;
99:
100:    for(n = 0; n < NR_MOUNT_POINTS; n++) {
101:        if(mount_table[n].used) {
102:            if(S_ISDIR(i_target->i_mode)) {
103:                if(mount_table[n].sb.root == i_target) {
104:                    return &mount_table[n];
105:                }
106:            }
107:            if(S_ISBLK(i_target->i_mode)) {
108:                if(mount_table[n].dev == i_target->rdev) {
109:                    return &mount_table[n];
110:                }
111:            }
112:        }
113:    }
114:    return NULL;
115: }
116:
117: struct superblock * get_superblock(__dev_t dev)
118: {
119:     int n;
120:
121:     for(n = 0; n < NR_MOUNT_POINTS; n++) {
122:         if(mount_table[n].used && mount_table[n].dev == dev) {
123:             return &mount_table[n].sb;
124:         }
125:     }
126:     return NULL;
127: }
128:
129: void sync_superblocks(__dev_t dev)
130: {
131:     struct superblock *sb;
132:     int n, errno;

```

fs/super.c

Page 3/4

```

133:
134:     lock_resource(&sync_resource);
135:     for(n = 0; n < NR_MOUNT_POINTS; n++) {
136:         if(mount_table[n].used && (!dev || mount_table[n].dev == dev)) {
137:             sb = &mount_table[n].sb;
138:             if(sb->dirty && !(sb->flags & MS_RDONLY)) {
139:                 if(sb->fsop && sb->fsop->write_superblock) {
140:                     errno = sb->fsop->write_superblock(sb);
141:                     if(errno) {
142:                         printk("WARNING: %s(): I/O error
on device %d,%d while syncing superblock.\n", __FUNCTION__, MAJOR(sb->dev), MINOR(sb->
dev));
143:                     }
144:                 }
145:             }
146:         }
147:     }
148:     unlock_resource(&sync_resource);
149: }
150:
151: /* pseudo-file systems are only mountable by the kernel */
152: int kern_mount(__dev_t dev, struct filesystems *fs)
153: {
154:     struct mount *mt;
155:
156:     if(!(mt = get_free_mount_point(dev))) {
157:         return -EBUSY;
158:     }
159:
160:     if(fs->fsop->read_superblock(dev, &mt->sb)) {
161:         release_mount_point(mt);
162:         return -EINVAL;
163:     }
164:
165:     mt->dev = dev;
166:     strcpy(mt->devname, "none");
167:     strcpy(mt->dirname, "none");
168:     mt->sb.dir = NULL;
169:     mt->fs = fs;
170:     fs->mt = mt;
171:     return 0;
172: }
173:
174: int mount_root(void)
175: {
176:     struct filesystems *fs;
177:     struct mount *mt;
178:
179:     /* FIXME: before trying to mount the filesystem, we should first
180:      * check if '_rootdev' is a device successfully registered.
181:      */
182:
183:     if(!_rootdev) {
184:         PANIC("root device not defined.\n");
185:     }
186:
187:     if(!(fs = get_filesystem(_rootfstype))) {
188:         printk("WARNING: %s(): '%s' is not a registered filesystem. Defa
ulting to 'minix'.\n", __FUNCTION__, _rootfstype);
189:         if(!(fs = get_filesystem("minix"))) {
190:             PANIC("minix filesystem is not registered!\n");
191:         }
192:     }
193:
194:     if(!(mt = get_free_mount_point(_rootdev))) {
195:         PANIC("unable to get a free mount point.\n");
196:     }

```

fs/super.c

Page 4/4

```
197:
198:     mt->sb.flags = MS_RDONLY;
199:     if(fs->fsop && fs->fsop->read_superblock) {
200:         if(fs->fsop->read_superblock(_rootdev, &mt->sb)) {
201:             PANIC("unable to mount root filesystem on %s.\n", _rootd
evname);
202:         }
203:     }
204:
205:     strcpy(mt->devname, "/dev/root");
206:     strcpy(mt->dirname, "/");
207:     mt->dev = _rootdev;
208:     mt->sb.root->mount_point = mt->sb.root;
209:     mt->sb.root->count++;
210:     mt->sb.dir = mt->sb.root;
211:     mt->sb.dir->count++;
212:     mt->fs = fs;
213:
214:     current->root = mt->sb.root;
215:     current->root->count++;
216:     current->pwd = mt->sb.root;
217:     current->pwd->count++;
218:     iput(mt->sb.root);
219:
220:     printk("mounted root device (%s filesystem) in readonly mode.\n", fs->na
me);
221:     return 0;
222: }
223:
224: void mount_init(void)
225: {
226:     memset_b(mount_table, NULL, mount_table_size);
227: }
```

fs/ext2/bitmaps.c

Page 1/5

```

1:  /*
2:  *  fiwix/fs/ext2/bitmaps.c
3:  *
4:  *  Copyright 2019, Jordi Sanfeliu. All rights reserved.
5:  *  Distributed under the terms of the Fiwix License.
6:  */
7:
8:  #include <fiwix/kernel.h>
9:  #include <fiwix/types.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/filesystems.h>
12: #include <fiwix/fs_ext2.h>
13: #include <fiwix/buffer.h>
14: #include <fiwix/errno.h>
15: #include <fiwix/stat.h>
16: #include <fiwix/stdio.h>
17: #include <fiwix/string.h>
18:
19: static int find_first_zero(struct superblock *sb, __blk_t block)
20: {
21:     unsigned char c;
22:     int blksize;
23:     int n, n2;
24:     struct buffer *buf;
25:
26:     blksize = sb->s_blocksize;
27:
28:     if(!(buf = bread(sb->dev, block, blksize))) {
29:         return -EIO;
30:     }
31:     for(n = 0; n < blksize; n++) {
32:         c = (unsigned char)buf->data[n];
33:         for(n2 = 0; n2 < 8; n2++) {
34:             if(!(c & (1 << n2))) {
35:                 brelse(buf);
36:                 return n2 + (n * 8) + 1;
37:             }
38:         }
39:     }
40:     brelse(buf);
41:     return 0;
42: }
43:
44: static int change_bit(int mode, struct superblock *sb, __blk_t block, int item)
45: {
46:     int byte, bit, mask;
47:     struct buffer *buf;
48:
49:     block += item / (sb->s_blocksize * 8);
50:     byte = (item % (sb->s_blocksize * 8)) / 8;
51:     bit = (item % (sb->s_blocksize * 8)) % 8;
52:     mask = 1 << bit;
53:
54:     if(!(buf = bread(sb->dev, block, sb->s_blocksize))) {
55:         return -EIO;
56:     }
57:
58:     if(mode == CLEAR_BIT) {
59:         if(!(buf->data[byte] & mask)) {
60:             brelse(buf);
61:             return 1;
62:         }
63:         buf->data[byte] &= ~mask;
64:     }
65:     if(mode == SET_BIT) {
66:         if((buf->data[byte] & mask)) {
67:             brelse(buf);

```

fs/ext2/bitmaps.c

Page 2/5

```

68:             return 1;
69:         }
70:         buf->data[byte] |= mask;
71:     }
72:
73:     bwrite(buf);
74:     return 0;
75: }
76:
77: /*
78:  * Unlike of what Ext2 specifies/suggests, this inode allocation does NOT
79:  * try to assign inodes in the same block group of the directory in which
80:  * they will be created.
81:  */
82: int ext2_ialloc(struct inode *i, int mode)
83: {
84:     __ino_t inode;
85:     __blk_t block;
86:     struct superblock *sb;
87:     struct ext2_group_desc *gd;
88:     struct buffer *buf;
89:     int bg, d, errno;
90:
91:     sb = i->sb;
92:     superblock_lock(sb);
93:
94:     block = SUPERBLOCK + sb->u.ext2.sb.s_first_data_block;
95:     inode = 0;
96:     buf = NULL;
97:
98:     /* read through all group descriptors to find the first unallocated inod
e */
99:     for(bg = 0, d = 0; bg < sb->u.ext2.block_groups; bg++, d++) {
100:         if(!(bg % (sb->s_blocksize / sizeof(struct ext2_group_desc)))) {
101:             if(buf) {
102:                 brelse(buf);
103:                 block++;
104:                 d = 0;
105:             }
106:             if(!(buf = bread(sb->dev, block, sb->s_blocksize))) {
107:                 superblock_unlock(sb);
108:                 return -EIO;
109:             }
110:         }
111:         gd = (struct ext2_group_desc *) (buf->data + (d * sizeof(struct e
xt2_group_desc)));
112:         if(gd->bg_free_inodes_count) {
113:             if((inode = find_first_zero(sb, gd->bg_inode_bitmap)) {
114:                 break;
115:             }
116:         }
117:     }
118:     if(!inode) {
119:         brelse(buf);
120:         superblock_unlock(sb);
121:         return -ENOSPC;
122:     }
123:
124:     errno = change_bit(SET_BIT, sb, gd->bg_inode_bitmap, inode - 1);
125:     if(errno) {
126:         if(errno < 0) {
127:             printk("WARNING: %s(): unable to set inode %d.\n", __FUN
CTION__, inode);
128:             brelse(buf);
129:             superblock_unlock(sb);
130:             return errno;
131:         } else {

```

fs/ext2/bitmaps.c

Page 3/5

```

132:                printk("WARNING: %s(): inode %d is already marked as use
d!\n", __FUNCTION__, inode);
133:                }
134:        }
135:
136:        inode += bg * EXT2_INODES_PER_GROUP(sb);
137:        gd->bg_free_inodes_count--;
138:        sb->u.ext2.sb.s_free_inodes_count--;
139:        if(S_ISDIR(mode)) {
140:                gd->bg_used_dirs_count++;
141:        }
142:        bwrite(buf);
143:
144:        i->inode = inode;
145:        i->i_atime = CURRENT_TIME;
146:        i->i_mtime = CURRENT_TIME;
147:        i->i_ctime = CURRENT_TIME;
148:
149:        superblock_unlock(sb);
150:        return 0;
151: }
152:
153: void ext2_ifree(struct inode *i)
154: {
155:         struct ext2_group_desc *gd;
156:         struct buffer *buf;
157:         struct superblock *sb;
158:         __blk_t b, bg;
159:         int errno;
160:
161:         if(!i->inode || i->inode > i->sb->u.ext2.sb.s_inodes_count) {
162:                 printk("WARNING: %s(): invalid inode %d!\n", __FUNCTION__, i->in
ode);
163:                 return;
164:         }
165:
166:         if(i->i_blocks) {
167:                 ext2_truncate(i, 0);
168:         }
169:
170:         sb = i->sb;
171:         superblock_lock(sb);
172:
173:         b = SUPERBLOCK + sb->u.ext2.sb.s_first_data_block;
174:         bg = (i->inode - 1) / EXT2_INODES_PER_GROUP(sb);
175:         if(!(buf = bread(sb->dev, b + (bg / EXT2_DESC_PER_BLOCK(sb)), sb->s_bloc
ksize))) {
176:                 superblock_unlock(sb);
177:                 return;
178:         }
179:         gd = (struct ext2_group_desc *) (buf->data + ((bg % EXT2_DESC_PER_BLOCK(s
b)) * sizeof(struct ext2_group_desc)));
180:         errno = change_bit(CLEAR_BIT, sb, gd->bg_inode_bitmap, (i->inode - 1) %
EXT2_INODES_PER_GROUP(sb));
181:
182:         if(errno) {
183:                 if(errno < 0) {
184:                         printk("WARNING: %s(): unable to clear inode %d.\n", __F
UNCTION__, i->inode);
185:                         brelse(buf);
186:                         superblock_unlock(sb);
187:                         return;
188:                 } else {
189:                         printk("WARNING: %s(): inode %d is already marked as fre
e!\n", __FUNCTION__, i->inode);
190:                 }
191:         }

```

fs/ext2/bitmaps.c

Page 4/5

```

192:
193:     gd->bg_free_inodes_count++;
194:     sb->u.ext2.sb.s_free_inodes_count++;
195:     if(S_ISDIR(i->i_mode)) {
196:         gd->bg_used_dirs_count--;
197:     }
198:     bwrite(buf);
199:
200:     i->i_size = 0;
201:     i->i_mtime = CURRENT_TIME;
202:     i->i_ctime = CURRENT_TIME;
203:     i->dirty = 1;
204:
205:     superblock_unlock(sb);
206:     return;
207: }
208:
209: int ext2_balloc(struct superblock *sb)
210: {
211:     __blk_t b, block;
212:     struct ext2_group_desc *gd;
213:     struct buffer *buf;
214:     int bg, d, errno;
215:
216:     superblock_lock(sb);
217:
218:     b = SUPERBLOCK + sb->u.ext2.sb.s_first_data_block;
219:     block = 0;
220:     buf = NULL;
221:
222:     /* read through all group descriptors to find the first unallocated bloc
k */
223:     for(bg = 0, d = 0; bg < sb->u.ext2.block_groups; bg++, d++) {
224:         if(!(bg % (sb->s_blocksize / sizeof(struct ext2_group_desc)))) {
225:             if(buf) {
226:                 brelse(buf);
227:                 b++;
228:                 d = 0;
229:             }
230:             if(!(buf = bread(sb->dev, b, sb->s_blocksize))) {
231:                 superblock_unlock(sb);
232:                 return -EIO;
233:             }
234:         }
235:         gd = (struct ext2_group_desc *) (buf->data + (d * sizeof(struct e
xt2_group_desc)));
236:         if(gd->bg_free_blocks_count) {
237:             if((block = find_first_zero(sb, gd->bg_block_bitmap))) {
238:                 break;
239:             }
240:         }
241:     }
242:     if(!block) {
243:         brelse(buf);
244:         superblock_unlock(sb);
245:         return -ENOSPC;
246:     }
247:
248:     errno = change_bit(SET_BIT, sb, gd->bg_block_bitmap, block - 1);
249:     if(errno) {
250:         if(errno < 0) {
251:             printk("WARNING: %s(): unable to set block %d.\n", __FUN
CTION__, block);
252:             brelse(buf);
253:             superblock_unlock(sb);
254:             return errno;
255:         } else {

```


fs/ext2/bitmaps.c

Page 5/5

```

256:             printk("WARNING: %s(): block %d is already marked as use
d!\n", __FUNCTION__, block);
257:         }
258:     }
259:
260:     block += bg * EXT2_BLOCKS_PER_GROUP(sb);
261:     gd->bg_free_blocks_count--;
262:     sb->u.ext2.sb.s_free_blocks_count--;
263:     bwrite(buf);
264:
265:     superblock_unlock(sb);
266:     return block;
267: }
268:
269: void ext2_bfree(struct superblock *sb, int block)
270: {
271:     struct ext2_group_desc *gd;
272:     struct buffer *buf;
273:     __blk_t b, bg;
274:     int errno;
275:
276:     if(!block || block > sb->u.ext2.sb.s_blocks_count) {
277:         printk("WARNING: %s(): invalid block %d!\n", __FUNCTION__, block
);
278:         return;
279:     }
280:
281:     superblock_lock(sb);
282:
283:     b = SUPERBLOCK + sb->u.ext2.sb.s_first_data_block;
284:     bg = (block - 1) / EXT2_BLOCKS_PER_GROUP(sb);
285:     if(!(buf = bread(sb->dev, b + (bg / EXT2_DESC_PER_BLOCK(sb)), sb->s_bloc
ksize))) {
286:         superblock_unlock(sb);
287:         return;
288:     }
289:     gd = (struct ext2_group_desc *) (buf->data + ((bg % EXT2_DESC_PER_BLOCK(s
b)) * sizeof(struct ext2_group_desc)));
290:     errno = change_bit(CLEAR_BIT, sb, gd->bg_block_bitmap, (block - 1) % EXT
2_BLOCKS_PER_GROUP(sb));
291:
292:     if(errno) {
293:         if(errno < 0) {
294:             printk("WARNING: %s(): unable to free block %d.\n", __FU
NCTION__, block);
295:             brelse(buf);
296:             superblock_unlock(sb);
297:             return;
298:         } else {
299:             printk("WARNING: %s(): block %d is already marked as fre
e!\n", __FUNCTION__, block);
300:         }
301:     }
302:
303:     gd->bg_free_blocks_count++;
304:     sb->u.ext2.sb.s_free_blocks_count++;
305:     bwrite(buf);
306:
307:     superblock_unlock(sb);
308:     return;
309: }

```

fs/ext2/dir.c

Page 1/3

```

1: /*
2:  * fiwix/fs/ext2/dir.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/errno.h>
10: #include <fiwix/buffer.h>
11: #include <fiwix/fs.h>
12: #include <fiwix/filesystems.h>
13: #include <fiwix/stat.h>
14: #include <fiwix/dirent.h>
15: #include <fiwix/mm.h>
16: #include <fiwix/stdio.h>
17: #include <fiwix/string.h>
18:
19: struct fs_operations ext2_dir_fsop = {
20:     0,
21:     0,
22:
23:     ext2_dir_open,
24:     ext2_dir_close,
25:     ext2_dir_read,
26:     ext2_dir_write,
27:     NULL, /* ioctl */
28:     NULL, /* lseek */
29:     ext2_dir_readdir,
30:     NULL, /* mmap */
31:     NULL, /* select */
32:
33:     NULL, /* readlink */
34:     NULL, /* followlink */
35:     ext2_bmap,
36:     ext2_lookup,
37:     ext2_rmdir,
38:     ext2_link,
39:     ext2_unlink,
40:     ext2_symlink,
41:     ext2_mkdir,
42:     ext2_mknod,
43:     NULL, /* truncate */
44:     ext2_create,
45:     ext2_rename,
46:
47:     NULL, /* read_block */
48:     NULL, /* write_block */
49:
50:     NULL, /* read_inode */
51:     NULL, /* write_inode */
52:     NULL, /* ialloc */
53:     NULL, /* ifree */
54:     NULL, /* statfs */
55:     NULL, /* read_superblock */
56:     NULL, /* remount_fs */
57:     NULL, /* write_superblock */
58:     NULL, /* release_superblock */
59: };
60:
61: int ext2_dir_open(struct inode *i, struct fd *fd_table)
62: {
63:     fd_table->offset = 0;
64:     return 0;
65: }
66:
67: int ext2_dir_close(struct inode *i, struct fd *fd_table)

```

fs/ext2/dir.c

Page 2/3

```

68: {
69:     return 0;
70: }
71:
72: int ext2_dir_read(struct inode *i, struct fd *fd_table, char *buffer, __size_t c
ount)
73: {
74:     return -EISDIR;
75: }
76:
77: int ext2_dir_write(struct inode *i, struct fd *fd_table, const char *buffer, __s
ize_t count)
78: {
79:     return -EBADF;
80: }
81:
82: int ext2_dir_readdir(struct inode *i, struct fd *fd_table, struct dirent *dirent
, unsigned int count)
83: {
84:     __blk_t block;
85:     unsigned int doffset, offset;
86:     unsigned int size, dirent_len;
87:     struct ext2_dir_entry_2 *d;
88:     int base_dirent_len;
89:     int blksize;
90:     struct buffer *buf;
91:
92:     if(!(S_ISDIR(i->i_mode))) {
93:         return -EBADF;
94:     }
95:
96:     blksize = i->sb->s_blocksize;
97:     if(fd_table->offset > i->i_size) {
98:         fd_table->offset = i->i_size;
99:     }
100:
101:     base_dirent_len = sizeof(dirent->d_ino) + sizeof(dirent->d_off) + sizeof
(dirent->d_reclen);
102:     doffset = offset = size = 0;
103:
104:     while(fd_table->offset < i->i_size && count > 0) {
105:         if((block = bmap(i, fd_table->offset, FOR_READING)) < 0) {
106:             return block;
107:         }
108:         if(block) {
109:             if(!(buf = bread(i->dev, block, blksize))) {
110:                 return -EIO;
111:             }
112:
113:             doffset = fd_table->offset;
114:             offset = fd_table->offset % blksize;
115:             while(offset < blksize) {
116:                 d = (struct ext2_dir_entry_2 *) (buf->data + offs
et);
117:                 if(d->inode) {
118:                     dirent_len = (base_dirent_len + (d->name
_len + 1)) + 3;
119:                     dirent_len &= ~3; /* round up */
120:                     dirent->d_ino = d->inode;
121:                     if((size + dirent_len) < count) {
122:                         dirent->d_off = doffset;
123:                         dirent->d_reclen = dirent_len;
124:                         memcpy_b(dirent->d_name, d->name
, d->name_len);
125:                         dirent->d_name[d->name_len] = NU
LL;
126:                         dirent = (struct dirent *) ((char

```

```
*)dirent + dirent_len);
127:                                     size += dirent_len;
128:                                     count -= size;
129:                                     } else {
130:                                     count = 0;
131:                                     break;
132:                                     }
133:                                     }
134:                                     doffset += d->rec_len;
135:                                     offset += d->rec_len;
136:                                     if(!d->rec_len) {
137:                                     break;
138:                                     }
139:                                     }
140:                                     brelse(buf);
141:                                     }
142:                                     fd_table->offset &= ~(blksize - 1);
143:                                     doffset = fd_table->offset;
144:                                     fd_table->offset += offset;
145:                                     doffset += blksize;
146:                                     }
147:
148:                                     return size;
149: }
```

fs/ext2/file.c

Page 1/2

```

1:  /*
2:  *  fiwix/fs/ext2/file.c
3:  *
4:  *  Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  *  Distributed under the terms of the Fiwix License.
6:  */
7:
8:  #include <fiwix/kernel.h>
9:  #include <fiwix/types.h>
10: #include <fiwix/errno.h>
11: #include <fiwix/buffer.h>
12: #include <fiwix/fs.h>
13: #include <fiwix/filesystems.h>
14: #include <fiwix/mm.h>
15: #include <fiwix/mman.h>
16: #include <fiwix/fcntl.h>
17: #include <fiwix/stdio.h>
18: #include <fiwix/string.h>
19:
20: struct fs_operations ext2_file_fsop = {
21:     0,
22:     0,
23:
24:     ext2_file_open,
25:     ext2_file_close,
26:     file_read,
27:     ext2_file_write,
28:     NULL,                /* ioctl */
29:     ext2_file_lseek,
30:     NULL,                /* readdir */
31:     NULL,                /* mmap */
32:     NULL,                /* select */
33:
34:     NULL,                /* readlink */
35:     NULL,                /* followlink */
36:     ext2_bmap,
37:     NULL,                /* lookup */
38:     NULL,                /* rmdir */
39:     NULL,                /* link */
40:     NULL,                /* unlink */
41:     NULL,                /* symlink */
42:     NULL,                /* mkdir */
43:     NULL,                /* mknod */
44:     ext2_truncate,
45:     NULL,                /* create */
46:     NULL,                /* rename */
47:
48:     NULL,                /* read_block */
49:     NULL,                /* write_block */
50:
51:     NULL,                /* read_inode */
52:     NULL,                /* write_inode */
53:     NULL,                /* ialloc */
54:     NULL,                /* ifree */
55:     NULL,                /* statfs */
56:     NULL,                /* read_superblock */
57:     NULL,                /* remount_fs */
58:     NULL,                /* write_superblock */
59:     NULL,                /* release_superblock */
60: };
61:
62: int ext2_file_open(struct inode *i, struct fd *fd_table)
63: {
64:     if(fd_table->flags & O_APPEND) {
65:         fd_table->offset = i->i_size;
66:     } else {
67:         fd_table->offset = 0;

```

```

68:         }
69:         if(fd_table->flags & O_TRUNC) {
70:             i->i_size = 0;
71:             ext2_truncate(i, 0);
72:         }
73:         return 0;
74:     }
75:
76: int ext2_file_close(struct inode *i, struct fd *fd_table)
77: {
78:     return 0;
79: }
80:
81: int ext2_file_write(struct inode *i, struct fd *fd_table, const char *buffer, __
size_t count)
82: {
83:     __blk_t block;
84:     __off_t total_written;
85:     unsigned int boffset, bytes;
86:     int blksize;
87:     struct buffer *buf;
88:
89:     inode_lock(i);
90:
91:     blksize = i->sb->s_blocksize;
92:     total_written = 0;
93:
94:     if(fd_table->flags & O_APPEND) {
95:         fd_table->offset = i->i_size;
96:     }
97:
98:     while(total_written < count) {
99:         boffset = fd_table->offset % blksize;
100:        if((block = bmap(i, fd_table->offset, FOR_WRITING)) < 0) {
101:            inode_unlock(i);
102:            return block;
103:        }
104:        bytes = blksize - boffset;
105:        bytes = MIN(bytes, (count - total_written));
106:        if(!(buf = bread(i->dev, block, blksize))) {
107:            inode_unlock(i);
108:            return -EIO;
109:        }
110:        memcpy_b(buf->data + boffset, buffer + total_written, bytes);
111:        update_page_cache(i, fd_table->offset, buffer + total_written, b
ytes);
112:        bwrite(buf);
113:        total_written += bytes;
114:        fd_table->offset += bytes;
115:    }
116:
117:    if(fd_table->offset > i->i_size) {
118:        i->i_size = fd_table->offset;
119:    }
120:    i->i_ctime = CURRENT_TIME;
121:    i->i_mtime = CURRENT_TIME;
122:    i->dirty = 1;
123:
124:    inode_unlock(i);
125:    return total_written;
126: }
127:
128: int ext2_file_lseek(struct inode *i, __off_t offset)
129: {
130:     return offset;
131: }

```

fs/ext2/inode.c

Page 1/8

```

1: /*
2:  * fiwix/fs/ext2/inode.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/filesystems.h>
11: #include <fiwix/fs_ext2.h>
12: #include <fiwix/fs_pipe.h>
13: #include <fiwix/statfs.h>
14: #include <fiwix/sleep.h>
15: #include <fiwix/stat.h>
16: #include <fiwix/sched.h>
17: #include <fiwix/buffer.h>
18: #include <fiwix/mm.h>
19: #include <fiwix/process.h>
20: #include <fiwix/errno.h>
21: #include <fiwix/stdio.h>
22: #include <fiwix/string.h>
23:
24: #define BLOCKS_PER_IND_BLOCK(sb)          (EXT2_BLOCK_SIZE(sb) / sizeof(unsigned i
nt))
25: #define BLOCKS_PER_DIND_BLOCK(sb)        (BLOCKS_PER_IND_BLOCK(sb) * BLOCKS_PER_I
ND_BLOCK(sb))
26: #define BLOCKS_PER_TIND_BLOCK(sb)        (BLOCKS_PER_IND_BLOCK(sb) * BLOCKS_PER_I
ND_BLOCK(sb) * BLOCKS_PER_IND_BLOCK(sb))
27:
28: #define EXT2_INODES_PER_BLOCK(sb)        (EXT2_BLOCK_SIZE(sb) / sizeof(struct ext
2_inode))
29:
30: static void free_indblock(struct inode *i, int block, int offset)
31: {
32:     int n;
33:     struct buffer *buf;
34:     __blk_t *indblock;
35:
36:     if(!(buf = bread(i->dev, block, i->sb->s_blocksize))) {
37:         printk("WARNING: %s(): error reading block %d.\n", __FUNCTION__,
block);
38:         return;
39:     }
40:     indblock = (__blk_t *)buf->data;
41:     for(n = offset; n < BLOCKS_PER_IND_BLOCK(i->sb); n++) {
42:         if(indblock[n]) {
43:             ext2_bfree(i->sb, indblock[n]);
44:             indblock[n] = 0;
45:             i->i_blocks -= i->sb->s_blocksize / 512;
46:         }
47:     }
48:     bwrite(buf);
49: }
50:
51: static int get_group_desc(struct superblock *sb, __blk_t block_group, struct ext
2_group_desc *gd)
52: {
53:     __blk_t group_desc_block;
54:     int group_desc;
55:     struct buffer *buf;
56:
57:     group_desc_block = block_group / EXT2_DESC_PER_BLOCK(sb);
58:     group_desc = block_group % EXT2_DESC_PER_BLOCK(sb);
59:     if(!(buf = bread(sb->dev, SUPERBLOCK + sb->u.ext2.sb.s_first_data_block
+ group_desc_block, sb->s_blocksize))) {
60:         return -EIO;

```

fs/ext2/inode.c

Page 2/8

```

61:         }
62:         memcpy_b(gd, (void *) (buf->data + (group_desc * sizeof(struct ext2_group
_desc))), sizeof(struct ext2_group_desc));
63:         brelse(buf);
64:         return 0;
65:     }
66:
67: int ext2_read_inode(struct inode *i)
68: {
69:     __blk_t block_group, block;
70:     unsigned int offset;
71:     struct superblock *sb;
72:     struct ext2_inode *ii;
73:     struct ext2_group_desc gd;
74:     struct buffer *buf;
75:
76:     if(!(sb = get_superblock(i->dev))) {
77:         printk("WARNING: %s(): get_superblock() has returned NULL.\n");
78:         return -EINVAL;
79:     }
80:     block_group = ((i->inode - 1) / EXT2_INODES_PER_GROUP(sb));
81:     if(get_group_desc(sb, block_group, &gd) {
82:         return -EIO;
83:     }
84:     block = (((i->inode - 1) % EXT2_INODES_PER_GROUP(sb)) / EXT2_INODES_PER_
BLOCK(sb));
85:
86:     if(!(buf = bread(i->dev, gd.bg_inode_table + block, i->sb->s_blocksize))
) {
87:         return -EIO;
88:     }
89:     offset = (((i->inode - 1) % EXT2_INODES_PER_GROUP(sb)) % EXT2_INODES_PE
R_BLOCK(sb)) * sizeof(struct ext2_inode));
90:
91:     ii = (struct ext2_inode *) (buf->data + offset);
92:     memcpy_b(&i->u.ext2.i_data, ii->i_block, sizeof(ii->i_block));
93:
94:     i->i_mode = ii->i_mode;
95:     i->i_uid = ii->i_uid;
96:     i->i_size = ii->i_size;
97:     i->i_atime = ii->i_atime;
98:     i->i_ctime = ii->i_ctime;
99:     i->i_mtime = ii->i_mtime;
100:    i->i_gid = ii->i_gid;
101:    i->i_nlink = ii->i_links_count;
102:    i->i_blocks = ii->i_blocks;
103:    i->i_flags = ii->i_flags;
104:    i->count = 1;
105:    switch(i->i_mode & S_IFMT) {
106:        case S_IFCHR:
107:            i->fsop = &def_chr_fsop;
108:            i->rdev = ii->i_block[0];
109:            break;
110:        case S_IFBLK:
111:            i->fsop = &def_blk_fsop;
112:            i->rdev = ii->i_block[0];
113:            break;
114:        case S_IFIFO:
115:            i->fsop = &pipefs_fsop;
116:            /* it's a union so we need to clear pipefs_i */
117:            memset_b(&i->u.pipefs, NULL, sizeof(struct pipefs_inode)
);
118:            break;
119:        case S_IFDIR:
120:            i->fsop = &ext2_dir_fsop;
121:            break;
122:        case S_IFREG:

```


fs/ext2/inode.c

Page 3/8

```

123:             i->fsop = &ext2_file_fsop;
124:             break;
125:         case S_IFLNK:
126:             i->fsop = &ext2_symlink_fsop;
127:             break;
128:         case S_IFSOCK:
129:             i->fsop = NULL;
130:             break;
131:         default:
132:             printk("WARNING: %s(): invalid inode (%d) mode %08o.\n",
__FUNCTION__, i->inode, i->i_mode);
133:             brelse(buf);
134:             return -ENOENT;
135:     }
136:     brelse(buf);
137:     return 0;
138: }
139:
140: int ext2_write_inode(struct inode *i)
141: {
142:     __blk_t block_group, block;
143:     short int offset;
144:     struct superblock *sb;
145:     struct ext2_inode *ii;
146:     struct ext2_group_desc gd;
147:     struct buffer *buf;
148:
149:     if(!(sb = get_superblock(i->dev))) {
150:         printk("WARNING: %s(): get_superblock() has returned NULL.\n");
151:         return -EINVAL;
152:     }
153:     block_group = ((i->inode - 1) / EXT2_INODES_PER_GROUP(sb));
154:     if(get_group_desc(sb, block_group, &gd) {
155:         return -EIO;
156:     }
157:     block = (((i->inode - 1) % EXT2_INODES_PER_GROUP(sb)) / EXT2_INODES_PER_
BLOCK(sb));
158:
159:     if(!(buf = bread(i->dev, gd.bg_inode_table + block, i->sb->s_blocksize))
) {
160:         return -EIO;
161:     }
162:     offset = (((i->inode - 1) % EXT2_INODES_PER_GROUP(sb)) % EXT2_INODES_PE
R_BLOCK(sb)) * sizeof(struct ext2_inode);
163:     ii = (struct ext2_inode *) (buf->data + offset);
164:     memset_b(ii, 0, sizeof(struct ext2_inode));
165:
166:     ii->i_mode = i->i_mode;
167:     ii->i_uid = i->i_uid;
168:     ii->i_size = i->i_size;
169:     ii->i_atime = i->i_atime;
170:     ii->i_ctime = i->i_ctime;
171:     ii->i_mtime = i->i_mtime;
172:     ii->i_dtime = i->u.ext2.i_dtime;
173:     ii->i_gid = i->i_gid;
174:     ii->i_links_count = i->i_nlink;
175:     ii->i_blocks = i->i_blocks;
176:     ii->i_flags = i->i_flags;
177:     if(S_ISCHR(i->i_mode) || S_ISBLK(i->i_mode)) {
178:         ii->i_block[0] = i->rdev;
179:     } else {
180:         memcpy_b(ii->i_block, &i->u.ext2.i_data, sizeof(i->u.ext2.i_data
));
181:     }
182:     i->dirty = 0;
183:     bwrite(buf);
184:     return 0;

```

fs/ext2/inode.c

Page 4/8

```

185: }
186:
187: int ext2_bmap(struct inode *i, __off_t offset, int mode)
188: {
189:     unsigned char level;
190:     __blk_t *indblock, *dindblock, *tindblock;
191:     __blk_t block, iblock, dblock, tblock, newblock;
192:     int blksize;
193:     struct buffer *buf, *buf2, *buf3, *buf4;
194:
195:     blksize = i->sb->s_blocksize;
196:     block = offset / blksize;
197:     level = 0;
198:     buf3 = NULL;    /* makes GCC happy */
199:
200:     if(block < EXT2_NDIR_BLOCKS) {
201:         level = EXT2_NDIR_BLOCKS - 1;
202:     } else {
203:         if(block < (BLOCKS_PER_IND_BLOCK(i->sb) + EXT2_NDIR_BLOCKS)) {
204:             level = EXT2_IND_BLOCK;
205:         } else if(block < ((BLOCKS_PER_IND_BLOCK(i->sb) * BLOCKS_PER_IND
_BLOCK(i->sb)) + BLOCKS_PER_IND_BLOCK(i->sb) + EXT2_NDIR_BLOCKS)) {
206:             level = EXT2_DIND_BLOCK;
207:         } else {
208:             level = EXT2_TIND_BLOCK;
209:         }
210:         block -= EXT2_NDIR_BLOCKS;
211:     }
212:
213:     if(level < EXT2_NDIR_BLOCKS) {
214:         if(!i->u.ext2.i_data[block] && mode == FOR_WRITING) {
215:             if((newblock = ext2_balloc(i->sb)) < 0) {
216:                 return -ENOSPC;
217:             }
218:             /* initialize the new block */
219:             if(!(buf = bread(i->dev, newblock, blksize))) {
220:                 ext2_bfree(i->sb, newblock);
221:                 return -EIO;
222:             }
223:             memset_b(buf->data, 0, blksize);
224:             bwrite(buf);
225:             i->u.ext2.i_data[block] = newblock;
226:             i->i_blocks += blksize / 512;
227:         }
228:         return i->u.ext2.i_data[block];
229:     }
230:
231:     if(!i->u.ext2.i_data[level]) {
232:         if(mode == FOR_WRITING) {
233:             if((newblock = ext2_balloc(i->sb)) < 0) {
234:                 return -ENOSPC;
235:             }
236:             /* initialize the new block */
237:             if(!(buf = bread(i->dev, newblock, blksize))) {
238:                 ext2_bfree(i->sb, newblock);
239:                 return -EIO;
240:             }
241:             memset_b(buf->data, 0, blksize);
242:             bwrite(buf);
243:             i->u.ext2.i_data[level] = newblock;
244:             i->i_blocks += blksize / 512;
245:         } else {
246:             return 0;
247:         }
248:     }
249:     if(!(buf = bread(i->dev, i->u.ext2.i_data[level], blksize))) {
250:         return -EIO;

```

fs/ext2/inode.c

Page 5/8

```

251:         }
252:         indblock = (__blk_t *)buf->data;
253:         dblock = block - BLOCKS_PER_IND_BLOCK(i->sb);
254:         tblock = block - (BLOCKS_PER_IND_BLOCK(i->sb) * BLOCKS_PER_IND_BLOCK(i->
sb)) - BLOCKS_PER_IND_BLOCK(i->sb);
255:
256:         if(level == EXT2_DIND_BLOCK) {
257:             block = dblock / BLOCKS_PER_IND_BLOCK(i->sb);
258:         }
259:         if(level == EXT2_TIND_BLOCK) {
260:             block = tblock / (BLOCKS_PER_IND_BLOCK(i->sb) * BLOCKS_PER_IND_B
LOCK(i->sb));
261:         }
262:
263:         if(!indblock[block]) {
264:             if(mode == FOR_WRITING) {
265:                 if((newblock = ext2_balloc(i->sb)) < 0) {
266:                     brelse(buf);
267:                     return -ENOSPC;
268:                 }
269:                 /* initialize the new block */
270:                 if(!(buf2 = bread(i->dev, newblock, blksize))) {
271:                     ext2_bfree(i->sb, newblock);
272:                     brelse(buf);
273:                     return -EIO;
274:                 }
275:                 memset_b(buf2->data, 0, blksize);
276:                 bwrite(buf2);
277:                 indblock[block] = newblock;
278:                 i->i_blocks += blksize / 512;
279:                 if(level == EXT2_IND_BLOCK) {
280:                     bwrite(buf);
281:                     return newblock;
282:                 }
283:                 buf->flags |= (BUFFER_DIRTY | BUFFER_VALID);
284:             } else {
285:                 brelse(buf);
286:                 return 0;
287:             }
288:         }
289:         if(level == EXT2_IND_BLOCK) {
290:             newblock = indblock[block];
291:             brelse(buf);
292:             return newblock;
293:         }
294:
295:         if(level == EXT2_TIND_BLOCK) {
296:             if(!(buf3 = bread(i->dev, indblock[block], blksize))) {
297:                 printk("%s(): returning -EIO\n", __FUNCTION__);
298:                 brelse(buf);
299:                 return -EIO;
300:             }
301:             tindblock = (__blk_t *)buf3->data;
302:             block = tindblock[tblock / BLOCKS_PER_IND_BLOCK(i->sb)];
303:             if(!block) {
304:                 if(mode == FOR_WRITING) {
305:                     if((newblock = ext2_balloc(i->sb)) < 0) {
306:                         brelse(buf);
307:                         brelse(buf3);
308:                         return -ENOSPC;
309:                     }
310:                     /* initialize the new block */
311:                     if(!(buf4 = bread(i->dev, newblock, blksize))) {
312:                         ext2_bfree(i->sb, newblock);
313:                         brelse(buf);
314:                         brelse(buf3);
315:                         return -EIO;

```

fs/ext2/inode.c

Page 6/8

```

316:         }
317:         memset_b(buf4->data, 0, blksize);
318:         bwrite(buf4);
319:         tindblock[tblock / BLOCKS_PER_IND_BLOCK(i->sb)]
= newblock;
320:         i->i_blocks += blksize / 512;
321:         buf3->flags |= (BUFFER_DIRTY | BUFFER_VALID);
322:         block = newblock;
323:         } else {
324:             brelse(buf);
325:             brelse(buf3);
326:             return 0;
327:         }
328:     }
329:     dblock = tblock;
330:     iblock = tblock / BLOCKS_PER_IND_BLOCK(i->sb);
331:     if(!(buf2 = bread(i->dev, block, blksize))) {
332:         printk("%s(): returning -EIO\n", __FUNCTION__);
333:         brelse(buf);
334:         brelse(buf3);
335:         return -EIO;
336:     }
337: } else {
338:     iblock = block;
339:     if(!(buf2 = bread(i->dev, indblock[iblock], blksize))) {
340:         printk("%s(): returning -EIO\n", __FUNCTION__);
341:         brelse(buf);
342:         return -EIO;
343:     }
344: }
345:
346: dindblock = (__blk_t *)buf2->data;
347: block = dindblock[dblock - (iblock * BLOCKS_PER_IND_BLOCK(i->sb))];
348: if(!block && mode == FOR_WRITING) {
349:     if((newblock = ext2_balloc(i->sb)) < 0) {
350:         brelse(buf);
351:         if(level == EXT2_TIND_BLOCK) {
352:             brelse(buf3);
353:         }
354:         brelse(buf2);
355:         return -ENOSPC;
356:     }
357:     /* initialize the new block */
358:     if(!(buf4 = bread(i->dev, newblock, blksize))) {
359:         ext2_bfree(i->sb, newblock);
360:         brelse(buf);
361:         if(level == EXT2_TIND_BLOCK) {
362:             brelse(buf3);
363:         }
364:         brelse(buf2);
365:         return -EIO;
366:     }
367:     memset_b(buf4->data, 0, blksize);
368:     bwrite(buf4);
369:     dindblock[dblock - (iblock * BLOCKS_PER_IND_BLOCK(i->sb))] = new
block;
370:     i->i_blocks += blksize / 512;
371:     buf2->flags |= (BUFFER_DIRTY | BUFFER_VALID);
372:     block = newblock;
373: }
374: brelse(buf);
375: if(level == EXT2_TIND_BLOCK) {
376:     brelse(buf3);
377: }
378: brelse(buf2);
379: return block;
380: }

```

fs/ext2/inode.c

Page 7/8

```

381:
382: int ext2_truncate(struct inode *i, __off_t length)
383: {
384:     __blk_t block, dblock, *indblock;
385:     struct buffer *buf;
386:     int blksize, n;
387:
388:     blksize = i->sb->s_blocksize;
389:     block = length / blksize;
390:
391:     if(!S_ISDIR(i->i_mode) && !S_ISREG(i->i_mode) && !S_ISLNK(i->i_mode)) {
392:         return -EINVAL;
393:     }
394:
395:     if(block < EXT2_NDIR_BLOCKS) {
396:         for(n = block; n < EXT2_NDIR_BLOCKS; n++) {
397:             if(i->u.ext2.i_data[n]) {
398:                 ext2_bfree(i->sb, i->u.ext2.i_data[n]);
399:                 i->u.ext2.i_data[n] = 0;
400:                 i->i_blocks -= blksize / 512;
401:             }
402:         }
403:         block = 0;
404:     }
405:
406:     if(!block || block < (BLOCKS_PER_IND_BLOCK(i->sb) + EXT2_NDIR_BLOCKS)) {
407:         if(block) {
408:             block -= EXT2_NDIR_BLOCKS;
409:         }
410:         if(i->u.ext2.i_data[EXT2_IND_BLOCK]) {
411:             free_indblock(i, i->u.ext2.i_data[EXT2_IND_BLOCK], block
);
412:             if(!block) {
413:                 ext2_bfree(i->sb, i->u.ext2.i_data[EXT2_IND_BLOCK
K]);
414:                 i->u.ext2.i_data[EXT2_IND_BLOCK] = 0;
415:                 i->i_blocks -= blksize / 512;
416:             }
417:         }
418:         block = 0;
419:     }
420:
421:     if(block) {
422:         block -= EXT2_NDIR_BLOCKS;
423:         block -= BLOCKS_PER_IND_BLOCK(i->sb);
424:     }
425:     if(i->u.ext2.i_data[EXT2_DIND_BLOCK]) {
426:         if(!(buf = bread(i->dev, i->u.ext2.i_data[EXT2_DIND_BLOCK], blksize))) {
427:             printk("%s(): error reading block %d.\n", __FUNCTION__,
i->u.ext2.i_data[EXT2_DIND_BLOCK]);
428:             return -EIO;
429:         }
430:         indblock = (__blk_t *)buf->data;
431:         dblock = block % BLOCKS_PER_IND_BLOCK(i->sb);
432:         for(n = block / BLOCKS_PER_IND_BLOCK(i->sb); n < BLOCKS_PER_IND_
BLOCK(i->sb); n++) {
433:             if(indblock[n]) {
434:                 free_indblock(i, indblock[n], dblock);
435:                 if(!dblock) {
436:                     ext2_bfree(i->sb, indblock[n]);
437:                     i->i_blocks -= blksize / 512;
438:                 }
439:             }
440:             dblock = 0;
441:         }
442:         bwrite(buf);

```

fs/ext2/inode.c

Page 8/8

```
443:             if(!block) {
444:                 ext2_bfree(i->sb, i->u.ext2.i_data[EXT2_DIND_BLOCK]);
445:                 i->u.ext2.i_data[EXT2_DIND_BLOCK] = 0;
446:                 i->i_blocks -= blksize / 512;
447:             }
448:         }
449:
450:         i->i_mtime = CURRENT_TIME;
451:         i->i_ctime = CURRENT_TIME;
452:         i->i_size = length;
453:         i->dirty = 1;
454:
455:         return 0;
456: }
```

fs/ext2/Makefile

Page 1/1

```
1: # fiwix/fs/ext2/Makefile
2: #
3: # Copyright 2018, Jordi Sanfeliu. All rights reserved.
4: # Distributed under the terms of the Fiwix License.
5: #
6: #
7: .S.o:
8:     $(CC) -traditional -I$(INCLUDE) -c -o $@ $<
9: .c.o:
10:     $(CC) $(CFLAGS) -c -o $@ $<
11:
12: OBJS = inode.o super.o namei.o symlink.o dir.o file.o bitmaps.o
13:
14: ext2:     $(OBJS)
15:             $(LD) $(LDFLAGS) -r $(OBJS) -o ext2.o
16:
17: clean:
18:             rm -f *.o
19:
```

fs/ext2/namei.c

Page 1/13

```

1: /*
2:  * fiwix/fs/ext2/namei.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/types.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/filesystems.h>
12: #include <fiwix/fs_ext2.h>
13: #include <fiwix/buffer.h>
14: #include <fiwix/mm.h>
15: #include <fiwix/errno.h>
16: #include <fiwix/stat.h>
17: #include <fiwix/stdio.h>
18: #include <fiwix/string.h>
19:
20: /* finds the entry 'name' with (optionally) inode 'i' in the directory 'dir' */
21: static struct buffer * find_dir_entry(struct inode *dir, struct inode *i, struct
ext2_dir_entry_2 **d_res, char *name)
22: {
23:     __blk_t block;
24:     unsigned int blksize;
25:     unsigned int offset, doffset;
26:     struct buffer *buf;
27:     int basesize, rlen, nlen;
28:
29:     basesize = sizeof((*d_res)->inode) + sizeof((*d_res)->rec_len) + sizeof(
(*d_res)->name_len) + sizeof((*d_res)->file_type);
30:     blksize = dir->sb->s_blocksize;
31:     offset = 0;
32:
33:     /* nlen is the length of the new entry, used when searching for the first
usable entry */
34:     nlen = basesize + strlen(name) + 3;
35:     nlen &= ~3;
36:
37:     while(offset < dir->i_size) {
38:         if((block = bmap(dir, offset, FOR_READING)) < 0) {
39:             break;
40:         }
41:         if(block) {
42:             if(!(buf = bread(dir->dev, block, blksize))) {
43:                 break;
44:             }
45:             doffset = 0;
46:             do {
47:                 *d_res = (struct ext2_dir_entry_2 *) (buf->data +
doffset);
48:                 if(!i) {
49:                     /* calculates the real length of the current
entry */
50:                     rlen = basesize + strlen((*d_res)->name)
+ 3;
51:                     rlen &= ~3;
52:                     /* returns the first entry where *name can
fit in */
53:                     if(!(*d_res)->inode) {
54:                         if(nlen <= (*d_res)->rec_len) {
55:                             return buf;
56:                         }
57:                     } else {
58:                         if(rlen + nlen <= (*d_res)->rec_
len) {
59:                             return buf;

```


fs/ext2/namei.c

Page 2/13

```

60:
61:                                     nrec_len = (*d_res)->rec
_len - rlen;
62:                                     (*d_res)->rec_len = rlen
;
63:                                     doffset += (*d_res)->rec
_len;
64:                                     *d_res = (struct ext2_di
r_entry_2 *) (buf->data + doffset);
65:                                     (*d_res)->rec_len = nrec
_len;
66:                                     return buf;
67:                                     }
68:                                     }
69:                                     doffset += (*d_res)->rec_len;
70:                                     } else {
71:                                     if((*d_res)->inode == i->inode) {
72:                                     /* returns the first matching in
ode */
73:                                     if(!name) {
74:                                     return buf;
75:                                     }
76:                                     /* returns the matching inode an
d name */
77:                                     if((*d_res)->name_len == strlen(
name)) {
78:                                     if(!strncmp((*d_res)->na
me, name, (*d_res)->name_len)) {
79:                                     return buf;
80:                                     }
81:                                     }
82:                                     }
83:                                     doffset += (*d_res)->rec_len;
84:                                     }
85:                                     } while(doffset < blksize);
86:                                     brelse(buf);
87:                                     offset += blksize;
88:                                     } else {
89:                                     break;
90:                                     }
91:                                     }
92:
93:                                     *d_res = NULL;
94:                                     return NULL;
95: }
96:
97: static struct buffer * add_dir_entry(struct inode *dir, struct ext2_dir_entry_2
**d_res, char *name)
98: {
99:     __blk_t block;
100:    struct buffer *buf;
101:
102:    if(!(buf = find_dir_entry(dir, NULL, d_res, name))) {
103:        if((block = bmap(dir, dir->i_size, FOR_WRITING)) < 0) {
104:            return NULL;
105:        }
106:        if(!(buf = bread(dir->dev, block, dir->sb->s_blocksize))) {
107:            return NULL;
108:        }
109:        *d_res = (struct ext2_dir_entry_2 *)buf->data;
110:        dir->i_size += dir->sb->s_blocksize;
111:        (*d_res)->rec_len = dir->sb->s_blocksize;
112:    }
113:
114:    return buf;
115: }
116:

```

fs/ext2/namei.c

Page 3/13

```

117: static int is_dir_empty(struct inode *dir)
118: {
119:     __blk_t block;
120:     unsigned int blksize;
121:     unsigned int offset, doffset;
122:     struct buffer *buf;
123:     struct ext2_dir_entry_2 *d;
124:
125:     blksize = dir->sb->s_blocksize;
126:     offset = 0;
127:
128:     while(offset < dir->i_size) {
129:         if((block = bmap(dir, offset, FOR_READING)) < 0) {
130:             break;
131:         }
132:         if(block) {
133:             if(!(buf = bread(dir->dev, block, blksize))) {
134:                 break;
135:             }
136:             doffset = 0;
137:             do {
138:                 if(doffset + offset >= dir->i_size) {
139:                     break;
140:                 }
141:                 d = (struct ext2_dir_entry_2 *) (buf->data + doff
set);
142:                 doffset += d->rec_len;
143:                 if(d->inode && d->name_len == 1 && d->name[0] ==
'.' ) {
144:                     continue;
145:                 }
146:                 if(d->inode && d->name_len == 2 && d->name[0] ==
'..' && d->name[1] == '.') {
147:                     continue;
148:                 }
149:                 if(d->inode) {
150:                     brelse(buf);
151:                     return 0;
152:                 }
153:             } while(doffset < blksize);
154:             brelse(buf);
155:             offset += blksize;
156:         } else {
157:             break;
158:         }
159:     }
160:
161:     return 1;
162: }
163:
164: static int is_subdir(struct inode *dir_new, struct inode *i_old)
165: {
166:     __ino_t inode;
167:     int errno;
168:
169:     errno = 0;
170:     dir_new->count++;
171:     for(;;) {
172:         if(dir_new == i_old) {
173:             errno = 1;
174:             break;
175:         }
176:         inode = dir_new->inode;
177:         if(ext2_lookup(".", dir_new, &dir_new)) {
178:             break;
179:         }
180:         if(dir_new->inode == inode) {

```

fs/ext2/namei.c

Page 4/13

```

181:                break;
182:            }
183:        }
184:        iput(dir_new);
185:        return errno;
186:    }
187:
188:    int ext2_lookup(const char *name, struct inode *dir, struct inode **i_res)
189:    {
190:        __blk_t block;
191:        unsigned int blksize;
192:        unsigned int offset, doffset;
193:        struct buffer *buf;
194:        struct ext2_dir_entry_2 *d;
195:        __ino_t inode;
196:
197:        blksize = dir->sb->s_blocksize;
198:        inode = offset = 0;
199:
200:        while(offset < dir->i_size && !inode) {
201:            if((block = bmap(dir, offset, FOR_READING)) < 0) {
202:                return block;
203:            }
204:            if(block) {
205:                if(!(buf = bread(dir->dev, block, blksize))) {
206:                    iput(dir);
207:                    return -EIO;
208:                }
209:                doffset = 0;
210:                do {
211:                    d = (struct ext2_dir_entry_2 *) (buf->data + doffset);
212:                    if(d->inode) {
213:                        if(d->name_len == strlen(name)) {
214:                            if(strncmp(d->name, name, d->name_len) == 0) {
215:                                inode = d->inode;
216:                            }
217:                        }
218:                    }
219:                    doffset += d->rec_len;
220:                } while((doffset < blksize) && (!inode));
221:
222:                brelse(buf);
223:                offset += blksize;
224:                if(inode) {
225:                    /*
226:                     * This prevents a deadlock in iget() when
227:                     * trying to lock '.' when 'dir' is the same
228:                     * directory (ls -lai <dir>).
229:                     */
230:                    if(inode == dir->inode) {
231:                        *i_res = dir;
232:                        return 0;
233:                    }
234:
235:                    if(!(*i_res = iget(dir->sb, inode))) {
236:                        iput(dir);
237:                        return -EACCES;
238:                    }
239:                    iput(dir);
240:                    return 0;
241:                }
242:            } else {
243:                break;
244:            }
245:        }

```

fs/ext2/namei.c

Page 5/13

```

246:         iput(dir);
247:         return -ENOENT;
248:     }
249:
250: int ext2_rmdir(struct inode *dir, struct inode *i)
251: {
252:     struct buffer *buf;
253:     struct ext2_dir_entry_2 *d;
254:
255:     inode_lock(i);
256:
257:     if(!is_dir_empty(i)) {
258:         inode_unlock(i);
259:         return -ENOTEMPTY;
260:     }
261:
262:     inode_lock(dir);
263:
264:     if(!(buf = find_dir_entry(dir, i, &d, NULL))) {
265:         inode_unlock(i);
266:         inode_unlock(dir);
267:         return -ENOENT;
268:     }
269:
270:     d->inode = 0;
271:     i->i_nlink = 0;
272:     dir->i_nlink--;
273:
274:     i->i_ctime = CURRENT_TIME;
275:     i->u.ext2.i_dtime = CURRENT_TIME;
276:     dir->i_mtime = CURRENT_TIME;
277:     dir->i_ctime = CURRENT_TIME;
278:
279:     i->dirty = 1;
280:     dir->dirty = 1;
281:
282:     bwrite(buf);
283:
284:     inode_unlock(i);
285:     inode_unlock(dir);
286:     return 0;
287: }
288:
289: int ext2_link(struct inode *i_old, struct inode *dir_new, char *name)
290: {
291:     struct buffer *buf;
292:     struct ext2_dir_entry_2 *d;
293:     char c;
294:     int n;
295:
296:     inode_lock(i_old);
297:     inode_lock(dir_new);
298:
299:     if(!(buf = add_dir_entry(dir_new, &d, name))) {
300:         inode_unlock(i_old);
301:         inode_unlock(dir_new);
302:         return -ENOSPC;
303:     }
304:
305:     d->inode = i_old->inode;
306:     d->name_len = strlen(name);
307:     /* strcpy() can't be used here because it places a trailing NULL */
308:     for(n = 0; n < NAME_MAX; n++) {
309:         if((c = name[n])) {
310:             d->name[n] = c;
311:             continue;
312:         }

```

fs/ext2/namei.c

Page 6/13

```

313:             break;
314:         }
315:         d->file_type = 0;           /* not used */
316:
317:         i_old->i_nlink++;
318:         i_old->i_ctime = CURRENT_TIME;
319:         dir_new->i_mtime = CURRENT_TIME;
320:         dir_new->i_ctime = CURRENT_TIME;
321:
322:         i_old->dirty = 1;
323:         dir_new->dirty = 1;
324:
325:         bwrite(buf);
326:
327:         inode_unlock(i_old);
328:         inode_unlock(dir_new);
329:         return 0;
330:     }
331:
332: int ext2_unlink(struct inode *dir, struct inode *i, char *name)
333: {
334:     struct buffer *buf;
335:     struct ext2_dir_entry_2 *d;
336:
337:     inode_lock(dir);
338:     inode_lock(i);
339:
340:     if (!(buf = find_dir_entry(dir, i, &d, name))) {
341:         inode_unlock(dir);
342:         inode_unlock(i);
343:         return -ENOENT;
344:     }
345:
346:     /*
347:      * FIXME: in order to avoid low performance when traversing large
348:      * directories plenty of blank entries, it would be interesting
349:      * to merge every removed entry with the previous entry.
350:      */
351:     d->inode = 0;
352:     if (!--i->i_nlink) {
353:         i->u.ext2.i_dtime = CURRENT_TIME;
354:     }
355:
356:     i->i_ctime = CURRENT_TIME;
357:     dir->i_mtime = CURRENT_TIME;
358:     dir->i_ctime = CURRENT_TIME;
359:
360:     i->dirty = 1;
361:     dir->dirty = 1;
362:
363:     bwrite(buf);
364:
365:     inode_unlock(dir);
366:     inode_unlock(i);
367:     return 0;
368: }
369:
370: int ext2_symlink(struct inode *dir, char *name, char *oldname)
371: {
372:     struct buffer *buf, *buf2;
373:     struct inode *i;
374:     struct ext2_dir_entry_2 *d;
375:     __blk_t block;
376:     char c, *data;
377:     int n;
378:
379:     inode_lock(dir);

```

fs/ext2/namei.c

Page 7/13

```

380:
381:     if (!(i = ialloc(dir->sb, S_IFLNK)) {
382:         inode_unlock(dir);
383:         return -ENOSPC;
384:     }
385:
386:     if (!(buf = add_dir_entry(dir, &d, name))) {
387:         iput(i);
388:         inode_unlock(dir);
389:         return -ENOSPC;
390:     }
391:
392:     i->i_mode = S_IFLNK | (S_IRWXU | S_IRWXG | S_IRWXO);
393:     i->i_uid = current->euid;
394:     i->i_gid = current->egid;
395:     i->dev = dir->dev;
396:     i->count = 1;
397:     i->fsop = &ext2_symlink_fsop;
398:
399:     if (strlen(oldname) >= EXT2_N_BLOCKS * sizeof(__u32)) {
400:         /* this will be a slow symlink */
401:         if ((block = ext2_balloc(dir->sb)) < 0) {
402:             iput(i);
403:             brelse(buf);
404:             inode_unlock(dir);
405:             return block;
406:         }
407:         if (!(buf2 = bread(dir->dev, block, dir->sb->s_blocksize))) {
408:             iput(i);
409:             brelse(buf);
410:             ext2_bfree(dir->sb, block);
411:             inode_unlock(dir);
412:             return -EIO;
413:         }
414:         i->u.ext2.i_data[0] = block;
415:         for (n = 0; n < NAME_MAX; n++) {
416:             if ((c = oldname[n])) {
417:                 buf2->data[n] = c;
418:                 continue;
419:             }
420:             break;
421:         }
422:         buf2->data[n] = 0;
423:         i->i_blocks = dir->sb->s_blocksize / 512;
424:         bwrite(buf2);
425:     } else {
426:         /* this will be a fast symlink */
427:         data = (char *)i->u.ext2.i_data;
428:         for (n = 0; n < NAME_MAX; n++) {
429:             if ((c = oldname[n])) {
430:                 data[n] = c;
431:                 continue;
432:             }
433:             break;
434:         }
435:         data[n] = 0;
436:     }
437:
438:     i->i_size = n;
439:     i->dirty = 1;
440:     i->i_nlink = 1;
441:     d->inode = i->inode;
442:     d->name_len = strlen(name);
443:     /* strcpy() can't be used here because it places a trailing NULL */
444:     for (n = 0; n < NAME_MAX; n++) {
445:         if ((c = name[n])) {
446:             d->name[n] = c;

```

fs/ext2/namei.c

Page 8/13

```

447:             continue;
448:         }
449:         break;
450:     }
451:     d->file_type = 0;        /* EXT2_FT_SYMLINK not used */
452:
453:     dir->i_mtime = CURRENT_TIME;
454:     dir->i_ctime = CURRENT_TIME;
455:     dir->dirty = 1;
456:
457:     bwrite(buf);
458:     iput(i);
459:     inode_unlock(dir);
460:     return 0;
461: }
462:
463: int ext2_mkdir(struct inode *dir, char *name, __mode_t mode)
464: {
465:     struct buffer *buf, *buf2;
466:     struct inode *i;
467:     struct ext2_dir_entry_2 *d, *d2;
468:     __blk_t block;
469:     char c;
470:     int n;
471:
472:     inode_lock(dir);
473:
474:     if(!(i = ialloc(dir->sb, S_IFDIR))) {
475:         inode_unlock(dir);
476:         return -ENOSPC;
477:     }
478:
479:     i->i_mode = ((mode & (S_IRWXU | S_IRWXG | S_IRWXO)) & ~current->umask);
480:     i->i_mode |= S_IFDIR;
481:     i->i_uid = current->euid;
482:     i->i_gid = current->egid;
483:     i->dev = dir->dev;
484:     i->count = 1;
485:     i->fsop = &ext2_dir_fsop;
486:
487:     if((block = bmap(i, 0, FOR_WRITING)) < 0) {
488:         iput(i);
489:         inode_unlock(dir);
490:         return block;
491:     }
492:
493:     if(!(buf2 = bread(i->dev, block, dir->sb->s_blocksize))) {
494:         ext2_bfree(dir->sb, block);
495:         iput(i);
496:         inode_unlock(dir);
497:         return -EIO;
498:     }
499:
500:     if(!(buf = add_dir_entry(dir, &d, name))) {
501:         ext2_bfree(dir->sb, block);
502:         iput(i);
503:         brelse(buf2);
504:         inode_unlock(dir);
505:         return -ENOSPC;
506:     }
507:
508:     d->inode = i->inode;
509:     d->name_len = strlen(name);
510:     /* strcpy() can't be used here because it places a trailing NULL */
511:     for(n = 0; n < NAME_MAX; n++) {
512:         if((c = name[n])) {
513:             if(c != '/') {

```

fs/ext2/namei.c

Page 9/13

```

514:                                     d->name[n] = c;
515:                                     continue;
516:                                     }
517:                                 }
518:                                break;
519:                            }
520:                            d->file_type = 0;          /* EXT2_FT_DIR not used */
521:
522:                            d2 = (struct ext2_dir_entry_2 *)buf2->data;
523:                            d2->inode = i->inode;
524:                            d2->name[0] = '.';
525:                            d2->name[1] = 0;
526:                            d2->name_len = 1;
527:                            d2->rec_len = 12;
528:                            d2->file_type = 0;          /* EXT2_FT_DIR not used */
529:                            i->i_nlink = 1;
530:                            d2 = (struct ext2_dir_entry_2 *) (buf2->data + 12);
531:                            d2->inode = dir->inode;
532:                            d2->name[0] = '.';
533:                            d2->name[1] = '.';
534:                            d2->name[2] = 0;
535:                            d2->name_len = 2;
536:                            d2->rec_len = i->sb->s_blocksize - 12;
537:                            d2->file_type = 0;          /* EXT2_FT_DIR not used */
538:                            i->i_nlink++;
539:                            i->i_size = i->sb->s_blocksize;
540:                            i->i_blocks = dir->sb->s_blocksize / 512;
541:                            i->dirty = 1;
542:
543:                            dir->i_mtime = CURRENT_TIME;
544:                            dir->i_ctime = CURRENT_TIME;
545:                            dir->i_nlink++;
546:                            dir->dirty = 1;
547:
548:                            bwrite(buf);
549:                            bwrite(buf2);
550:                            iput(i);
551:                            inode_unlock(dir);
552:                            return 0;
553:    }
554:
555: int ext2_mknod(struct inode *dir, char *name, __mode_t mode, __dev_t dev)
556: {
557:     struct buffer *buf;
558:     struct inode *i;
559:     struct ext2_dir_entry_2 *d;
560:     char c;
561:     int n;
562:
563:     inode_lock(dir);
564:
565:     if (!(i = ialloc(dir->sb, mode & S_IFMT))) {
566:         inode_unlock(dir);
567:         return -ENOSPC;
568:     }
569:
570:     if (!(buf = add_dir_entry(dir, &d, name))) {
571:         i->i_nlink = 0;
572:         iput(i);
573:         inode_unlock(dir);
574:         return -ENOSPC;
575:     }
576:
577:     d->inode = i->inode;
578:     d->name_len = strlen(name);
579:     /* strcpy() can't be used here because it places a trailing NULL */
580:     for (n = 0; n < NAME_MAX; n++) {

```


fs/ext2/namei.c

Page 10/13

```

581:         if((c = name[n])) {
582:             d->name[n] = c;
583:             continue;
584:         }
585:         break;
586:     }
587:
588:     i->i_mode = (mode & ~current->umask) & ~S_IFMT;
589:     i->i_uid = current->euid;
590:     i->i_gid = current->egid;
591:     i->i_nlink = 1;
592:     i->dev = dir->dev;
593:     i->count = 1;
594:     i->dirty = 1;
595:
596:     switch(mode & S_IFMT) {
597:         case S_IFCHR:
598:             i->fsop = &def_chr_fsop;
599:             i->rdev = dev;
600:             i->i_mode |= S_IFCHR;
601:             d->file_type = 0;          /* EXT2_FT_CHRDEV not used */
602:             break;
603:         case S_IFBLK:
604:             i->fsop = &def_blk_fsop;
605:             i->rdev = dev;
606:             i->i_mode |= S_IFBLK;
607:             d->file_type = 0;          /* EXT2_FT_BLKDEV not used */
608:             break;
609:         case S_IFIFO:
610:             i->fsop = &pipefs_fsop;
611:             i->i_mode |= S_IFIFO;
612:             /* it's a union so we need to clear pipefs_i */
613:             memset_b(&i->u.pipefs, NULL, sizeof(struct pipefs_inode)
);
614:             d->file_type = 0;          /* EXT2_FT_FIFO not used */
615:             break;
616:     }
617:
618:     dir->i_mtime = CURRENT_TIME;
619:     dir->i_ctime = CURRENT_TIME;
620:     dir->dirty = 1;
621:
622:     bwrite(buf);
623:     iput(i);
624:     inode_unlock(dir);
625:     return 0;
626: }
627:
628: int ext2_create(struct inode *dir, char *name, __mode_t mode, struct inode **i_r
es)
629: {
630:     struct buffer *buf;
631:     struct inode *i;
632:     struct ext2_dir_entry_2 *d;
633:     char c;
634:     int n;
635:
636:     if(IS_RDONLY_FS(dir)) {
637:         return -EROFS;
638:     }
639:
640:     inode_lock(dir);
641:
642:     if(!(i = ialloc(dir->sb, S_IFREG))) {
643:         inode_unlock(dir);
644:         return -ENOSPC;
645:     }

```

fs/ext2/namei.c

Page 11/13

```

646:
647:     if (!(buf = add_dir_entry(dir, &d, name))) {
648:         i->i_nlink = 0;
649:         iput(i);
650:         inode_unlock(dir);
651:         return -ENOSPC;
652:     }
653:
654:     d->inode = i->inode;
655:     d->name_len = strlen(name);
656:     /* strcpy() can't be used here because it places a trailing NULL */
657:     for (n = 0; n < NAME_MAX; n++) {
658:         if ((c = name[n])) {
659:             d->name[n] = c;
660:             continue;
661:         }
662:         break;
663:     }
664:     d->file_type = 0;      /* EXT2_FT_REG_FILE not used */
665:
666:     i->i_mode = (mode & ~current->umask) & ~S_IFMT;
667:     i->i_mode |= S_IFREG;
668:     i->i_uid = current->euid;
669:     i->i_gid = current->egid;
670:     i->i_nlink = 1;
671:     i->i_blocks = 0;
672:     i->dev = dir->dev;
673:     i->fsop = &ext2_file_fsop;
674:     i->count = 1;
675:     i->dirty = 1;
676:
677:     i->u.ext2.i_dtime = 0;
678:
679:     dir->i_mtime = CURRENT_TIME;
680:     dir->i_ctime = CURRENT_TIME;
681:     dir->dirty = 1;
682:
683:     *i_res = i;
684:     bwrite(buf);
685:     inode_unlock(dir);
686:     return 0;
687: }
688:
689: int ext2_rename(struct inode *i_old, struct inode *dir_old, struct inode *i_new,
struct inode *dir_new, char *oldpath, char *newpath)
690: {
691:     struct buffer *buf_old, *buf_new;
692:     struct ext2_dir_entry_2 *d_old, *d_new;
693:     char c;
694:     int n, errno;
695:
696:     errno = 0;
697:
698:     if (is_subdir(dir_new, i_old)) {
699:         return -EINVAL;
700:     }
701:
702:     inode_lock(i_old);
703:     inode_lock(dir_old);
704:     if (dir_old != dir_new) {
705:         inode_lock(dir_new);
706:     }
707:
708:     if (!(buf_old = find_dir_entry(dir_old, i_old, &d_old, oldpath))) {
709:         errno = -ENOENT;
710:         goto end;
711:     }

```

fs/ext2/namei.c

Page 12/13

```

712:         if(dir_old == dir_new) {
713:             /* free that buffer now to not block buf_new */
714:             brelse(buf_old);
715:             buf_old = NULL;
716:         }
717:
718:         if(i_new) {
719:             if(S_ISDIR(i_old->i_mode)) {
720:                 if(!is_dir_empty(i_new)) {
721:                     if(buf_old) {
722:                         brelse(buf_old);
723:                     }
724:                     errno = -ENOTEMPTY;
725:                     goto end;
726:                 }
727:             }
728:             if(!(buf_new = find_dir_entry(dir_new, i_new, &d_new, newpath)))
{
729:                 if(buf_old) {
730:                     brelse(buf_old);
731:                 }
732:                 errno = -ENOENT;
733:                 goto end;
734:             }
735:         } else {
736:             if(!(buf_new = add_dir_entry(dir_new, &d_new, newpath))) {
737:                 if(buf_old) {
738:                     brelse(buf_old);
739:                 }
740:                 errno = -ENOSPC;
741:                 goto end;
742:             }
743:             if(S_ISDIR(i_old->i_mode)) {
744:                 dir_old->i_nlink--;
745:                 dir_new->i_nlink++;
746:             }
747:         }
748:         if(i_new) {
749:             i_new->i_nlink--;
750:         } else {
751:             i_new = i_old;
752:             d_new->name_len = strlen(newpath);
753:             /* strcpy() can't be used here because it places a trailing NULL
*/
754:             for(n = 0; n < NAME_MAX; n++) {
755:                 if((c = newpath[n])) {
756:                     d_new->name[n] = c;
757:                     continue;
758:                 }
759:                 break;
760:             }
761:         }
762:
763:         d_new->inode = i_old->inode;
764:         dir_new->i_mtime = CURRENT_TIME;
765:         dir_new->i_ctime = CURRENT_TIME;
766:         i_new->dirty = 1;
767:         dir_new->dirty = 1;
768:
769:         dir_old->i_mtime = CURRENT_TIME;
770:         dir_old->i_ctime = CURRENT_TIME;
771:         i_old->dirty = 1;
772:         dir_old->dirty = 1;
773:         bwrite(buf_new);
774:
775:         if(!buf_old) {
776:             if(!(buf_old = find_dir_entry(dir_old, i_old, &d_old, oldpath)))

```

fs/ext2/namei.c

Page 13/13

```
{
777:             errno = -ENOENT;
778:             goto end;
779:         }
780:     }
781:     d_old->inode = 0;
782:     bwrite(buf_old);
783:
784:     /* update the parent directory */
785:     if(S_ISDIR(i_old->i_mode)) {
786:         buf_new = find_dir_entry(i_old, dir_old, &d_new, "..");
787:         if(buf_new) {
788:             d_new->inode = dir_new->inode;
789:             bwrite(buf_new);
790:         }
791:     }
792:
793: end:
794:     inode_unlock(i_old);
795:     inode_unlock(dir_old);
796:     inode_unlock(dir_new);
797:     return errno;
798: }
```

fs/ext2/super.c

Page 1/4

```

1: /*
2:  * fiwix/fs/ext2/super.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/types.h>
10: #include <fiwix/errno.h>
11: #include <fiwix/fs.h>
12: #include <fiwix/filesystems.h>
13: #include <fiwix/fs_ext2.h>
14: #include <fiwix/buffer.h>
15: #include <fiwix/sched.h>
16: #include <fiwix/stdio.h>
17: #include <fiwix/string.h>
18:
19: struct fs_operations ext2_fsop = {
20:     FSOP_REQUIRES_DEV,
21:     NULL,
22:
23:     NULL,          /* open */
24:     NULL,          /* close */
25:     NULL,          /* read */
26:     NULL,          /* write */
27:     NULL,          /* ioctl */
28:     NULL,          /* lseek */
29:     NULL,          /* readdir */
30:     NULL,          /* mmap */
31:     NULL,          /* select */
32:
33:     NULL,          /* readlink */
34:     NULL,          /* followlink */
35:     NULL,          /* bmap */
36:     NULL,          /* lookup */
37:     NULL,          /* rmdir */
38:     NULL,          /* link */
39:     NULL,          /* unlink */
40:     NULL,          /* symlink */
41:     NULL,          /* mkdir */
42:     NULL,          /* mknod */
43:     NULL,          /* truncate */
44:     NULL,          /* create */
45:     NULL,          /* rename */
46:
47:     NULL,          /* read_block */
48:     NULL,          /* write_block */
49:
50:     ext2_read_inode,
51:     ext2_write_inode,
52:     ext2_ialloc,
53:     ext2_ifree,
54:     ext2_statfs,
55:     ext2_read_superblock,
56:     ext2_remount_fs,
57:     ext2_write_superblock,
58:     ext2_release_superblock
59: };
60:
61: static void check_superblock(struct ext2_super_block *sb)
62: {
63:     if(!(sb->s_state & EXT2_VALID_FS)) {
64:         printk("WARNING: filesystem unchecked, fsck recommended.\n");
65:     } else if((sb->s_state & EXT2_ERROR_FS)) {
66:         printk("WARNING: filesystem contains errors, fsck recommended.\n
");

```

fs/ext2/super.c

Page 2/4

```

67:         } else if (sb->s_max_mnt_count >= 0 && sb->s_mnt_count >= (unsigned short
int) sb->s_max_mnt_count) {
68:             printk("WARNING: maximal mount count reached, fsck recommended.\n"
n");
69:         } else if (sb->s_checkinterval && (sb->s_lastcheck + sb->s_checkinterval
<= CURRENT_TIME)) {
70:             printk("WARNING: checktime reached, fsck recommended.\n");
71:         }
72:     }
73:
74: void ext2_statfs(struct superblock *sb, struct statfs *statfsbuf)
75: {
76:     statfsbuf->f_type = EXT2_SUPER_MAGIC;
77:     statfsbuf->f_bsize = sb->s_blocksize;
78:     statfsbuf->f_blocks = sb->u.ext2.sb.s_blocks_count;
79:     statfsbuf->f_bfree = sb->u.ext2.sb.s_free_blocks_count;
80:     if (statfsbuf->f_bfree >= sb->u.ext2.sb.s_r_blocks_count) {
81:         statfsbuf->f_bavail = statfsbuf->f_bfree - sb->u.ext2.sb.s_r_blo
cks_count;
82:     } else {
83:         statfsbuf->f_bavail = 0;
84:     }
85:     statfsbuf->f_files = sb->u.ext2.sb.s_inodes_count;
86:     statfsbuf->f_ffree = sb->u.ext2.sb.s_free_inodes_count;
87:     /* statfsbuf->f_fsid = ? */
88:     statfsbuf->f_namelen = EXT2_NAME_LEN;
89: }
90:
91: int ext2_read_superblock(__dev_t dev, struct superblock *sb)
92: {
93:     struct buffer *buf;
94:     struct ext2_super_block *ext2sb;
95:
96:     superblock_lock(sb);
97:     if (!(buf = bread(dev, SUPERBLOCK, BLKSIZE_1K))) {
98:         printk("WARNING: %s(): I/O error on device %d,%d.\n", __FUNCTION
__, MAJOR(dev), MINOR(dev));
99:         superblock_unlock(sb);
100:        return -EIO;
101:    }
102:
103:    ext2sb = (struct ext2_super_block *)buf->data;
104:    if (ext2sb->s_magic != EXT2_SUPER_MAGIC) {
105:        printk("WARNING: %s(): invalid filesystem type or bad superblock
on device %d,%d.\n", __FUNCTION__, MAJOR(dev), MINOR(dev));
106:        superblock_unlock(sb);
107:        brelse(buf);
108:        return -EINVAL;
109:    }
110:
111:    if (ext2sb->s_minor_rev_level || ext2sb->s_rev_level) {
112:        printk("WARNING: %s(): unsupported ext2 filesystem revision.\n",
__FUNCTION__);
113:        printk("Only revision 0 (original without features) is supported
.\n");
114:        superblock_unlock(sb);
115:        brelse(buf);
116:        return -EINVAL;
117:    }
118:
119:    sb->dev = dev;
120:    sb->fsop = &ext2_fsop;
121:    sb->s_blocksize = EXT2_MIN_BLOCK_SIZE << ext2sb->s_log_block_size;
122:    memcpy_b(&sb->u.ext2.sb, ext2sb, sizeof(struct ext2_super_block));
123:    EXT2_DESC_PER_BLOCK(sb) = sb->s_blocksize / sizeof(struct ext2_group_des
c);
124:    sb->u.ext2.block_groups = 1 + (ext2sb->s_blocks_count - 1) / ext2sb->s_b

```

fs/ext2/super.c

Page 3/4

```

locks_per_group;
125:
126:         if(!(sb->root = iget(sb, EXT2_ROOT_INO))) {
127:             printk("WARNING: %s(): unable to get root inode.\n", __FUNCTION_
_);
128:             superblock_unlock(sb);
129:             brelse(buf);
130:             return -EINVAL;
131:         }
132:
133:         check_superblock(ext2sb);
134:         if(!(sb->flags & MS_RDONLY)) {
135:             sb->u.ext2.sb.s_state &= ~EXT2_VALID_FS;
136:             sb->u.ext2.sb.s_mnt_count++;
137:             sb->u.ext2.sb.s_mtime = CURRENT_TIME;
138:             memcpy_b(buf->data, &sb->u.ext2.sb, sizeof(struct ext2_super_blo
ck));
139:             bwrite(buf);
140:         } else {
141:             brelse(buf);
142:         }
143:         superblock_unlock(sb);
144:         return 0;
145:     }
146:
147: int ext2_remount_fs(struct superblock *sb, int flags)
148: {
149:     struct buffer *buf;
150:     struct ext2_super_block *ext2sb;
151:
152:     if((flags & MS_RDONLY) == (sb->flags & MS_RDONLY)) {
153:         return 0;
154:     }
155:
156:     superblock_lock(sb);
157:     if(!(buf = bread(sb->dev, SUPERBLOCK, BLKSIZE_1K))) {
158:         superblock_unlock(sb);
159:         return -EIO;
160:     }
161:     ext2sb = (struct ext2_super_block *)buf->data;
162:
163:     if(flags & MS_RDONLY) {
164:         /* switching from RW to RO */
165:         sb->u.ext2.sb.s_state |= EXT2_VALID_FS;
166:         ext2sb->s_state |= EXT2_VALID_FS;
167:     } else {
168:         /* switching from RO to RW */
169:         check_superblock(ext2sb);
170:         memcpy_b(&sb->u.ext2.sb, ext2sb, sizeof(struct ext2_super_block)
);
171:         sb->u.ext2.sb.s_state &= ~EXT2_VALID_FS;
172:         sb->u.ext2.sb.s_mnt_count++;
173:         sb->u.ext2.sb.s_mtime = CURRENT_TIME;
174:         ext2sb->s_state &= ~EXT2_VALID_FS;
175:     }
176:
177:     sb->dirty = 1;
178:     superblock_unlock(sb);
179:     bwrite(buf);
180:     return 0;
181: }
182:
183: int ext2_write_superblock(struct superblock *sb)
184: {
185:     struct buffer *buf;
186:
187:     superblock_lock(sb);

```

fs/ext2/super.c

Page 4/4

```
188:         if(!(buf = bread(sb->dev, SUPERBLOCK, BLKSIZE_1K))) {
189:             superblock_unlock(sb);
190:             return -EIO;
191:         }
192:
193:         memcpy_b(buf->data, &sb->u.ext2.sb, sizeof(struct ext2_super_block));
194:         sb->dirty = 0;
195:         superblock_unlock(sb);
196:         bwrite(buf);
197:         return 0;
198:     }
199:
200: void ext2_release_superblock(struct superblock *sb)
201: {
202:     if(sb->flags & MS_RDONLY) {
203:         return;
204:     }
205:
206:     superblock_lock(sb);
207:
208:     sb->u.ext2.sb.s_state |= EXT2_VALID_FS;
209:     sb->dirty = 1;
210:
211:     superblock_unlock(sb);
212: }
213:
214: int ext2_init(void)
215: {
216:     return register_filesystem("ext2", &ext2_fsop);
217: }
```


fs/ext2/symlink.c

Page 1/3

```

1: /*
2:  * fiwix/fs/ext2/symlink.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/errno.h>
10: #include <fiwix/buffer.h>
11: #include <fiwix/fs.h>
12: #include <fiwix/filesystems.h>
13: #include <fiwix/stat.h>
14: #include <fiwix/mm.h>
15: #include <fiwix/stdio.h>
16: #include <fiwix/string.h>
17:
18: struct fs_operations ext2_symlink_fsop = {
19:     0,
20:     0,
21:
22:     NULL,          /* open */
23:     NULL,          /* close */
24:     NULL,          /* read */
25:     NULL,          /* write */
26:     NULL,          /* ioctl */
27:     NULL,          /* lseek */
28:     NULL,          /* readdir */
29:     NULL,          /* mmap */
30:     NULL,          /* select */
31:
32:     ext2_readlink,
33:     ext2_followlink,
34:     NULL,          /* bmap */
35:     NULL,          /* lookup */
36:     NULL,          /* rmdir */
37:     NULL,          /* link */
38:     NULL,          /* unlink */
39:     NULL,          /* symlink */
40:     NULL,          /* mkdir */
41:     NULL,          /* mknod */
42:     NULL,          /* truncate */
43:     NULL,          /* create */
44:     NULL,          /* rename */
45:
46:     NULL,          /* read_block */
47:     NULL,          /* write_block */
48:
49:     NULL,          /* read_inode */
50:     NULL,          /* write_inode */
51:     NULL,          /* ialloc */
52:     NULL,          /* ifree */
53:     NULL,          /* statfs */
54:     NULL,          /* read_superblock */
55:     NULL,          /* remount_fs */
56:     NULL,          /* write_superblock */
57:     NULL,          /* release_superblock */
58: };
59:
60: int ext2_readlink(struct inode *i, char *buffer, __size_t count)
61: {
62:     __u32 blksize;
63:     struct buffer *buf;
64:
65:     if(!S_ISLNK(i->i_mode)) {
66:         printk("%s(): Oops, inode '%d' is not a symlink (!?)\n", __FUNC
TION__, i->inode);

```

fs/ext2/symlink.c

Page 2/3

```

67:         return 0;
68:     }
69:
70:     inode_lock(i);
71:     blksize = i->sb->s_blocksize;
72:     count = MIN(count, i->i_size);
73:     if(!count) {
74:         inode_unlock(i);
75:         return 0;
76:     }
77:     count = MIN(count, blksize);
78:     if(i->i_blocks) { /* slow symlink */
79:         if(!(buf = bread(i->dev, i->u.ext2.i_data[0], blksize))) {
80:             inode_unlock(i);
81:             return -EIO;
82:         }
83:         memcpy_b(buffer, buf->data, count);
84:         brelse(buf);
85:     } else { /* fast symlink */
86:         memcpy_b(buffer, (char *)i->u.ext2.i_data, count);
87:     }
88:     buffer[count] = NULL;
89:     inode_unlock(i);
90:     return count;
91: }
92:
93: int ext2_followlink(struct inode *dir, struct inode *i, struct inode **i_res)
94: {
95:     struct buffer *buf;
96:     char *name;
97:     __ino_t errno;
98:
99:     if(!i) {
100:         return -ENOENT;
101:     }
102:
103:     if(!S_ISLNK(i->i_mode)) {
104:         printk("%s(): Oops, inode '%d' is not a symlink (!?)\n", __FUNC
TION__, i->inode);
105:         return 0;
106:     }
107:
108:     if(current->loopcnt > MAX_SYMLINKS) {
109:         printk("%s(): too many nested symbolic links!\n", __FUNCTION__);
110:         return -ELOOP;
111:     }
112:
113:     inode_lock(i);
114:     if(i->i_blocks) { /* slow symlink */
115:         if(!(buf = bread(i->dev, i->u.ext2.i_data[0], i->sb->s_blocksize
))) {
116:             inode_unlock(i);
117:             return -EIO;
118:         }
119:         name = buf->data;
120:     } else { /* fast symlink */
121:         buf = NULL;
122:         name = (char *)i->u.ext2.i_data;
123:     }
124:     inode_unlock(i);
125:
126:     current->loopcnt++;
127:     iput(i);
128:     if(buf) {
129:         brelse(buf);
130:     }
131:     errno = parse_namei(name, dir, i_res, NULL, FOLLOW_LINKS);

```

fs/ext2/symlink.c

Page 3/3

```
132:         current->loopcnt--;  
133:         return errno;  
134: }
```

fs/iso9660/dir.c

Page 1/4

```

1: /*
2:  * fiwix/fs/iso9660/dir.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/errno.h>
10: #include <fiwix/buffer.h>
11: #include <fiwix/fs.h>
12: #include <fiwix/filesystems.h>
13: #include <fiwix/stat.h>
14: #include <fiwix/dirent.h>
15: #include <fiwix/stdio.h>
16: #include <fiwix/string.h>
17:
18: struct fs_operations iso9660_dir_fsop = {
19:     0,
20:     0,
21:
22:     iso9660_dir_open,
23:     iso9660_dir_close,
24:     iso9660_dir_read,
25:     NULL, /* write */
26:     NULL, /* ioctl */
27:     NULL, /* lseek */
28:     iso9660_dir_readdir,
29:     NULL, /* mmap */
30:     NULL, /* select */
31:
32:     NULL, /* readlink */
33:     NULL, /* followlink */
34:     iso9660_bmap,
35:     iso9660_lookup,
36:     NULL, /* rmdir */
37:     NULL, /* link */
38:     NULL, /* unlink */
39:     NULL, /* symlink */
40:     NULL, /* mkdir */
41:     NULL, /* mknod */
42:     NULL, /* truncate */
43:     NULL, /* create */
44:     NULL, /* rename */
45:
46:     NULL, /* read_block */
47:     NULL, /* write_block */
48:
49:     NULL, /* read_inode */
50:     NULL, /* write_inode */
51:     NULL, /* ialloc */
52:     NULL, /* ifree */
53:     NULL, /* statsfs */
54:     NULL, /* read_superblock */
55:     NULL, /* remount_fs */
56:     NULL, /* write_superblock */
57:     NULL, /* release_superblock */
58: };
59:
60: int iso9660_dir_open(struct inode *i, struct fd *fd_table)
61: {
62:     fd_table->offset = 0;
63:     return 0;
64: }
65:
66: int iso9660_dir_close(struct inode *i, struct fd *fd_table)
67: {

```

fs/iso9660/dir.c

Page 2/4

```

68:         return 0;
69:     }
70:
71: int iso9660_dir_read(struct inode *i, struct fd *fd_table, char *buffer, __size_
t count)
72: {
73:     return -EISDIR;
74: }
75:
76: int iso9660_dir_readdir(struct inode *i, struct fd *fd_table, struct dirent *dir
ent, unsigned int count)
77: {
78:     __blk_t block;
79:     unsigned int doffset, offset;
80:     unsigned int size, dirent_len;
81:     struct iso9660_directory_record *d;
82:     int base_dirent_len;
83:     int blksize;
84:     struct buffer *buf;
85:     int nm_len;
86:     char nm_name[NAME_MAX + 1];
87:
88:     if(!(S_ISDIR(i->i_mode))) {
89:         return -EBADF;
90:     }
91:
92:     blksize = i->sb->s_blocksize;
93:     if(fd_table->offset > i->i_size) {
94:         fd_table->offset = i->i_size;
95:     }
96:
97:     base_dirent_len = sizeof(dirent->d_ino) + sizeof(dirent->d_off) + sizeof
(dirent->d_reclen);
98:     doffset = size = 0;
99:
100:    while(doffset < count) {
101:        if((block = bmap(i, fd_table->offset, FOR_READING)) < 0) {
102:            return block;
103:        }
104:        if(block) {
105:            if(!(buf = bread(i->dev, block, blksize))) {
106:                return -EIO;
107:            }
108:
109:            doffset = fd_table->offset;
110:            offset = fd_table->offset % blksize;
111:
112:            while(doffset < i->i_size && offset < blksize) {
113:                d = (struct iso9660_directory_record *) (buf->dat
a + offset);
114:                if(isonum_711(d->length)) {
115:                    dirent_len = (base_dirent_len + (isonum_
711(d->name_len) + 1)) + 3;
116:                    dirent_len &= ~3;        /* round up */
117:                    if((size + dirent_len) < count) {
118:                        dirent->d_ino = (block << ISO966
0_INODE_BITS) + (doffset & ISO9660_INODE_MASK);
119:                        dirent->d_off = doffset;
120:                        dirent->d_reclen = dirent_len;
121:                        if(isonum_711(d->name_len) == 1
&& d->name[0] == 0) {
122:                            dirent->d_name[0] = '.';
123:                            dirent->d_name[1] = NULL
;
124:                        } else if(isonum_711(d->name_len
) == 1 && d->name[0] == 1) {
125:                            dirent->d_name[0] = '.';

```

fs/iso9660/dir.c

Page 3/4

```

126:                                dirent->d_name[1] = '.';
127:                                dirent->d_name[2] = NULL;
;
128:                                dirent_len = 16;
129:                                dirent->d_reclen = 16;
130:                                if(i->u.iso9660.i_parent
) {
131:                                    dirent->d_ino =
i->u.iso9660.i_parent->inode;
132:                                } else {
133:                                    dirent->d_ino =
i->inode;
134:                                }
135:                                } else {
136:                                    nm_len = 0;
137:                                    if(i->sb->u.iso9660.rrip
) {
138:                                        nm_len = get_rri
p_filename(d, i, nm_name);
139:                                    }
140:                                    if(nm_len) {
141:                                        dirent->d_reclen
= (base_dirent_len + nm_len + 1) + 3;
142:                                        dirent->d_reclen
&= ~3; /* round up */
143:                                        dirent_len = dir
ent->d_reclen;
144:                                        if((size + diren
t_len) < count) {
145:                                            dirent->
d_name[nm_len] = NULL;
146:                                            memcpy_b
(dirent->d_name, nm_name, nm_len);
147:                                            } else {
148:                                                break;
149:                                            }
150:                                        } else {
151:                                            memcpy_b(dirent-
>d_name, d->name, isonum_711(d->name_len));
152:                                            dirent->d_name[i
sonum_711(d->name_len)] = NULL;
153:                                        }
154:                                        }
155:                                        if(!((char)d->flags[0] & ISO9660
_FILE_ISDIR)) {
156:                                            iso9660_cleanfilename(di
rent->d_name, isonum_711(d->name_len));
157:                                        }
158:                                        dirent = (struct dirent *)((char
*)dirent + dirent_len);
159:                                        size += dirent_len;
160:                                        } else {
161:                                            break;
162:                                        }
163:                                        doffset += isonum_711(d->length);
164:                                        offset += isonum_711(d->length);
165:                                        } else {
166:                                            doffset &= ~(blksize - 1);
167:                                            doffset += blksize;
168:                                            break;
169:                                        }
170:                                    }
171:                                    brelse(buf);
172:                                }
173:                                fd_table->offset = doffset;
174:                            }
175:                            return size;

```

```
176: }
```

fs/iso9660/file.c

```

1:  /*
2:  *  fiwix/fs/iso9660/file.c
3:  *
4:  *  Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  *  Distributed under the terms of the Fiwix License.
6:  */
7:
8:  #include <fiwix/types.h>
9:  #include <fiwix/errno.h>
10: #include <fiwix/buffer.h>
11: #include <fiwix/fs.h>
12: #include <fiwix/filesystems.h>
13: #include <fiwix/mm.h>
14: #include <fiwix/mman.h>
15: #include <fiwix/fcntl.h>
16: #include <fiwix/stdio.h>
17: #include <fiwix/string.h>
18:
19: struct fs_operations iso9660_file_fsop = {
20:     0,
21:     0,
22:
23:     iso9660_file_open,
24:     iso9660_file_close,
25:     file_read,
26:     NULL,                /* write */
27:     NULL,                /* ioctl */
28:     iso9660_file_lseek,
29:     NULL,                /* readdir */
30:     NULL,                /* mmap */
31:     NULL,                /* select */
32:
33:     NULL,                /* readlink */
34:     NULL,                /* followlink */
35:     iso9660_bmap,
36:     NULL,                /* lookup */
37:     NULL,                /* rmdir */
38:     NULL,                /* link */
39:     NULL,                /* unlink */
40:     NULL,                /* symlink */
41:     NULL,                /* mkdir */
42:     NULL,                /* mknod */
43:     NULL,                /* truncate */
44:     NULL,                /* create */
45:     NULL,                /* rename */
46:
47:     NULL,                /* read_block */
48:     NULL,                /* write_block */
49:
50:     NULL,                /* read_inode */
51:     NULL,                /* write_inode */
52:     NULL,                /* ialloc */
53:     NULL,                /* ifree */
54:     NULL,                /* statfs */
55:     NULL,                /* read_superblock */
56:     NULL,                /* remount_fs */
57:     NULL,                /* write_superblock */
58:     NULL,                /* release_superblock */
59: };
60:
61: int iso9660_file_open(struct inode *i, struct fd *fd_table)
62: {
63:     if(fd_table->flags & (O_WRONLY | O_RDWR | O_TRUNC | O_APPEND)) {
64:         return -ENOENT;
65:     }
66:     fd_table->offset = 0;
67:     return 0;

```



```
68: }
69:
70: int iso9660_file_close(struct inode *i, struct fd *fd_table)
71: {
72:     return 0;
73: }
74:
75: int iso9660_file_lseek(struct inode *i, __off_t offset)
76: {
77:     return offset;
78: }
```

fs/iso9660/inode.c

Page 1/3

```

1: /*
2:  * fiwix/fs/iso9660/inode.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/types.h>
10: #include <fiwix/errno.h>
11: #include <fiwix/fs.h>
12: #include <fiwix/filesystems.h>
13: #include <fiwix/fs_iso9660.h>
14: #include <fiwix/fs_pipe.h>
15: #include <fiwix/buffer.h>
16: #include <fiwix/stat.h>
17: #include <fiwix/mm.h>
18: #include <fiwix/sched.h>
19: #include <fiwix/stdio.h>
20: #include <fiwix/string.h>
21:
22: static int read_pathtable(struct inode *i)
23: {
24:     int n, offset, pt_len, pt_blk;
25:     struct iso9660_sb_info *sbi;
26:     struct iso9660_pathtable_record *ptr;
27:     struct buffer *buf;
28:
29:     sbi = (struct iso9660_sb_info *)&i->sb->u.iso9660;
30:     pt_len = isonum_733(sbi->sb->path_table_size);
31:     pt_blk = isonum_731(sbi->sb->type_l_path_table);
32:
33:     if(pt_len > PAGE_SIZE) {
34:         printk("WARNING: %s(): path table record size (%d) > 4096, not s
supported yet.\n", __FUNCTION__, pt_len);
35:         return -EINVAL;
36:     }
37:
38:     if(!(sbi->pathtable_raw = (void *)kmalloc())) {
39:         return -ENOMEM;
40:     }
41:     offset = 0;
42:     while(offset < pt_len) {
43:         if(!(buf = bread(i->dev, pt_blk++, BLKSIZE_2K))) {
44:             kfree((unsigned int)sbi->pathtable_raw);
45:             return -EIO;
46:         }
47:         memcpy_b(sbi->pathtable_raw + offset, (void *)buf->data, MIN(pt_
len - offset, BLKSIZE_2K));
48:         offset += MIN(pt_len - offset, BLKSIZE_2K);
49:         brelse(buf);
50:     }
51:
52:     /* allocate and count the number of records in the Path Table */
53:     offset = n = 0;
54:     if(!(sbi->pathtable = (struct iso9660_pathtable_record **)kmalloc())) {
55:         kfree((unsigned int)sbi->pathtable_raw);
56:         return -ENOMEM;
57:     }
58:     sbi->pathtable[n] = NULL;
59:     while(offset < pt_len) {
60:         ptr = (struct iso9660_pathtable_record *) (sbi->pathtable_raw + o
ffset);
61:         sbi->pathtable[++n] = ptr;
62:         offset += sizeof(struct iso9660_pathtable_record) + isonum_711(p
tr->length) + (isonum_711(ptr->length) & 1);
63:     }

```

fs/iso9660/inode.c

Page 2/3

```

64:         sbi->paths = n;
65:
66:         return 0;
67:     }
68:
69: static int get_parent_dir_size(struct superblock *sb, __blk_t extent)
70: {
71:     int n;
72:     struct iso9660_pathtable_record *ptr;
73:     __blk_t parent;
74:
75:     for(n = 0; n < sb->u.iso9660.paths; n++) {
76:         ptr = (struct iso9660_pathtable_record *)sb->u.iso9660.pathtable
[n];
77:         if(isonum_731(ptr->extent) == extent) {
78:
79:             parent = isonum_723(ptr->parent);
80:             ptr = (struct iso9660_pathtable_record *)sb->u.iso9660.p
athtable[parent];
81:             parent = isonum_731(ptr->extent);
82:             return parent;
83:         }
84:     }
85:     printk("WARNING: %s(): unable to locate extent '%d' in path table.\n", _
_FUNCTION__, extent);
86:     return 0;
87: }
88:
89: int iso9660_read_inode(struct inode *i)
90: {
91:     int errno;
92:     __u32 blksize;
93:     struct superblock *sb;
94:     struct iso9660_directory_record *d;
95:     struct buffer *buf;
96:     __blk_t dblock;
97:     __off_t doffset;
98:
99:     sb = (struct superblock *)i->sb;
100:    if(!sb->u.iso9660.pathtable) {
101:        if((errno = read_pathtable(i))) {
102:            return errno;
103:        }
104:    }
105:
106:    dblock = (i->inode & ~ISO9660_INODE_MASK) >> ISO9660_INODE_BITS;
107:    doffset = i->inode & ISO9660_INODE_MASK;
108:    blksize = i->sb->s_blocksize;
109:
110:    /* FIXME: it only looks in one directory block */
111:    if(!(buf = bread(i->dev, dblock, blksize))) {
112:        return -EIO;
113:    }
114:
115:    if(doffset >= blksize) {
116:        printk("WARNING: %s(): inode %d (dblock=%d, doffset=%d) not foun
d in directory entry.\n", _FUNCTION__, i->inode, dblock, doffset);
117:        brelse(buf);
118:        return -EIO;
119:    }
120:    d = (struct iso9660_directory_record *) (buf->data + doffset);
121:
122:    i->i_mode = S_IFREG;
123:    if((char)d->flags[0] & ISO9660_FILE_ISDIR) {
124:        i->i_mode = S_IFDIR;
125:    }
126:    if(!(char)d->flags[0] & ISO9660_FILE_HASOWNER) {

```

fs/iso9660/inode.c

Page 3/3

```

127:             i->i_mode |= S_IRUSR | S_IXUSR | S_IRGRP | S_IXGRP | S_IROTH | S
_IXOTH;
128:         }
129:         i->i_uid = 0;
130:         i->i_size = isonum_733(d->size);
131:         i->i_atime = isodate(d->date);
132:         i->i_ctime = isodate(d->date);
133:         i->i_mtime = isodate(d->date);
134:         i->i_gid = 0;
135:         i->i_nlink = 1;
136:         i->count = 1;
137:
138:         i->u.iso9660.i_extent = isonum_733(d->extent);
139:         check_rrip_inode(d, i);
140:         brelse(buf);
141:
142:         switch(i->i_mode & S_IFMT) {
143:             case S_IFCHR:
144:                 i->fsop = &def_chr_fsop;
145:                 break;
146:             case S_IFBLK:
147:                 i->fsop = &def_blk_fsop;
148:                 break;
149:             case S_IFIFO:
150:                 i->fsop = &pipefs_fsop;
151:                 /* it's a union so we need to clear pipefs_inode */
152:                 memset_b(&i->u.pipefs, NULL, sizeof(struct pipefs_inode)
);
153:                 break;
154:             case S_IFDIR:
155:                 i->fsop = &iso9660_dir_fsop;
156:                 i->i_nlink++;
157:                 break;
158:             case S_IFREG:
159:                 i->fsop = &iso9660_file_fsop;
160:                 break;
161:             case S_IFLNK:
162:                 i->fsop = &iso9660_symlink_fsop;
163:                 break;
164:             case S_IFSOCK:
165:                 i->fsop = NULL;
166:                 break;
167:             default:
168:                 PANIC("invalid inode (%d) mode %08o.\n", i->inode, i->i
mode);
169:         }
170:         return 0;
171:     }
172:
173:     int iso9660_bmap(struct inode *i, __off_t offset, int mode)
174:     {
175:         __blk_t block;
176:
177:         block = i->u.iso9660.i_extent + (offset / i->sb->s_blocksize);
178:         return block;
179:     }

```

fs/iso9660/Makefile

Page 1/1

```
1: # fiwix/fs/iso9660/Makefile
2: #
3: # Copyright 2018, Jordi Sanfeliu. All rights reserved.
4: # Distributed under the terms of the Fiwix License.
5: #
6:
7: .S.o:
8:     $(CC) -traditional -I$(INCLUDE) -c -o $@ $<
9: .c.o:
10:     $(CC) $(CFLAGS) -c -o $@ $<
11:
12: OBJS = inode.o super.o namei.o dir.o file.o rrip.o symlink.o
13:
14: iso9660:$ (OBJS)
15:     $(LD) $(LDFLAGS) -r $(OBJS) -o iso9660.o
16:
17: clean:
18:     rm -f *.o
19:
```

fs/iso9660/namei.c

Page 1/2

```

1: /*
2:  * fiwix/fs/iso9660/namei.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/filesystems.h>
11: #include <fiwix/fs_iso9660.h>
12: #include <fiwix/buffer.h>
13: #include <fiwix/stat.h>
14: #include <fiwix/mm.h>
15: #include <fiwix/errno.h>
16: #include <fiwix/stdio.h>
17: #include <fiwix/string.h>
18:
19: int iso9660_lookup(const char *name, struct inode *dir, struct inode **i_res)
20: {
21:     __blk_t dblock;
22:     __u32 blksize;
23:     int len, dnlen;
24:     unsigned int offset, doffset;
25:     struct buffer *buf;
26:     struct iso9660_directory_record *d;
27:     __ino_t inode;
28:     int nm_len;
29:     char *nm_name;
30:
31:     blksize = dir->sb->s_blocksize;
32:     inode = offset = 0;
33:     len = strlen(name);
34:
35:     while(offset < dir->i_size && !inode) {
36:         if((dblock = bmap(dir, offset, FOR_READING)) < 0) {
37:             return dblock;
38:         }
39:         if(dblock) {
40:             if(!(buf = bread(dir->dev, dblock, blksize))) {
41:                 return -EIO;
42:             }
43:             doffset = 0;
44:             do {
45:                 d = (struct iso9660_directory_record *) (buf->data
a + doffset);
46:                 if(isonum_711(d->length) == 0) {
47:                     break;
48:                 }
49:                 if(len == 1) {
50:                     if(name[0] == '.' && name[1] == NULL) {
51:                         if(isonum_711(d->name_len) == 1
&& d->name[0] == 0) {
52:                             inode = dir->inode;
53:                         }
54:                     }
55:                 }
56:                 if(len == 2) {
57:                     if(name[0] == '.' && name[1] == '.' && n
ame[2] == NULL) {
58:                         if(isonum_711(d->name_len) == 1
&& d->name[0] == 1) {
59:                             inode = dir->u.iso9660.i
_parent->inode;
60:                         }
61:                     }
62:                 }

```

fs/iso9660/namei.c

Page 2/2

```

63:         if(!(nm_name = (char *)kmalloc())) {
64:             return -ENOMEM;
65:         }
66:         nm_len = 0;
67:         if(dir->sb->u.iso9660.rrip) {
68:             nm_len = get_rrip_filename(d, dir, nm_na
me);
69:         }
70:         if(nm_len) {
71:             dnlen = nm_len;
72:         } else {
73:             dnlen = isonum_711(d->name_len);
74:             if(!((char)d->flags[0] & ISO9660_FILE_IS
DIR)) {
75:                 iso9660_cleanfilename(d->name, d
nlen);
76:                 dnlen = strlen(d->name);
77:             }
78:         }
79:         if(len == dnlen) {
80:             if(nm_len) {
81:                 if(strncmp(nm_name, name, dnlen)
== 0) {
82:                     inode = (dblock << ISO96
60_INODE_BITS) + (doffset & ISO9660_INODE_MASK);
83:                 }
84:             } else {
85:                 if(strncmp(d->name, name, dnlen)
== 0) {
86:                     inode = (dblock << ISO96
60_INODE_BITS) + (doffset & ISO9660_INODE_MASK);
87:                 }
88:             }
89:         }
90:         kfree((unsigned int)nm_name);
91:         doffset += isonum_711(d->length);
92:     } while((doffset < blksize) && (!inode));
93:     brelse(buf);
94:     offset += blksize;
95:     if(inode) {
96:         /*
97:          * This prevents a deadlock in iget() when
98:          * trying to lock '.' when 'dir' is the same
99:          * directory (ls -lai <tmp>).
100:        */
101:         if(inode == dir->inode) {
102:             *i_res = dir;
103:             return 0;
104:         }
105:
106:         if(!(*i_res = iget(dir->sb, inode))) {
107:             return -EACCES;
108:         }
109:         if(S_ISDIR((*i_res)->i_mode)) {
110:             if(!(*i_res)->u.iso9660.i_parent) {
111:                 (*i_res)->u.iso9660.i_parent = d
ir;
112:             }
113:         }
114:         iput(dir);
115:         return 0;
116:     }
117: }
118: }
119: iput(dir);
120: return -ENOENT;
121: }

```

fs/iso9660/rrip.c

Page 1/6

```

1: /*
2:  * fiwix/fs/iso9660/rrip.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/errno.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/stat.h>
12: #include <fiwix/filesystems.h>
13: #include <fiwix/buffer.h>
14: #include <fiwix/fs_iso9660.h>
15: #include <fiwix/mm.h>
16: #include <fiwix/stdio.h>
17: #include <fiwix/string.h>
18:
19: void check_rrip_inode(struct iso9660_directory_record *d, struct inode *i)
20: {
21:     unsigned int total_len;
22:     unsigned int len;
23:     unsigned int sig;
24:     int n, nm_len, rootflag;
25:     struct susp_rrip *rrip;
26:     unsigned int dev_h, dev_l;
27:     unsigned int ce_block, ce_offset, ce_size;
28:     struct buffer *buf;
29:     unsigned char *sue;
30:     int sl_len;
31:     struct rrip_sl_component *slc;
32:
33:     ce_block = ce_offset = ce_size = 0;
34:     buf = NULL;
35:     total_len = isonum_711(d->length);
36:     len = isonum_711(d->name_len);
37:     if(!(len % 2)) {
38:         len++;
39:     }
40:     sue = (unsigned char *)d->name;
41:     nm_len = 0;
42:
43: loop:
44:     if(ce_block && ce_size) {
45:         /* FIXME: it only looks in one directory block */
46:         if(!(buf = bread(i->dev, ce_block, i->sb->s_blocksize))) {
47:             return;
48:         }
49:         sue = (unsigned char *)buf->data + ce_offset;
50:         total_len = ce_size;
51:         len = 0;
52:     }
53:
54:     while(len < total_len) {
55:         rrip = (struct susp_rrip *) (sue + len);
56:         if(rrip->len == 0) {
57:             break;
58:         }
59:         sig = GET_SIG(rrip->signature[0], rrip->signature[1]);
60:         switch(sig) {
61:             case GET_SIG('S', 'P'):
62:                 if(rrip->u.sp.magic[0] != SP_MAGIC1 || rrip->u.s
p.magic[1] != SP_MAGIC2) {
63:                     if(ce_block) {
64:                         brelse(buf);
65:                     }
66:                     return;

```


fs/iso9660/rrip.c

Page 2/6

```

67:         }
68:         break;
69:     case GET_SIG('C', 'E'):
70:         if(ce_block) {
71:             brelse(buf);
72:         }
73:         ce_block = isonum_733(rrip->u.ce.block);
74:         ce_offset = isonum_733(rrip->u.ce.offset);
75:         ce_size = isonum_733(rrip->u.ce.size);
76:         goto loop;
77:         break;
78:     case GET_SIG('E', 'R'):
79:         i->sb->u.iso9660.rrip = 1;
80:         printk("ISO 9660 Extensions: ");
81:         for(n = 0; n < rrip->u.er.len_id; n++) {
82:             printk("%c", rrip->u.er.data[n]);
83:         }
84:         printk("\n");
85:         break;
86:     case GET_SIG('P', 'X'):
87:         i->i_mode = isonum_733(rrip->u.px.mode);
88:         i->i_nlink = isonum_733(rrip->u.px.nlink);
89:         i->i_uid = isonum_733(rrip->u.px.uid);
90:         i->i_gid = isonum_733(rrip->u.px.gid);
91:         break;
92:     case GET_SIG('P', 'N'):
93:         if(S_ISBLK(i->i_mode) || S_ISCHR(i->i_mode)) {
94:             dev_h = isonum_733(rrip->u.pn.dev_h);
95:             dev_l = isonum_733(rrip->u.pn.dev_l);
96:             i->rdev = MKDEV(dev_h, dev_l);
97:         }
98:         break;
99:     case GET_SIG('S', 'L'):
100:        sl_len = rootflag = 0;
101:        slc = (struct rrip_sl_component *)&rrip->u.sl.ar
ea;
102:        while(sl_len < (rrip->len - 5)) {
103:            if(sl_len && !rootflag) {
104:                nm_len++;
105:            }
106:            rootflag = 0;
107:            switch(slc->flags & 0xE) {
108:                case 0:
109:                    nm_len += slc->len;
110:                    break;
111:                case SL_CURRENT:
112:                    nm_len += 1;
113:                    break;
114:                case SL_PARENT:
115:                    nm_len += 2;
116:                    break;
117:                case SL_ROOT:
118:                    nm_len += 1;
119:                    rootflag = 1;
120:                    break;
121:                default:
122:                    printk("WARNING: %s(): u
nsupported RRIP SL flags %d.\n", __FUNCTION__, slc->flags & 0xE);
123:            }
124:            slc = (struct rrip_sl_component *)(((cha
r *)slc) + slc->len + sizeof(struct rrip_sl_component));
125:            sl_len += slc->len + sizeof(struct rrip_
sl_component);
126:        }
127:        i->i_size = nm_len;
128:        break;
129:     case GET_SIG('T', 'F'):

```

fs/iso9660/rrip.c

Page 3/6

```

130:         n = 0;
131:         if(rrip->u.tf.flags & TF_CREATION) {
132:             i->i_ctime = isodate(rrip->u.tf.times[n+
+] .time);
133:         }
134:         if(rrip->u.tf.flags & TF_MODIFY) {
135:             i->i_mtime = isodate(rrip->u.tf.times[n+
+] .time);
136:         }
137:         if(rrip->u.tf.flags & TF_ACCESS) {
138:             i->i_atime = isodate(rrip->u.tf.times[n+
+] .time);
139:         }
140:         if(rrip->u.tf.flags & TF_ATTRIBUTES) {
141:             i->i_ctime = isodate(rrip->u.tf.times[n+
+] .time);
142:         }
143:         break;
144:     }
145:     len += rrip->len;
146: }
147: if(ce_block) {
148:     brelse(buf);
149: }
150: }
151:
152: int get_rrip_filename(struct iso9660_directory_record *d, struct inode *i, char
*name)
153: {
154:     unsigned int total_len;
155:     unsigned int len;
156:     unsigned int sig;
157:     int nm_len;
158:     struct susp_rrip *rrip;
159:     unsigned int ce_block, ce_offset, ce_size;
160:     struct buffer *buf;
161:     unsigned char *sue;
162:
163:     ce_block = ce_offset = ce_size = 0;
164:     buf = NULL;
165:     total_len = isonum_711(d->length);
166:     len = isonum_711(d->name_len);
167:     if(!(len % 2)) {
168:         len++;
169:     }
170:     sue = (unsigned char *)d->name;
171:     nm_len = 0;
172:
173: loop:
174:     if(ce_block && ce_size) {
175:         /* FIXME: it only looks in one directory block */
176:         if(!(buf = bread(i->dev, ce_block, i->sb->s_blocksize))) {
177:             return 0;
178:         }
179:         sue = (unsigned char *)buf->data + ce_offset;
180:         total_len = ce_size;
181:         len = 0;
182:     }
183:
184:     while(len < total_len) {
185:         rrip = (struct susp_rrip *) (sue + len);
186:         if(rrip->len == 0) {
187:             break;
188:         }
189:         sig = GET_SIG(rrip->signature[0], rrip->signature[1]);
190:         switch(sig) {
191:             case GET_SIG('S', 'P'):
```

fs/iso9660/rrip.c

Page 4/6

```

192:                                if(rrip->u.sp.magic[0] != SP_MAGIC1 || rrip->u.s
p.magic[1] != SP_MAGIC2) {
193:                                    if(ce_block) {
194:                                        brelse(buf);
195:                                    }
196:                                    return 0;
197:                                }
198:                                break;
199:                                case GET_SIG('C', 'E'):
200:                                    if(ce_block) {
201:                                        brelse(buf);
202:                                    }
203:                                    ce_block = isonum_733(rrip->u.ce.block);
204:                                    ce_offset = isonum_733(rrip->u.ce.offset);
205:                                    ce_size = isonum_733(rrip->u.ce.size);
206:                                    goto loop;
207:                                case GET_SIG('N', 'M'):
208:                                    if(rrip->u.nm.flags) { /* FIXME: & ~(NM_CONTINUE
| NM_CURRENT | NM_PARENT)) { */
209:                                        printk("WARNING: %s(): unsupported NM fl
ag settings (%d).\n", __FUNCTION__, rrip->u.nm.flags);
210:                                        if(ce_block) {
211:                                            brelse(buf);
212:                                        }
213:                                        return 0;
214:                                    }
215:                                    nm_len = rrip->len - 5;
216:                                    memcpy_b(name, rrip->u.nm.name, nm_len);
217:                                    name[nm_len] = NULL;
218:                                    break;
219:                                }
220:                                len += rrip->len;
221:                            }
222:                            if(ce_block) {
223:                                brelse(buf);
224:                            }
225:                            return nm_len;
226:    }
227:
228: int get_rrip_symlink(struct inode *i, char *name)
229: {
230:     unsigned int total_len;
231:     unsigned int len;
232:     unsigned int sig;
233:     int nm_len;
234:     struct susp_rrip *rrip;
235:     unsigned int ce_block, ce_offset, ce_size;
236:     struct buffer *buf;
237:     struct buffer *buf2;
238:     unsigned char *sue;
239:     struct iso9660_directory_record *d;
240:     __blk_t dblock;
241:     __off_t doffset;
242:     int sl_len, rootflag;
243:     struct rrip_sl_component *slc;
244:
245:     dblock = (i->inode & ~ISO9660_INODE_MASK) >> ISO9660_INODE_BITS;
246:     doffset = i->inode & ISO9660_INODE_MASK;
247:     /* FIXME: it only looks in one directory block */
248:     if(!(buf = bread(i->dev, dblock, i->sb->s_blocksize))) {
249:         return -EIO;
250:     }
251:     d = (struct iso9660_directory_record *) (buf->data + doffset);
252:
253:     ce_block = ce_offset = ce_size = 0;
254:     buf2 = NULL;
255:     total_len = isonum_711(d->length);

```

fs/iso9660/rrip.c

Page 5/6

```

256:     len = isonum_711(d->name_len);
257:     if(!(len % 2)) {
258:         len++;
259:     }
260:     sue = (unsigned char *)d->name;
261:     nm_len = 0;
262:
263: loop:
264:     if(ce_block && ce_size) {
265:         /* FIXME: it only looks in one directory block */
266:         if(!(buf2 = bread(i->dev, ce_block, i->sb->s_blocksize))) {
267:             return 0;
268:         }
269:         sue = (unsigned char *)buf2->data + ce_offset;
270:         total_len = ce_size;
271:         len = 0;
272:     }
273:
274:     while(len < total_len) {
275:         rrip = (struct susp_rrip *) (sue + len);
276:         if(rrip->len == 0) {
277:             break;
278:         }
279:         sig = GET_SIG(rrip->signature[0], rrip->signature[1]);
280:         switch(sig) {
281:             case GET_SIG('S', 'P'):
282:                 if(rrip->u.sp.magic[0] != SP_MAGIC1 || rrip->u.s
p.magic[1] != SP_MAGIC2) {
283:                     if(ce_block) {
284:                         brelse(buf2);
285:                     }
286:                     return 0;
287:                 }
288:                 break;
289:             case GET_SIG('C', 'E'):
290:                 if(ce_block) {
291:                     brelse(buf2);
292:                 }
293:                 ce_block = isonum_733(rrip->u.ce.block);
294:                 ce_offset = isonum_733(rrip->u.ce.offset);
295:                 ce_size = isonum_733(rrip->u.ce.size);
296:                 goto loop;
297:             case GET_SIG('S', 'L'):
298:                 sl_len = rootflag = 0;
299:                 slc = (struct rrip_sl_component *)&rrip->u.sl.ar
ea;
300:                 while(sl_len < (rrip->len - 5)) {
301:                     if(sl_len && !rootflag) {
302:                         strcat(name, "/");
303:                         nm_len++;
304:                     }
305:                     rootflag = 0;
306:                     switch(slc->flags & 0xE) {
307:                         case 0:
308:                             nm_len += slc->len;
309:                             strncat(name, slc->name,
slc->len);
310:                             break;
311:                         case SL_CURRENT:
312:                             nm_len += 1;
313:                             strcat(name, ".");
314:                             break;
315:                         case SL_PARENT:
316:                             nm_len += 2;
317:                             strcat(name, "..");
318:                             break;
319:                         case SL_ROOT:

```

fs/iso9660/rrip.c

Page 6/6

```
320:                                     nm_len += 1;
321:                                     rootflag = 1;
322:                                     strcat(name, "/");
323:                                     break;
324:                                     default:
325:                                     printk("WARNING: %s(): u
nsupported RRIP SL flags %d.\n", __FUNCTION__, slc->flags & 0xE);
326:                                     }
327:                                     slc = (struct rrip_sl_component *)(((cha
r *)slc) + slc->len + sizeof(struct rrip_sl_component));
328:                                     sl_len += slc->len + sizeof(struct rrip_
sl_component);
329:                                     }
330:                                     name[nm_len] = NULL;
331:                                     break;
332:                                     }
333:                                     len += rrip->len;
334:                                     }
335:                                     if(ce_block) {
336:                                     brelse(buf2);
337:                                     }
338:                                     brelse(buf);
339:                                     return nm_len;
340: }
```

fs/iso9660/super.c

Page 1/4

```

1: /*
2:  * fiwix/fs/iso9660/super.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/errno.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/filesystems.h>
12: #include <fiwix/fs_iso9660.h>
13: #include <fiwix/buffer.h>
14: #include <fiwix/time.h>
15: #include <fiwix/sched.h>
16: #include <fiwix/mm.h>
17: #include <fiwix/stdio.h>
18: #include <fiwix/string.h>
19:
20: struct fs_operations iso9660_fsop = {
21:     FSOP_REQUIRES_DEV,
22:     NULL,
23:
24:     NULL,                /* open */
25:     NULL,                /* close */
26:     NULL,                /* read */
27:     NULL,                /* write */
28:     NULL,                /* ioctl */
29:     NULL,                /* lseek */
30:     NULL,                /* readdir */
31:     NULL,                /* mmap */
32:     NULL,                /* select */
33:
34:     NULL,                /* readlink */
35:     NULL,                /* followlink */
36:     NULL,                /* bmap */
37:     NULL,                /* lookup */
38:     NULL,                /* rmdir */
39:     NULL,                /* link */
40:     NULL,                /* unlink */
41:     NULL,                /* symlink */
42:     NULL,                /* mkdir */
43:     NULL,                /* mknod */
44:     NULL,                /* truncate */
45:     NULL,                /* create */
46:     NULL,                /* rename */
47:
48:     NULL,                /* read_block */
49:     NULL,                /* write_block */
50:
51:     iso9660_read_inode,
52:     NULL,                /* write_inode */
53:     NULL,                /* ialloc */
54:     NULL,                /* ifree */
55:     iso9660_statfs,
56:     iso9660_read_superblock,
57:     NULL,                /* remount_fs */
58:     NULL,                /* write_superblock */
59:     iso9660_release_superblock
60: };
61:
62: int isonum_711(char *str)
63: {
64:     unsigned char *le;
65:
66:     le = (unsigned char *)str;
67:     return le[0];

```

fs/iso9660/super.c

Page 2/4

```

68: }
69:
70: /* return a 16bit little-endian number */
71: int isonum_723(char *str)
72: {
73:     unsigned char *le;
74:
75:     le = (unsigned char *)str;
76:     return le[0] | (le[1] << 8);
77: }
78:
79: /* return a 32bit little-endian number */
80: int isonum_731(char *str)
81: {
82:     unsigned char *le;
83:
84:     le = (unsigned char *)str;
85:     return le[0] | (le[1] << 8) | (le[2] << 16) | (le[3] << 24);
86: }
87:
88: /* return a 32bit little-endian number */
89: int isonum_733(char *p)
90: {
91:     return isonum_731(p);
92: }
93:
94: /* return a date and time format */
95: unsigned long int isodate(char *p)
96: {
97:     struct mt mt;
98:
99:     if(!p[0]) {
100:         return 0;
101:     }
102:
103:     mt.mt_sec = p[5];
104:     mt.mt_min = p[4];
105:     mt.mt_hour = p[3];
106:     mt.mt_day = p[2];
107:     mt.mt_month = p[1];
108:     mt.mt_year = p[0];
109:     mt.mt_year += 1900;
110:     mt.mt_min += p[6] * 15;
111:
112:     return mktime(&mt);
113: }
114:
115: /* return a clean filename */
116: int iso9660_cleanfilename(char *filename, int len)
117: {
118:     int n;
119:     char *p;
120:
121:     p = filename;
122:     if(len > 2) {
123:         for(n = 0; n < len; n++) {
124:             if((len - n) == 2) {
125:                 if(p[n] == ';' && p[n + 1] == '1') {
126:                     filename[n] = NULL;
127:                     if(p[n - 1] == '.') {
128:                         filename[n - 1] = NULL;
129:                     }
130:                     return 0;
131:                 }
132:             }
133:         }
134:     }

```

fs/iso9660/super.c

Page 3/4

```

135:         return 1;
136:     }
137:
138: void iso9660_statfs(struct superblock *sb, struct statfs *statfsbuf)
139: {
140:     statfsbuf->f_type = ISO9660_SUPER_MAGIC;
141:     statfsbuf->f_bsize = sb->s_blocksize;
142:     statfsbuf->f_blocks = isonum_733(sb->u.iso9660.sb->volume_space_size);
143:     statfsbuf->f_bfree = 0;
144:     statfsbuf->f_bavail = 0;
145:     statfsbuf->f_files = 0;          /* FIXME */
146:     statfsbuf->f_ffree = 0;
147:     /* statfsbuf->f_fsid = ? */
148:     statfsbuf->f_namelen = NAME_MAX;
149: }
150:
151: int iso9660_read_superblock(__dev_t dev, struct superblock *sb)
152: {
153:     struct buffer *buf;
154:     struct iso9660_super_block *iso9660sb;
155:     struct iso9660_super_block *pvd;
156:     struct iso9660_directory_record *dr;
157:     __ino_t root_inode;
158:     int n;
159:
160:     superblock_lock(sb);
161:     pvd = NULL;
162:
163:     for(n = 0; n < ISO9660_MAX_VD; n++) {
164:         if(!(buf = bread(dev, ISO9660_SUPERBLOCK + n, BLKSIZE_2K))) {
165:             superblock_unlock(sb);
166:             return -EIO;
167:         }
168:
169:         iso9660sb = (struct iso9660_super_block *)buf->data;
170:         if(strncmp(iso9660sb->id, ISO9660_STANDARD_ID, sizeof(iso9660sb-
>id)) || (isonum_711(iso9660sb->type) == ISO9660_VD_END)) {
171:             break;
172:         }
173:         if(isonum_711(iso9660sb->type) == ISO9660_VD_PRIMARY) {
174:             pvd = (struct iso9660_super_block *)buf->data;
175:             break;
176:         }
177:         brelse(buf);
178:     }
179:     if(!pvd) {
180:         printk("WARNING: %s(): invalid filesystem type or bad superblock
on device %d,%d.\n", __FUNCTION__, MAJOR(dev), MINOR(dev));
181:         superblock_unlock(sb);
182:         brelse(buf);
183:         return -EINVAL;
184:     }
185:
186:     dr = (struct iso9660_directory_record *)pvd->root_directory_record;
187:     root_inode = isonum_711(dr->extent);
188:
189:     sb->dev = dev;
190:     sb->fsop = &iso9660_fsop;
191:     sb->flags = MS_RDONLY;
192:     sb->s_blocksize = isonum_723(pvd->logical_block_size);
193:     sb->u.iso9660.rrip = 0;
194:     if(!(sb->u.iso9660.sb = (void *)kmallocc())) {
195:         superblock_unlock(sb);
196:         brelse(buf);
197:         return -ENOMEM;
198:     }
199:     memcpy_b(sb->u.iso9660.sb, pvd, sizeof(struct iso9660_super_block));

```


fs/iso9660/super.c

Page 4/4

```
200:         brelse(buf);
201:
202:         root_inode = (root_inode << ISO9660_INODE_BITS) + (0 & ISO9660_INODE_MAS
K);
203:         if(!(sb->root = iget(sb, root_inode))) {
204:             printk("WARNING: %s(): unable to get root inode.\n", __FUNCTION_
_);
205:             superblock_unlock(sb);
206:             return -EINVAL;
207:         }
208:         sb->u.iso9660.s_root_inode = root_inode;
209:
210:         superblock_unlock(sb);
211:         return 0;
212:     }
213:
214: void iso9660_release_superblock(struct superblock *sb)
215: {
216:     kfree((unsigned int) sb->u.iso9660.sb);
217:     kfree((unsigned int) sb->u.iso9660.pathtable);
218:     kfree((unsigned int) sb->u.iso9660.pathtable_raw);
219: }
220:
221: int iso9660_init(void)
222: {
223:     return register_filesystem("iso9660", &iso9660_fsop);
224: }
```

fs/iso9660/symlink.c

Page 1/2

```

1:  /*
2:  *  fiwix/fs/iso9660/symlink.c
3:  *
4:  *  Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  *  Distributed under the terms of the Fiwix License.
6:  */
7:
8:  #include <fiwix/types.h>
9:  #include <fiwix/errno.h>
10: #include <fiwix/buffer.h>
11: #include <fiwix/fs.h>
12: #include <fiwix/filesystems.h>
13: #include <fiwix/fs_iso9660.h>
14: #include <fiwix/stat.h>
15: #include <fiwix/mm.h>
16: #include <fiwix/stdio.h>
17: #include <fiwix/string.h>
18:
19: struct fs_operations iso9660_symlink_fsop = {
20:     0,
21:     0,
22:
23:     NULL,                /* open */
24:     NULL,                /* close */
25:     NULL,                /* read */
26:     NULL,                /* write */
27:     NULL,                /* ioctl */
28:     NULL,                /* lseek */
29:     NULL,                /* readdir */
30:     NULL,                /* mmap */
31:     NULL,                /* select */
32:
33:     iso9660_readlink,
34:     iso9660_followlink,
35:     NULL,                /* bmap */
36:     NULL,                /* lookup */
37:     NULL,                /* rmdir */
38:     NULL,                /* link */
39:     NULL,                /* unlink */
40:     NULL,                /* symlink */
41:     NULL,                /* mkdir */
42:     NULL,                /* mknod */
43:     NULL,                /* truncate */
44:     NULL,                /* create */
45:     NULL,                /* rename */
46:
47:     NULL,                /* read_block */
48:     NULL,                /* write_block */
49:
50:     NULL,                /* read_inode */
51:     NULL,                /* write_inode */
52:     NULL,                /* ialloc */
53:     NULL,                /* ifree */
54:     NULL,                /* statfs */
55:     NULL,                /* read_superblock */
56:     NULL,                /* remount_fs */
57:     NULL,                /* write_superblock */
58:     NULL,                /* release_superblock */
59: };
60:
61: int iso9660_readlink(struct inode *i, char *buffer, __size_t count)
62: {
63:     __off_t size_read;
64:     char *name;
65:
66:     if (!(name = (char *)kmalloc())) {
67:         return -ENOMEM;

```

fs/iso9660/symlink.c

Page 2/2

```

68:         }
69:
70:         inode_lock(i);
71:         name[0] = NULL;
72:         if((size_read = get_rrip_symlink(i, name))) {
73:             size_read = MIN(size_read, count);
74:             memcpy_b(buffer, name, size_read);
75:         }
76:         kfree((unsigned int)name);
77:         inode_unlock(i);
78:         return size_read;
79:     }
80:
81: int iso9660_followlink(struct inode *dir, struct inode *i, struct inode **i_res)
82: {
83:     char *name;
84:     __ino_t errno;
85:
86:     if(!i) {
87:         return -ENOENT;
88:     }
89:     if(!S_ISLNK(i->i_mode)) {
90:         printk("%s(): Oops, inode '%d' is not a symlink (!?)\n", __FUNC
TION__, i->inode);
91:         return 0;
92:     }
93:
94:     if(!(name = (char *)kmalloc())) {
95:         return -ENOMEM;
96:     }
97:
98:     name[0] = NULL;
99:     if(get_rrip_symlink(i, name)) {
100:         iput(i);
101:         if((errno = parse_namei(name, dir, i_res, NULL, FOLLOW_LINKS)))
{
102:             kfree((unsigned int)name);
103:             return errno;
104:         }
105:     }
106:     kfree((unsigned int)name);
107:     return 0;
108: }

```

fs/minix/bitmaps.c

Page 1/3

```

1: /*
2:  * fiwix/fs/minix/bitmaps.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/filesystems.h>
11: #include <fiwix/fs_minix.h>
12: #include <fiwix/buffer.h>
13: #include <fiwix/errno.h>
14: #include <fiwix/stdio.h>
15: #include <fiwix/string.h>
16:
17: #define COUNT          1
18: #define FIRST_ZERO    2
19:
20: static int count_bits(struct superblock *sb, __blk_t offset, int num, int blocks
, int mode)
21: {
22:     unsigned char c;
23:     int blksize;
24:     int n, n2, last, bits, count, mapb;
25:     struct buffer *buf;
26:
27:     count = mapb = 0;
28:     blksize = sb->s_blocksize;
29:
30:     while(offset < blocks) {
31:         if(!(buf = bread(sb->dev, offset, blksize))) {
32:             return -EIO;
33:         }
34:         last = (num / 8) > blksize ? blksize : (num / 8);
35:         for(n = 0; n < last; n++) {
36:             c = (unsigned char)buf->data[n];
37:             bits = n < last ? 8 : num & 8;
38:             for(n2 = 0; n2 < bits; n2++) {
39:                 if(c & (1 << n2)) {
40:                     if(mode == COUNT) {
41:                         count++;
42:                     }
43:                 } else {
44:                     if(mode == FIRST_ZERO) {
45:                         brelse(buf);
46:                         return n2 + ((n * 8) + (mapb * b
blksize * 8));
47:                     }
48:                 }
49:             }
50:         }
51:         offset++;
52:         mapb++;
53:         num -= (blksize * 8);
54:         brelse(buf);
55:     }
56:     return count;
57: }
58:
59: int minix_change_bit(int mode, struct superblock *sb, int map, int item)
60: {
61:     int byte, bit, mask;
62:     struct buffer *buf;
63:
64:     map += item / (sb->s_blocksize * 8);
65:     byte = (item % (sb->s_blocksize * 8)) / 8;

```

fs/minix/bitmaps.c

Page 2/3

```

66:         bit = (item % (sb->s_blocksize * 8)) % 8;
67:         mask = 1 << bit;
68:
69:         if(!(buf = bread(sb->dev, map, sb->s_blocksize))) {
70:             return -EIO;
71:         }
72:
73:         if(mode == CLEAR_BIT) {
74:             if(!(buf->data[byte] & mask)) {
75:                 brelse(buf);
76:                 return 1;
77:             }
78:             buf->data[byte] &= ~mask;
79:         }
80:         if(mode == SET_BIT) {
81:             if((buf->data[byte] & mask)) {
82:                 brelse(buf);
83:                 return 1;
84:             }
85:             buf->data[byte] |= mask;
86:         }
87:
88:         bwrite(buf);
89:         return 0;
90: }
91:
92: int minix_balloc(struct superblock *sb)
93: {
94:     int map, block, errno;
95:
96:     superblock_lock(sb);
97:
98:     map = 1 + SUPERBLOCK + sb->u.minix.sb.s_imap_blocks;
99:
100:    if(!(block = minix_find_first_zero(sb, map, sb->u.minix.nzones, map + sb
->u.minix.sb.s_zmap_blocks))) {
101:        superblock_unlock(sb);
102:        return -ENOSPC;
103:    }
104:
105:    errno = minix_change_bit(SET_BIT, sb, map, block);
106:    block += sb->u.minix.sb.s_firstdatazone - 1;
107:
108:    if(errno) {
109:        if(errno < 0) {
110:            printk("WARNING: %s(): unable to set block %d.\n", __FUN
CTION__, block);
111:            superblock_unlock(sb);
112:            return errno;
113:        } else {
114:            printk("WARNING: %s(): block %d is already marked as use
d!\n", __FUNCTION__, block);
115:        }
116:    }
117:
118:    superblock_unlock(sb);
119:    return block;
120: }
121:
122: void minix_bfree(struct superblock *sb, int block)
123: {
124:     int map, errno;
125:
126:     if(block < sb->u.minix.sb.s_firstdatazone || block > sb->u.minix.nzones)
{
127:         printk("WARNING: %s(): block %d is not in datazone.\n", __FUNCTI
ON__, block);

```

fs/minix/bitmaps.c

Page 3/3

```

128:         return;
129:     }
130:
131:     superblock_lock(sb);
132:
133:     map = 1 + SUPERBLOCK + sb->u.minix.sb.s_imap_blocks;
134:     block -= sb->u.minix.sb.s_firstdatazone - 1;
135:     errno = minix_change_bit(CLEAR_BIT, sb, map, block);
136:
137:     if(errno) {
138:         if(errno < 0) {
139:             printk("WARNING: %s(): unable to free block %d.\n", __FU
NCTION__, block);
140:         } else {
141:             printk("WARNING: %s(): block %d is already marked as fre
e!\n", __FUNCTION__, block);
142:         }
143:     }
144:
145:     superblock_unlock(sb);
146:     return;
147: }
148:
149: int minix_count_free_inodes(struct superblock *sb)
150: {
151:     __blk_t offset;
152:
153:     offset = 1 + SUPERBLOCK;
154:     return count_bits(sb, offset, sb->u.minix.sb.s_ninodes, offset + sb->u.m
inix.sb.s_imap_blocks, COUNT);
155: }
156:
157: int minix_count_free_blocks(struct superblock *sb)
158: {
159:     __blk_t offset;
160:
161:     offset = 1 + SUPERBLOCK + sb->u.minix.sb.s_imap_blocks;
162:     return count_bits(sb, offset, sb->u.minix.nzones, offset + sb->u.minix.s
b.s_zmap_blocks, COUNT);
163: }
164:
165: int minix_find_first_zero(struct superblock *sb, __blk_t offset, int num, int bl
ocks)
166: {
167:     return count_bits(sb, offset, num, blocks, FIRST_ZERO);
168: }

```

fs/minix/dir.c

Page 1/3

```

1: /*
2:  * fiwix/fs/minix/dir.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/errno.h>
10: #include <fiwix/buffer.h>
11: #include <fiwix/fs.h>
12: #include <fiwix/filesystems.h>
13: #include <fiwix/stat.h>
14: #include <fiwix/dirent.h>
15: #include <fiwix/mm.h>
16: #include <fiwix/stdio.h>
17: #include <fiwix/string.h>
18:
19: struct fs_operations minix_dir_fsop = {
20:     0,
21:     0,
22:
23:     minix_dir_open,
24:     minix_dir_close,
25:     minix_dir_read,
26:     minix_dir_write,
27:     NULL,                /* ioctl */
28:     NULL,                /* lseek */
29:     minix_dir_readdir,
30:     NULL,                /* mmap */
31:     NULL,                /* select */
32:
33:     NULL,                /* readlink */
34:     NULL,                /* followlink */
35:     minix_bmap,
36:     minix_lookup,
37:     minix_rmdir,
38:     minix_link,
39:     minix_unlink,
40:     minix_symlink,
41:     minix_mkdir,
42:     minix_mknod,
43:     NULL,                /* truncate */
44:     minix_create,
45:     minix_rename,
46:
47:     NULL,                /* read_block */
48:     NULL,                /* write_block */
49:
50:     NULL,                /* read_inode */
51:     NULL,                /* write_inode */
52:     NULL,                /* ialloc */
53:     NULL,                /* ifree */
54:     NULL,                /* statfs */
55:     NULL,                /* read_superblock */
56:     NULL,                /* remount_fs */
57:     NULL,                /* write_superblock */
58:     NULL,                /* release_superblock */
59: };
60:
61: int minix_dir_open(struct inode *i, struct fd *fd_table)
62: {
63:     fd_table->offset = 0;
64:     return 0;
65: }
66:
67: int minix_dir_close(struct inode *i, struct fd *fd_table)

```

fs/minix/dir.c

Page 2/3

```

68: {
69:     return 0;
70: }
71:
72: int minix_dir_read(struct inode *i, struct fd *fd_table, char *buffer, __size_t
count)
73: {
74:     return -EISDIR;
75: }
76:
77: int minix_dir_write(struct inode *i, struct fd *fd_table, const char *buffer, __
size_t count)
78: {
79:     return -EBADF;
80: }
81:
82: int minix_dir_readdir(struct inode *i, struct fd *fd_table, struct dirent *diren
t, unsigned int count)
83: {
84:     __blk_t block;
85:     unsigned int doffset, offset;
86:     unsigned int size, dirent_len;
87:     struct minix_dir_entry *d;
88:     int base_dirent_len;
89:     int blksize;
90:     struct buffer *buf;
91:
92:     if(!(S_ISDIR(i->i_mode))) {
93:         return -EBADF;
94:     }
95:
96:     blksize = i->sb->s_blocksize;
97:     if(fd_table->offset > i->i_size) {
98:         fd_table->offset = i->i_size;
99:     }
100:
101:     base_dirent_len = sizeof(dirent->d_ino) + sizeof(dirent->d_off) + sizeof
(dirent->d_reclen);
102:     doffset = offset = size = 0;
103:
104:     while(fd_table->offset < i->i_size && count > 0) {
105:         if((block = bmap(i, fd_table->offset, FOR_READING)) < 0) {
106:             return block;
107:         }
108:         if(block) {
109:             if(!(buf = bread(i->dev, block, blksize))) {
110:                 return -EIO;
111:             }
112:
113:             doffset = fd_table->offset;
114:             offset = fd_table->offset % blksize;
115:             while(offset < blksize) {
116:                 d = (struct minix_dir_entry *) (buf->data + offse
t);
117:                 if(d->inode) {
118:                     dirent_len = (base_dirent_len + (strlen(
d->name) + 1)) + 3;
119:                     dirent_len &= ~3; /* round up */
120:                     dirent->d_ino = d->inode;
121:                     if((size + dirent_len) < count) {
122:                         dirent->d_off = doffset;
123:                         dirent->d_reclen = dirent_len;
124:                         memcpy_b(dirent->d_name, d->name
, strlen(d->name));
125:                         dirent->d_name[strlen(d->name)]
= NULL;
126:                         dirent = (struct dirent *) ((char

```


fs/minix/dir.c

Page 3/3

```
*)dirent + dirent_len);
127:                                     size += dirent_len;
128:                                     count -= size;
129:                                     } else {
130:                                     count = 0;
131:                                     break;
132:                                     }
133:                                     }
134:                                     doffset += i->sb->u.minix.dirsize;
135:                                     offset += i->sb->u.minix.dirsize;
136:                                     }
137:                                     brelse(buf);
138:                                     }
139:                                     fd_table->offset &= ~(blksize - 1);
140:                                     doffset = fd_table->offset;
141:                                     fd_table->offset += offset;
142:                                     doffset += blksize;
143:                                     }
144:
145:                                     return size;
146: }
```

fs/minix/file.c

Page 1/2

```

1:  /*
2:  *  fiwix/fs/minix/file.c
3:  *
4:  *  Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  *  Distributed under the terms of the Fiwix License.
6:  */
7:
8:  #include <fiwix/kernel.h>
9:  #include <fiwix/types.h>
10: #include <fiwix/errno.h>
11: #include <fiwix/buffer.h>
12: #include <fiwix/fs.h>
13: #include <fiwix/filesystems.h>
14: #include <fiwix/mm.h>
15: #include <fiwix/mman.h>
16: #include <fiwix/fcntl.h>
17: #include <fiwix/stdio.h>
18: #include <fiwix/string.h>
19:
20: struct fs_operations minix_file_fsop = {
21:     0,
22:     0,
23:
24:     minix_file_open,
25:     minix_file_close,
26:     file_read,
27:     minix_file_write,
28:     NULL,                /* ioctl */
29:     minix_file_lseek,
30:     NULL,                /* readdir */
31:     NULL,                /* mmap */
32:     NULL,                /* select */
33:
34:     NULL,                /* readlink */
35:     NULL,                /* followlink */
36:     minix_bmap,
37:     NULL,                /* lookup */
38:     NULL,                /* rmdir */
39:     NULL,                /* link */
40:     NULL,                /* unlink */
41:     NULL,                /* symlink */
42:     NULL,                /* mkdir */
43:     NULL,                /* mknod */
44:     minix_truncate,
45:     NULL,                /* create */
46:     NULL,                /* rename */
47:
48:     NULL,                /* read_block */
49:     NULL,                /* write_block */
50:
51:     NULL,                /* read_inode */
52:     NULL,                /* write_inode */
53:     NULL,                /* ialloc */
54:     NULL,                /* ifree */
55:     NULL,                /* statfs */
56:     NULL,                /* read_superblock */
57:     NULL,                /* remount_fs */
58:     NULL,                /* write_superblock */
59:     NULL,                /* release_superblock */
60: };
61:
62: int minix_file_open(struct inode *i, struct fd *fd_table)
63: {
64:     if(fd_table->flags & O_APPEND) {
65:         fd_table->offset = i->i_size;
66:     } else {
67:         fd_table->offset = 0;

```

fs/minix/file.c

Page 2/2

```

68:         }
69:         if(fd_table->flags & O_TRUNC) {
70:             i->i_size = 0;
71:             minix_truncate(i, 0);
72:         }
73:         return 0;
74:     }
75:
76: int minix_file_close(struct inode *i, struct fd *fd_table)
77: {
78:     return 0;
79: }
80:
81: int minix_file_write(struct inode *i, struct fd *fd_table, const char *buffer, _
__size_t count)
82: {
83:     __blk_t block;
84:     __off_t total_written;
85:     unsigned int boffset, bytes;
86:     int blksize;
87:     struct buffer *buf;
88:
89:     inode_lock(i);
90:
91:     blksize = i->sb->s_blocksize;
92:     total_written = 0;
93:
94:     if(fd_table->flags & O_APPEND) {
95:         fd_table->offset = i->i_size;
96:     }
97:
98:     while(total_written < count) {
99:         boffset = fd_table->offset % blksize;
100:        if((block = bmap(i, fd_table->offset, FOR_WRITING)) < 0) {
101:            inode_unlock(i);
102:            return block;
103:        }
104:        bytes = blksize - boffset;
105:        bytes = MIN(bytes, (count - total_written));
106:        if(!(buf = bread(i->dev, block, blksize))) {
107:            inode_unlock(i);
108:            return -EIO;
109:        }
110:        memcpy_b(buf->data + boffset, buffer + total_written, bytes);
111:        update_page_cache(i, fd_table->offset, buffer + total_written, b
ytes);
112:        bwrite(buf);
113:        total_written += bytes;
114:        fd_table->offset += bytes;
115:    }
116:
117:    if(fd_table->offset > i->i_size) {
118:        i->i_size = fd_table->offset;
119:    }
120:    i->i_ctime = CURRENT_TIME;
121:    i->i_mtime = CURRENT_TIME;
122:    i->dirty = 1;
123:
124:    inode_unlock(i);
125:    return total_written;
126: }
127:
128: int minix_file_lseek(struct inode *i, __off_t offset)
129: {
130:     return offset;
131: }

```

fs/minix/inode.c

Page 1/2

```
1: /*
2:  * fiwix/fs/minix/inode.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/filesystems.h>
11: #include <fiwix/fs_minix.h>
12: #include <fiwix/fs_pipe.h>
13: #include <fiwix/statfs.h>
14: #include <fiwix/sleep.h>
15: #include <fiwix/stat.h>
16: #include <fiwix/sched.h>
17: #include <fiwix/buffer.h>
18: #include <fiwix/process.h>
19: #include <fiwix/errno.h>
20: #include <fiwix/stdio.h>
21: #include <fiwix/string.h>
22:
23: int minix_read_inode(struct inode *i)
24: {
25:     if(i->sb->u.minix.version == 1) {
26:         return v1_minix_read_inode(i);
27:     }
28:
29:     return v2_minix_read_inode(i);
30: }
31:
32: int minix_write_inode(struct inode *i)
33: {
34:     if(i->sb->u.minix.version == 1) {
35:         return v1_minix_write_inode(i);
36:     }
37:
38:     return v2_minix_write_inode(i);
39: }
40:
41: int minix_ialloc(struct inode *i, int mode)
42: {
43:     if(i->sb->u.minix.version == 1) {
44:         return v1_minix_ialloc(i, mode);
45:     }
46:
47:     return v2_minix_ialloc(i, mode);
48: }
49:
50: void minix_ifree(struct inode *i)
51: {
52:     if(i->sb->u.minix.version == 1) {
53:         return v1_minix_ifree(i);
54:     }
55:
56:     return v2_minix_ifree(i);
57: }
58:
59: int minix_bmap(struct inode *i, __off_t offset, int mode)
60: {
61:     if(i->sb->u.minix.version == 1) {
62:         return v1_minix_bmap(i, offset, mode);
63:     }
64:
65:     return v2_minix_bmap(i, offset, mode);
66: }
67:
```

fs/minix/inode.c

Page 2/2

```
68: int minix_truncate(struct inode *i, __off_t length)
69: {
70:     if(i->sb->u.minix.version == 1) {
71:         return v1_minix_truncate(i, length);
72:     }
73:
74:     return v2_minix_truncate(i, length);
75: }
```

fs/minix/Makefile

Page 1/1

```
1: # fiwix/fs/minix/Makefile
2: #
3: # Copyright 2018, Jordi Sanfeliu. All rights reserved.
4: # Distributed under the terms of the Fiwix License.
5: #
6:
7: .S.o:
8:     $(CC) -traditional -I$(INCLUDE) -c -o $@ $<
9: .c.o:
10:     $(CC) $(CFLAGS) -c -o $@ $<
11:
12: OBJS = super.o bitmaps.o inode.o namei.o symlink.o dir.o file.o v1_inode.o v2_in
ode.o
13:
14: minix: $(OBJS)
15:     $(LD) $(LDFLAGS) -r $(OBJS) -o minix.o
16:
17: clean:
18:     rm -f *.o
19:
```

fs/minix/namei.c

Page 1/12

```

1: /*
2:  * fiwix/fs/minix/namei.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/types.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/filesystems.h>
12: #include <fiwix/fs_minix.h>
13: #include <fiwix/buffer.h>
14: #include <fiwix/errno.h>
15: #include <fiwix/stat.h>
16: #include <fiwix/stdio.h>
17: #include <fiwix/string.h>
18:
19: static int is_dir_empty(struct inode *dir)
20: {
21:     __blk_t block;
22:     unsigned int blksize;
23:     unsigned int offset, doffset;
24:     struct buffer *buf;
25:     struct minix_dir_entry *d;
26:
27:     blksize = dir->sb->s_blocksize;
28:     doffset = dir->sb->u.minix.dirsize * 2; /* accept only "." and ".." */
29:     offset = 0;
30:
31:     while(offset < dir->i_size) {
32:         if((block = bmap(dir, offset, FOR_READING)) < 0) {
33:             break;
34:         }
35:         if(block) {
36:             if(!(buf = bread(dir->dev, block, blksize))) {
37:                 break;
38:             }
39:             do {
40:                 if(doffset + offset >= dir->i_size) {
41:                     break;
42:                 }
43:                 d = (struct minix_dir_entry *) (buf->data + doffs
et);
44:                 if(d->inode) {
45:                     brelse(buf);
46:                     return 0;
47:                 }
48:                 doffset += dir->sb->u.minix.dirsize;
49:             } while(doffset < blksize);
50:             brelse(buf);
51:             offset += blksize;
52:             doffset = 0;
53:         } else {
54:             break;
55:         }
56:     }
57:
58:     return 1;
59: }
60:
61: /* finds the entry 'name' with inode 'i' in the directory 'dir' */
62: static struct buffer * find_dir_entry(struct inode *dir, struct inode *i, struct
minix_dir_entry **d_res, char *name)
63: {
64:     __blk_t block;
65:     unsigned int blksize;

```

fs/minix/namei.c

Page 2/12

```

66:         unsigned int offset, doffset;
67:         struct buffer *buf;
68:
69:         blksize = dir->sb->s_blocksize;
70:         offset = 0;
71:
72:         while(offset < dir->i_size) {
73:             if((block = bmap(dir, offset, FOR_READING)) < 0) {
74:                 break;
75:             }
76:             if(block) {
77:                 if(!(buf = bread(dir->dev, block, blksize))) {
78:                     break;
79:                 }
80:                 doffset = 0;
81:                 do {
82:                     *d_res = (struct minix_dir_entry *) (buf->data +
doffset);
83:                     if(!i) {
84:                         /* returns the first empty entry */
85:                         if(!(*d_res)->inode || (doffset + offset
>= dir->i_size)) {
86:                             /* the directory grows by direct
ory entry size */
87:                             if(doffset + offset >= dir->i_si
ze) {
88:                                 dir->i_size += dir->sb->
u.minix.dirsize;
89:                             }
90:                             return buf;
91:                         }
92:                     } else {
93:                         if((*d_res)->inode == i->inode) {
94:                             /* returns the first matching in
ode */
95:                             if(!name) {
96:                                 return buf;
97:                             }
98:                             /* returns the matching inode an
d name */
99:                             if(!strcmp((*d_res)->name, name)
) {
100:                                 return buf;
101:                             }
102:                         }
103:                     }
104:                     doffset += dir->sb->u.minix.dirsize;
105:                 } while(doffset < blksize);
106:                 brelse(buf);
107:                 offset += blksize;
108:             } else {
109:                 break;
110:             }
111:         }
112:
113:         *d_res = NULL;
114:         return NULL;
115:     }
116:
117:     static struct buffer * add_dir_entry(struct inode *dir, struct minix_dir_entry *
*d_res)
118:     {
119:         __blk_t block;
120:         struct buffer *buf;
121:
122:         if(!(buf = find_dir_entry(dir, NULL, d_res, NULL))) {
123:             if((block = bmap(dir, dir->i_size, FOR_WRITING)) < 0) {

```


fs/minix/namei.c

Page 3/12

```

124:             return NULL;
125:         }
126:         if (!(buf = bread(dir->dev, block, dir->sb->s_blocksize))) {
127:             return NULL;
128:         }
129:         *d_res = (struct minix_dir_entry *)buf->data;
130:         dir->i_size += dir->sb->u.minix.dirsize;
131:     }
132:
133:     return buf;
134: }
135:
136: static int is_subdir(struct inode *dir_new, struct inode *i_old)
137: {
138:     __ino_t inode;
139:     int errno;
140:
141:     errno = 0;
142:     dir_new->count++;
143:     for(;;) {
144:         if(dir_new == i_old) {
145:             errno = 1;
146:             break;
147:         }
148:         inode = dir_new->inode;
149:         if(minix_lookup("../", dir_new, &dir_new)) {
150:             break;
151:         }
152:         if(dir_new->inode == inode) {
153:             break;
154:         }
155:     }
156:     iput(dir_new);
157:     return errno;
158: }
159:
160: int minix_lookup(const char *name, struct inode *dir, struct inode **i_res)
161: {
162:     __blk_t block;
163:     unsigned int blksize;
164:     unsigned int offset, doffset;
165:     struct buffer *buf;
166:     struct minix_dir_entry *d;
167:     __ino_t inode;
168:
169:     blksize = dir->sb->s_blocksize;
170:     inode = offset = 0;
171:
172:     while(offset < dir->i_size && !inode) {
173:         if((block = bmap(dir, offset, FOR_READING)) < 0) {
174:             iput(dir);
175:             return block;
176:         }
177:         if(block) {
178:             if (!(buf = bread(dir->dev, block, blksize))) {
179:                 iput(dir);
180:                 return -EIO;
181:             }
182:             doffset = 0;
183:             do {
184:                 d = (struct minix_dir_entry *) (buf->data + doffs
et);
185:                 if(d->inode) {
186:                     if(strlen(d->name) == strlen(name)) {
187:                         if (!(strcmp(d->name, name))) {
188:                             inode = d->inode;
189:                         }

```

fs/minix/namei.c

Page 4/12

```

190:         }
191:     }
192:     doffset += dir->sb->u.minix.dirsize;
193: } while((doffset < blksize) && (!inode));
194:
195: brelse(buf);
196: if(inode) {
197:     if(!(*i_res = iget(dir->sb, inode))) {
198:         iput(dir);
199:         return -EACCES;
200:     }
201:     iput(dir);
202:     return 0;
203: }
204: offset += blksize;
205: } else {
206:     break;
207: }
208: }
209: iput(dir);
210: return -ENOENT;
211: }
212:
213: int minix_rmdir(struct inode *dir, struct inode *i)
214: {
215:     struct buffer *buf;
216:     struct minix_dir_entry *d;
217:
218:     inode_lock(i);
219:
220:     if(!is_dir_empty(i)) {
221:         inode_unlock(i);
222:         return -ENOTEMPTY;
223:     }
224:
225:     inode_lock(dir);
226:
227:     if(!(buf = find_dir_entry(dir, i, &d, NULL))) {
228:         inode_unlock(i);
229:         inode_unlock(dir);
230:         return -ENOENT;
231:     }
232:
233:     d->inode = 0;
234:     i->i_nlink = 0;
235:     dir->i_nlink--;
236:
237:     i->i_ctime = CURRENT_TIME;
238:     dir->i_mtime = CURRENT_TIME;
239:     dir->i_ctime = CURRENT_TIME;
240:
241:     i->dirty = 1;
242:     dir->dirty = 1;
243:
244:     bwrite(buf);
245:
246:     inode_unlock(i);
247:     inode_unlock(dir);
248:     return 0;
249: }
250:
251: int minix_link(struct inode *i_old, struct inode *dir_new, char *name)
252: {
253:     struct buffer *buf;
254:     struct minix_dir_entry *d;
255:     int n;
256:

```

fs/minix/namei.c

Page 5/12

```

257:     inode_lock(i_old);
258:     inode_lock(dir_new);
259:
260:     if(!(buf = add_dir_entry(dir_new, &d))) {
261:         inode_unlock(i_old);
262:         inode_unlock(dir_new);
263:         return -ENOSPC;
264:     }
265:
266:     d->inode = i_old->inode;
267:     for(n = 0; n < i_old->sb->u.minix.namelen; n++) {
268:         d->name[n] = name[n];
269:         if(!name[n]) {
270:             break;
271:         }
272:     }
273:     for(; n < i_old->sb->u.minix.namelen; n++) {
274:         d->name[n] = 0;
275:     }
276:
277:     i_old->i_nlink++;
278:     i_old->i_ctime = CURRENT_TIME;
279:     dir_new->i_mtime = CURRENT_TIME;
280:     dir_new->i_ctime = CURRENT_TIME;
281:
282:     i_old->dirty = 1;
283:     dir_new->dirty = 1;
284:
285:     bwrite(buf);
286:
287:     inode_unlock(i_old);
288:     inode_unlock(dir_new);
289:     return 0;
290: }
291:
292: int minix_unlink(struct inode *dir, struct inode *i, char *name)
293: {
294:     struct buffer *buf;
295:     struct minix_dir_entry *d;
296:
297:     inode_lock(dir);
298:     inode_lock(i);
299:
300:     if(!(buf = find_dir_entry(dir, i, &d, name))) {
301:         inode_unlock(dir);
302:         inode_unlock(i);
303:         return -ENOENT;
304:     }
305:
306:     d->inode = 0;
307:     i->i_nlink--;
308:
309:     i->i_ctime = CURRENT_TIME;
310:     dir->i_mtime = CURRENT_TIME;
311:     dir->i_ctime = CURRENT_TIME;
312:
313:     i->dirty = 1;
314:     dir->dirty = 1;
315:
316:     bwrite(buf);
317:
318:     inode_unlock(dir);
319:     inode_unlock(i);
320:     return 0;
321: }
322:
323: int minix_symlink(struct inode *dir, char *name, char *oldname)

```

```

324: {
325:     struct buffer *buf, *buf_new;
326:     struct inode *i;
327:     struct minix_dir_entry *d;
328:     unsigned int blksize;
329:     int n, block;
330:     char c;
331:
332:     inode_lock(dir);
333:
334:     if(!(i = ialloc(dir->sb, S_IFLNK))) {
335:         inode_unlock(dir);
336:         return -ENOSPC;
337:     }
338:
339:     i->i_mode = S_IFLNK | (S_IRWXU | S_IRWXG | S_IRWXO);
340:     i->i_uid = current->euid;
341:     i->i_gid = current->egid;
342:     i->i_nlink = 1;
343:     i->dev = dir->dev;
344:     i->count = 1;
345:     i->fsop = &minix_symlink_fsop;
346:     i->dirty = 1;
347:
348:     block = minix_balloc(dir->sb);
349:     if(block < 0) {
350:         i->i_nlink = 0;
351:         iput(i);
352:         inode_unlock(dir);
353:         return -ENOSPC;
354:     }
355:
356:     if(i->sb->u.minix.version == 1) {
357:         i->u.minix.u.i1_zone[0] = block;
358:     } else {
359:         i->u.minix.u.i2_zone[0] = block;
360:     }
361:     blksize = dir->sb->s_blocksize;
362:     if(!(buf_new = bread(dir->dev, block, blksize))) {
363:         minix_bfree(dir->sb, block);
364:         i->i_nlink = 0;
365:         iput(i);
366:         inode_unlock(dir);
367:         return -EIO;
368:     }
369:
370:     if(!(buf = add_dir_entry(dir, &d))) {
371:         minix_bfree(dir->sb, block);
372:         i->i_nlink = 0;
373:         iput(i);
374:         inode_unlock(dir);
375:         return -ENOSPC;
376:     }
377:
378:     d->inode = i->inode;
379:     for(n = 0; n < i->sb->u.minix.namelen; n++) {
380:         d->name[n] = name[n];
381:         if(!name[n]) {
382:             break;
383:         }
384:     }
385:     for(; n < i->sb->u.minix.namelen; n++) {
386:         d->name[n] = 0;
387:     }
388:
389:     for(n = 0; n < NAME_MAX; n++) {
390:         if((c = oldname[n])) {

```

fs/minix/namei.c

Page 7/12

```

391:             buf_new->data[n] = c;
392:             continue;
393:         }
394:         break;
395:     }
396:     buf_new->data[n] = 0;
397:     i->i_size = n;
398:
399:     dir->i_mtime = CURRENT_TIME;
400:     dir->i_ctime = CURRENT_TIME;
401:     dir->dirty = 1;
402:
403:     bwrite(buf);
404:     bwrite(buf_new);
405:     iput(i);
406:     inode_unlock(dir);
407:     return 0;
408: }
409:
410: int minix_mkdir(struct inode *dir, char *name, __mode_t mode)
411: {
412:     struct buffer *buf, *buf_new;
413:     struct inode *i;
414:     struct minix_dir_entry *d, *d_new;
415:     unsigned int blksize;
416:     int n, block;
417:
418:     if(strlen(name) > dir->sb->u.minix.namelen) {
419:         return -ENAMETOOLONG;
420:     }
421:
422:     inode_lock(dir);
423:
424:     if(!(i = ialloc(dir->sb, S_IFDIR))) {
425:         inode_unlock(dir);
426:         return -ENOSPC;
427:     }
428:
429:     i->i_mode = ((mode & (S_IRWXU | S_IRWXG | S_IRWXO)) & ~current->umask);
430:     i->i_mode |= S_IFDIR;
431:     i->i_uid = current->euid;
432:     i->i_gid = current->egid;
433:     i->i_nlink = 1;
434:     i->dev = dir->dev;
435:     i->count = 1;
436:     i->fsop = &minix_dir_fsop;
437:     i->dirty = 1;
438:
439:     if((block = bmap(i, 0, FOR_WRITING)) < 0) {
440:         i->i_nlink = 0;
441:         iput(i);
442:         inode_unlock(dir);
443:         return -ENOSPC;
444:     }
445:
446:     blksize = dir->sb->s_blocksize;
447:     if(!(buf_new = bread(i->dev, block, blksize))) {
448:         minix_bfree(dir->sb, block);
449:         i->i_nlink = 0;
450:         iput(i);
451:         inode_unlock(dir);
452:         return -EIO;
453:     }
454:
455:     if(!(buf = add_dir_entry(dir, &d))) {
456:         minix_bfree(dir->sb, block);
457:         i->i_nlink = 0;

```

fs/minix/namei.c

Page 8/12

```

458:         iput(i);
459:         inode_unlock(dir);
460:         return -ENOSPC;
461:     }
462:
463:     d->inode = i->inode;
464:     for(n = 0; n < i->sb->u.minix.namelen; n++) {
465:         d->name[n] = name[n];
466:         if(!name[n] || name[n] == '/') {
467:             break;
468:         }
469:     }
470:     for(; n < i->sb->u.minix.namelen; n++) {
471:         d->name[n] = 0;
472:     }
473:
474:     d_new = (struct minix_dir_entry *)buf_new->data;
475:     d_new->inode = i->inode;
476:     d_new->name[0] = '.';
477:     d_new->name[1] = 0;
478:     i->i_size += i->sb->u.minix.dirsize;
479:     i->i_nlink++;
480:     d_new = (struct minix_dir_entry *) (buf_new->data + i->sb->u.minix.dirsiz
e);
481:     d_new->inode = dir->inode;
482:     d_new->name[0] = '.';
483:     d_new->name[1] = '.';
484:     d_new->name[2] = 0;
485:     i->i_size += i->sb->u.minix.dirsize;
486:
487:     dir->i_mtime = CURRENT_TIME;
488:     dir->i_ctime = CURRENT_TIME;
489:     dir->i_nlink++;
490:     dir->dirty = 1;
491:
492:     bwrite(buf);
493:     bwrite(buf_new);
494:     iput(i);
495:     inode_unlock(dir);
496:     return 0;
497: }
498:
499: int minix_mknod(struct inode *dir, char *name, __mode_t mode, __dev_t dev)
500: {
501:     struct buffer *buf;
502:     struct inode *i;
503:     struct minix_dir_entry *d;
504:     int n;
505:
506:     inode_lock(dir);
507:
508:     if(!(i = ialloc(dir->sb, mode & S_IFMT))) {
509:         inode_unlock(dir);
510:         return -ENOSPC;
511:     }
512:
513:     if(!(buf = add_dir_entry(dir, &d))) {
514:         i->i_nlink = 0;
515:         iput(i);
516:         inode_unlock(dir);
517:         return -ENOSPC;
518:     }
519:
520:     d->inode = i->inode;
521:     for(n = 0; n < i->sb->u.minix.namelen; n++) {
522:         d->name[n] = name[n];
523:         if(!name[n]) {

```

fs/minix/namei.c

Page 9/12

```

524:             break;
525:         }
526:     }
527:     for(; n < i->sb->u.minix.namelen; n++) {
528:         d->name[n] = 0;
529:     }
530:
531:     i->i_mode = (mode & ~current->umask) & ~S_IFMT;
532:     i->i_uid = current->euid;
533:     i->i_gid = current->egid;
534:     i->i_nlink = 1;
535:     i->dev = dir->dev;
536:     i->count = 1;
537:     i->dirty = 1;
538:
539:     switch(mode & S_IFMT) {
540:         case S_IFCHR:
541:             i->fsop = &def_chr_fsop;
542:             i->rdev = dev;
543:             i->i_mode |= S_IFCHR;
544:             break;
545:         case S_IFBLK:
546:             i->fsop = &def_blk_fsop;
547:             i->rdev = dev;
548:             i->i_mode |= S_IFBLK;
549:             break;
550:         case S_IFIFO:
551:             i->fsop = &pipefs_fsop;
552:             i->i_mode |= S_IFIFO;
553:             /* it's a union so we need to clear pipefs_i */
554:             memset_b(&i->u.pipefs, NULL, sizeof(struct pipefs_inode)
);
555:             break;
556:     }
557:
558:     dir->i_mtime = CURRENT_TIME;
559:     dir->i_ctime = CURRENT_TIME;
560:     dir->dirty = 1;
561:
562:     bwrite(buf);
563:     iput(i);
564:     inode_unlock(dir);
565:     return 0;
566: }
567:
568: int minix_create(struct inode *dir, char *name, __mode_t mode, struct inode **i_
res)
569: {
570:     struct buffer *buf;
571:     struct inode *i;
572:     struct minix_dir_entry *d;
573:     int n;
574:
575:     if(IS_RDONLY_FS(dir)) {
576:         return -EROFS;
577:     }
578:
579:     inode_lock(dir);
580:
581:     if(!(i = ialloc(dir->sb, S_IFREG))) {
582:         inode_unlock(dir);
583:         return -ENOSPC;
584:     }
585:
586:     if(!(buf = add_dir_entry(dir, &d))) {
587:         i->i_nlink = 0;
588:         iput(i);

```

fs/minix/namei.c

Page 10/12

```

589:         inode_unlock(dir);
590:         return -ENOSPC;
591:     }
592:
593:     d->inode = i->inode;
594:     for(n = 0; n < i->sb->u.minix.namelen; n++) {
595:         d->name[n] = name[n];
596:         if(!name[n]) {
597:             break;
598:         }
599:     }
600:     for(; n < i->sb->u.minix.namelen; n++) {
601:         d->name[n] = 0;
602:     }
603:
604:     i->i_mode = (mode & ~current->umask) & ~S_IFMT;
605:     i->i_mode |= S_IFREG;
606:     i->i_uid = current->euid;
607:     i->i_gid = current->egid;
608:     i->i_nlink = 1;
609:     i->dev = dir->dev;
610:     i->fsop = &minix_file_fsop;
611:     i->count = 1;
612:     i->dirty = 1;
613:
614:     dir->i_mtime = CURRENT_TIME;
615:     dir->i_ctime = CURRENT_TIME;
616:     dir->dirty = 1;
617:
618:     *i_res = i;
619:     bwrite(buf);
620:     inode_unlock(dir);
621:     return 0;
622: }
623:
624: int minix_rename(struct inode *i_old, struct inode *dir_old, struct inode *i_new
, struct inode *dir_new, char *oldpath, char *newpath)
625: {
626:     struct buffer *buf_old, *buf_new;
627:     struct minix_dir_entry *d_old, *d_new;
628:     int errno;
629:
630:     errno = 0;
631:
632:     if(is_subdir(dir_new, i_old)) {
633:         return -EINVAL;
634:     }
635:
636:     inode_lock(i_old);
637:     inode_lock(dir_old);
638:     if(dir_old != dir_new) {
639:         inode_lock(dir_new);
640:     }
641:
642:     if(!(buf_old = find_dir_entry(dir_old, i_old, &d_old, oldpath))) {
643:         errno = -ENOENT;
644:         goto end;
645:     }
646:     if(dir_old == dir_new) {
647:         /* free that buffer now to not block buf_new */
648:         brelse(buf_old);
649:         buf_old = NULL;
650:     }
651:
652:     if(i_new) {
653:         if(S_ISDIR(i_old->i_mode)) {
654:             if(!is_dir_empty(i_new)) {

```


fs/minix/namei.c

Page 11/12

```

655:             if(buf_old) {
656:                 brelse(buf_old);
657:             }
658:             errno = -ENOTEMPTY;
659:             goto end;
660:         }
661:     }
662:     if(!(buf_new = find_dir_entry(dir_new, i_new, &d_new, newpath)))
{
663:         if(buf_old) {
664:             brelse(buf_old);
665:         }
666:         errno = -ENOENT;
667:         goto end;
668:     }
669: } else {
670:     if(!(buf_new = add_dir_entry(dir_new, &d_new))) {
671:         if(buf_old) {
672:             brelse(buf_old);
673:         }
674:         errno = -ENOSPC;
675:         goto end;
676:     }
677:     if(S_ISDIR(i_old->i_mode)) {
678:         dir_old->i_nlink--;
679:         dir_new->i_nlink++;
680:     }
681: }
682: if(i_new) {
683:     i_new->i_nlink--;
684: } else {
685:     i_new = i_old;
686:     strcpy(d_new->name, newpath);
687: }
688:
689: d_new->inode = i_old->inode;
690: dir_new->i_mtime = CURRENT_TIME;
691: dir_new->i_ctime = CURRENT_TIME;
692: i_new->dirty = 1;
693: dir_new->dirty = 1;
694:
695: dir_old->i_mtime = CURRENT_TIME;
696: dir_old->i_ctime = CURRENT_TIME;
697: i_old->dirty = 1;
698: dir_old->dirty = 1;
699: bwrite(buf_new);
700:
701: if(!buf_old) {
702:     if(!(buf_old = find_dir_entry(dir_old, i_old, &d_old, oldpath)))
{
703:         errno = -ENOENT;
704:         goto end;
705:     }
706: }
707: d_old->inode = 0;
708: bwrite(buf_old);
709:
710: /* update the parent directory */
711: if(S_ISDIR(i_old->i_mode)) {
712:     buf_new = find_dir_entry(i_old, dir_old, &d_new, "..");
713:     if(buf_new) {
714:         d_new->inode = dir_new->inode;
715:         bwrite(buf_new);
716:     }
717: }
718:
719: end:

```

fs/minix/namei.c

Page 12/12

```
720:         inode_unlock(i_old);
721:         inode_unlock(dir_old);
722:         inode_unlock(dir_new);
723:         return errno;
724: }
```

fs/minix/super.c

Page 1/5

```

1: /*
2:  * fiwix/fs/minix/super.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/types.h>
10: #include <fiwix/errno.h>
11: #include <fiwix/fs.h>
12: #include <fiwix/filesystems.h>
13: #include <fiwix/fs_minix.h>
14: #include <fiwix/buffer.h>
15: #include <fiwix/sched.h>
16: #include <fiwix/stdio.h>
17: #include <fiwix/string.h>
18:
19: struct fs_operations minix_fsop = {
20:     FSOP_REQUIRES_DEV,
21:     NULL,
22:
23:     NULL, /* open */
24:     NULL, /* close */
25:     NULL, /* read */
26:     NULL, /* write */
27:     NULL, /* ioctl */
28:     NULL, /* lseek */
29:     NULL, /* readdir */
30:     NULL, /* mmap */
31:     NULL, /* select */
32:
33:     NULL, /* readlink */
34:     NULL, /* followlink */
35:     NULL, /* bmap */
36:     NULL, /* lookup */
37:     NULL, /* rmdir */
38:     NULL, /* link */
39:     NULL, /* unlink */
40:     NULL, /* symlink */
41:     NULL, /* mkdir */
42:     NULL, /* mknod */
43:     NULL, /* truncate */
44:     NULL, /* create */
45:     NULL, /* rename */
46:
47:     NULL, /* read_block */
48:     NULL, /* write_block */
49:
50:     minix_read_inode,
51:     minix_write_inode,
52:     minix_ialloc,
53:     minix_ifree,
54:     minix_statfs,
55:     minix_read_superblock,
56:     minix_remount_fs,
57:     minix_write_superblock,
58:     minix_release_superblock
59: };
60:
61: static void check_superblock(struct minix_super_block *sb)
62: {
63:     if(!(sb->s_state & MINIX_VALID_FS)) {
64:         printk("WARNING: filesystem not checked, fsck recommended.\n");
65:     }
66:     if(sb->s_state & MINIX_ERROR_FS) {
67:         printk("WARNING: filesystem contains errors, fsck recommended.\n

```

fs/minix/super.c

Page 2/5

```

");
68:         }
69:     }
70:
71: void minix_statfs(struct superblock *sb, struct statfs *statfsbuf)
72: {
73:     statfsbuf->f_type = sb->u.minix.sb.s_magic;
74:     statfsbuf->f_bsize = sb->s_blocksize;
75:     statfsbuf->f_blocks = sb->u.minix.nzones << sb->u.minix.sb.s_log_zone_si
ze;
76:     statfsbuf->f_bfree = sb->u.minix.nzones - minix_count_free_blocks(sb);
77:     statfsbuf->f_bavail = statfsbuf->f_bfree;
78:
79:     statfsbuf->f_files = sb->u.minix.sb.s_ninodes;
80:     statfsbuf->f_ffree = sb->u.minix.sb.s_ninodes - minix_count_free_inodes(
sb);
81:     /* statfsbuf->f_fsid = ? */
82:     statfsbuf->f_namelen = sb->u.minix.namelen;
83: }
84:
85: int minix_read_superblock(__dev_t dev, struct superblock *sb)
86: {
87:     struct buffer *buf;
88:     int maps;
89:
90:     superblock_lock(sb);
91:     if(!(buf = bread(dev, SUPERBLOCK, BLKSIZE_1K))) {
92:         printk("WARNING: %s(): I/O error on device %d,%d.\n", __FUNCTION
__, MAJOR(dev), MINOR(dev));
93:         superblock_unlock(sb);
94:         return -EIO;
95:     }
96:     memcpy_b(&sb->u.minix.sb, buf->data, sizeof(struct minix_super_block));
97:
98:     switch(sb->u.minix.sb.s_magic) {
99:         case MINIX_SUPER_MAGIC:
100:             sb->u.minix.namelen = 14;
101:             sb->u.minix.dirsize = sizeof(__u16) + sb->u.minix.namele
n;
102:             sb->u.minix.version = 1;
103:             sb->u.minix.nzones = sb->u.minix.sb.s_nzones;
104:             printk("minix v1 (14 char names) filesystem detected on
device %d,%d.\n", MAJOR(dev), MINOR(dev));
105:             break;
106:         case MINIX_SUPER_MAGIC2:
107:             sb->u.minix.namelen = 30;
108:             sb->u.minix.dirsize = sizeof(__u16) + sb->u.minix.namele
n;
109:             sb->u.minix.version = 1;
110:             sb->u.minix.nzones = sb->u.minix.sb.s_nzones;
111:             printk("minix v1 (30 char names) filesystem detected on
device %d,%d.\n", MAJOR(dev), MINOR(dev));
112:             break;
113:         case MINIX2_SUPER_MAGIC:
114:             sb->u.minix.namelen = 14;
115:             sb->u.minix.dirsize = sizeof(__u16) + sb->u.minix.namele
n;
116:             sb->u.minix.version = 2;
117:             sb->u.minix.nzones = sb->u.minix.sb.s_zones;
118:             printk("minix v2 (14 char names) filesystem detected on
device %d,%d.\n", MAJOR(dev), MINOR(dev));
119:             break;
120:         case MINIX2_SUPER_MAGIC2:
121:             sb->u.minix.namelen = 30;
122:             sb->u.minix.dirsize = sizeof(__u16) + sb->u.minix.namele
n;
123:             sb->u.minix.version = 2;

```

fs/minix/super.c

Page 3/5

```

124:             sb->u.minix.nzones = sb->u.minix.sb.s_zones;
125:             printk("minix v2 (30 char names) filesystem detected on
device %d,%d.\n", MAJOR(dev), MINOR(dev));
126:             break;
127:             default:
128:             printk("ERROR: %s(): invalid filesystem type or bad supe
rblock on device %d,%d.\n", __FUNCTION__, MAJOR(dev), MINOR(dev));
129:             superblock_unlock(sb);
130:             brelse(buf);
131:             return -EINVAL;
132:         }
133:
134:         sb->dev = dev;
135:         sb->fsop = &minix_fsop;
136:         sb->s_blocksize = BLKSIZE_1K << sb->u.minix.sb.s_log_zone_size;
137:
138:         if(sb->s_blocksize != BLKSIZE_1K) {
139:             printk("ERROR: %s(): block sizes > %d not supported in this file
system.\n", __FUNCTION__, BLKSIZE_1K);
140:             superblock_unlock(sb);
141:             brelse(buf);
142:             return -EINVAL;
143:         }
144:
145:         /*
146:         printk("s_ninodes          = %d\n", sb->u.minix.sb.s_ninodes);
147:         printk("s_nzones          = %d (nzones = %d)\n", sb->u.minix.sb.s_nzones,
sb->u.minix.nzones);
148:         printk("s_imap_blocks     = %d\n", sb->u.minix.sb.s_imap_blocks);
149:         printk("s_zmap_blocks     = %d\n", sb->u.minix.sb.s_zmap_blocks);
150:         printk("s_firstdatazone = %d\n", sb->u.minix.sb.s_firstdatazone);
151:         printk("s_log_zone_size  = %d\n", sb->u.minix.sb.s_log_zone_size);
152:         printk("s_max_size       = %d\n", sb->u.minix.sb.s_max_size);
153:         printk("s_magic         = %x\n", sb->u.minix.sb.s_magic);
154:         printk("s_state         = %d\n", sb->u.minix.sb.s_state);
155:         printk("s_zones         = %d\n", sb->u.minix.sb.s_zones);
156:         */
157:
158:         /* Minix fs size is limited to: # of bitmaps * 8192 * 1024 */
159:         if(sb->u.minix.version == 1) {
160:             maps = V1_MAX_BITMAP_BLOCKS;    /* 64MB limit */
161:         }
162:         if(sb->u.minix.version == 2) {
163:             maps = V2_MAX_BITMAP_BLOCKS;    /* 1GB limit */
164:         }
165:
166:         if(sb->u.minix.sb.s_imap_blocks > maps) {
167:             printk("ERROR: %s(): number of imap blocks (%d) is greater than
%d!\n", __FUNCTION__, sb->u.minix.sb.s_imap_blocks, maps);
168:             superblock_unlock(sb);
169:             brelse(buf);
170:             return -EINVAL;
171:         }
172:         if(sb->u.minix.sb.s_zmap_blocks > maps) {
173:             printk("ERROR: %s(): number of zmap blocks (%d) is greater than
%d!\n", __FUNCTION__, sb->u.minix.sb.s_zmap_blocks, maps);
174:             superblock_unlock(sb);
175:             brelse(buf);
176:             return -EINVAL;
177:         }
178:
179:         superblock_unlock(sb);
180:
181:         if(!(sb->root = iget(sb, MINIX_ROOT_INO))) {
182:             printk("ERROR: %s(): unable to get root inode.\n", __FUNCTION__)
;
183:             brelse(buf);

```

fs/minix/super.c

Page 4/5

```

184:         return -EINVAL;
185:     }
186:
187:     check_superblock(&sb->u.minix.sb);
188:
189:     if(!(sb->flags & MS_RDONLY)) {
190:         sb->u.minix.sb.s_state &= ~MINIX_VALID_FS;
191:         memcpy_b(buf->data, &sb->u.minix.sb, sizeof(struct minix_super_b
lock));
192:         bwrite(buf);
193:     } else {
194:         brelse(buf);
195:     }
196:
197:     return 0;
198: }
199:
200: int minix_remount_fs(struct superblock *sb, int flags)
201: {
202:     struct buffer *buf;
203:     struct minix_super_block *minixsb;
204:
205:     if((flags & MS_RDONLY) == (sb->flags & MS_RDONLY)) {
206:         return 0;
207:     }
208:
209:     superblock_lock(sb);
210:     if!(buf = bread(sb->dev, SUPERBLOCK, BLKSIZE_1K)) {
211:         superblock_unlock(sb);
212:         return -EIO;
213:     }
214:     minixsb = (struct minix_super_block *)buf->data;
215:
216:     if(flags & MS_RDONLY) {
217:         /* switching from RW to RO */
218:         sb->u.minix.sb.s_state |= MINIX_VALID_FS;
219:         minixsb->s_state |= MINIX_VALID_FS;
220:     } else {
221:         /* switching from RO to RW */
222:         check_superblock(minixsb);
223:         memcpy_b(&sb->u.minix.sb, minixsb, sizeof(struct minix_super_blo
ck));
224:         sb->u.minix.sb.s_state &= ~MINIX_VALID_FS;
225:         minixsb->s_state &= ~MINIX_VALID_FS;
226:     }
227:
228:     sb->dirty = 1;
229:     superblock_unlock(sb);
230:     bwrite(buf);
231:     return 0;
232: }
233:
234: int minix_write_superblock(struct superblock *sb)
235: {
236:     struct buffer *buf;
237:
238:     superblock_lock(sb);
239:     if!(buf = bread(sb->dev, SUPERBLOCK, BLKSIZE_1K)) {
240:         superblock_unlock(sb);
241:         return -EIO;
242:     }
243:
244:     memcpy_b(buf->data, &sb->u.minix.sb, sizeof(struct minix_super_block));
245:     sb->dirty = 0;
246:     superblock_unlock(sb);
247:     bwrite(buf);
248:     return 0;

```

fs/minix/super.c

Page 5/5

```
249: }
250:
251: void minix_release_superblock(struct superblock *sb)
252: {
253:     if (sb->flags & MS_RDONLY) {
254:         return;
255:     }
256:
257:     superblock_lock(sb);
258:
259:     sb->u.minix.sb.s_state |= MINIX_VALID_FS;
260:     sb->dirty = 1;
261:
262:     superblock_unlock(sb);
263: }
264:
265: int minix_init(void)
266: {
267:     return register_filesystem("minix", &minix_fsop);
268: }
```

fs/minix/symlink.c

Page 1/3

```

1: /*
2:  * fiwix/fs/minix/symlink.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/errno.h>
10: #include <fiwix/buffer.h>
11: #include <fiwix/fs.h>
12: #include <fiwix/filesystems.h>
13: #include <fiwix/stat.h>
14: #include <fiwix/stdio.h>
15: #include <fiwix/string.h>
16:
17: struct fs_operations minix_symlink_fsop = {
18:     0,
19:     0,
20:
21:     NULL,          /* open */
22:     NULL,          /* close */
23:     NULL,          /* read */
24:     NULL,          /* write */
25:     NULL,          /* ioctl */
26:     NULL,          /* lseek */
27:     NULL,          /* readdir */
28:     NULL,          /* mmap */
29:     NULL,          /* select */
30:
31:     minix_readlink,
32:     minix_followlink,
33:     NULL,          /* bmap */
34:     NULL,          /* lookup */
35:     NULL,          /* rmdir */
36:     NULL,          /* link */
37:     NULL,          /* unlink */
38:     NULL,          /* symlink */
39:     NULL,          /* mkdir */
40:     NULL,          /* mknod */
41:     NULL,          /* truncate */
42:     NULL,          /* create */
43:     NULL,          /* rename */
44:
45:     NULL,          /* read_block */
46:     NULL,          /* write_block */
47:
48:     NULL,          /* read_inode */
49:     NULL,          /* write_inode */
50:     NULL,          /* ialloc */
51:     NULL,          /* ifree */
52:     NULL,          /* statfs */
53:     NULL,          /* read_superblock */
54:     NULL,          /* remount_fs */
55:     NULL,          /* write_superblock */
56:     NULL,          /* release_superblock */
57: };
58:
59: int minix_readlink(struct inode *i, char *buffer, __size_t count)
60: {
61:     __u32 blksize;
62:     struct buffer *buf;
63:
64:     if(!S_ISLNK(i->i_mode)) {
65:         printk("%s(): Oops, inode '%d' is not a symlink (!?)\n", __FUNC
TION__, i->inode);
66:         return 0;

```


fs/minix/symlink.c

Page 2/3

```

67:         }
68:
69:         inode_lock(i);
70:         blksize = i->sb->s_blocksize;
71:         count = MIN(count, i->i_size);
72:         if(!count) {
73:             inode_unlock(i);
74:             return 0;
75:         }
76:         count = MIN(count, blksize);
77:         if(i->sb->u.minix.version == 1) {
78:             if(!(buf = bread(i->dev, i->u.minix.u.i1_zone[0], blksize))) {
79:                 inode_unlock(i);
80:                 return -EIO;
81:             }
82:         } else {
83:             if(!(buf = bread(i->dev, i->u.minix.u.i2_zone[0], blksize))) {
84:                 inode_unlock(i);
85:                 return -EIO;
86:             }
87:         }
88:         memcpy_b(buffer, buf->data, count);
89:         brelse(buf);
90:         buffer[count] = NULL;
91:         inode_unlock(i);
92:         return count;
93:     }
94:
95: int minix_followlink(struct inode *dir, struct inode *i, struct inode **i_res)
96: {
97:     struct buffer *buf;
98:     char *name;
99:     __ino_t errno;
100:
101:     if(!i) {
102:         return -ENOENT;
103:     }
104:
105:     if(!S_ISLNK(i->i_mode)) {
106:         printk("%s(): Oops, inode '%d' is not a symlink (!?)\n", __FUNC
TION__, i->inode);
107:         return 0;
108:     }
109:
110:     if(current->loopcnt > MAX_SYMLINKS) {
111:         printk("%s(): too many nested symbolic links!\n", __FUNCTION__);
112:         return -ELOOP;
113:     }
114:
115:     inode_lock(i);
116:     if(i->sb->u.minix.version == 1) {
117:         if(!(buf = bread(i->dev, i->u.minix.u.i1_zone[0], i->sb->s_block
size))) {
118:             inode_unlock(i);
119:             return -EIO;
120:         }
121:     } else {
122:         if(!(buf = bread(i->dev, i->u.minix.u.i2_zone[0], i->sb->s_block
size))) {
123:             inode_unlock(i);
124:             return -EIO;
125:         }
126:     }
127:     name = buf->data;
128:     inode_unlock(i);
129:
130:     current->loopcnt++;

```

fs/minix/symlink.c

Page 3/3

```
131:         iput(i);
132:         brelse(buf);
133:         errno = parse_namei(name, dir, i_res, NULL, FOLLOW_LINKS);
134:         current->loopcnt--;
135:         return errno;
136: }
```

fs/minix/v1_inode.c

Page 1/7

```

1: /*
2:  * fiwix/fs/minix/v1_inode.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/filesystems.h>
11: #include <fiwix/fs_minix.h>
12: #include <fiwix/fs_pipe.h>
13: #include <fiwix/statfs.h>
14: #include <fiwix/sleep.h>
15: #include <fiwix/stat.h>
16: #include <fiwix/sched.h>
17: #include <fiwix/buffer.h>
18: #include <fiwix/process.h>
19: #include <fiwix/errno.h>
20: #include <fiwix/stdio.h>
21: #include <fiwix/string.h>
22:
23: #define BLOCKS_PER_IND_BLOCK(sb)          (sb->s_blocksize / sizeof(__u16))
24: #define MINIX_INODES_PER_BLOCK(sb)       (sb->s_blocksize / sizeof(struct minix_i
node))
25:
26: #define MINIX_NDIR_BLOCKS                7
27: #define MINIX_IND_BLOCK                  MINIX_NDIR_BLOCKS
28: #define MINIX_DIND_BLOCK                  (MINIX_NDIR_BLOCKS + 1)
29:
30: static void free_zone(struct inode *i, int block, int offset)
31: {
32:     int n;
33:     struct buffer *buf;
34:     __u16 *zone;
35:
36:     if(!(buf = bread(i->dev, block, i->sb->s_blocksize))) {
37:         printk("WARNING: %s(): error reading block %d.\n", __FUNCTION__,
block);
38:         return;
39:     }
40:     zone = (__u16 *)buf->data;
41:     for(n = offset; n < BLOCKS_PER_IND_BLOCK(i->sb); n++) {
42:         if(zone[n]) {
43:             minix_bfree(i->sb, zone[n]);
44:             zone[n] = 0;
45:         }
46:     }
47:     bwrite(buf);
48: }
49:
50: int v1_minix_read_inode(struct inode *i)
51: {
52:     __ino_t block;
53:     short int offset;
54:     struct minix_inode *ii;
55:     struct buffer *buf;
56:     int errno;
57:
58:     block = 1 + SUPERBLOCK + i->sb->u.minix.sb.s_imap_blocks + i->sb->u.mini
x.sb.s_zmap_blocks + (i->inode - 1) / MINIX_INODES_PER_BLOCK(i->sb);
59:
60:     if(!(buf = bread(i->dev, block, i->sb->s_blocksize))) {
61:         return -EIO;
62:     }
63:     offset = (i->inode - 1) % MINIX_INODES_PER_BLOCK(i->sb);
64:     ii = ((struct minix_inode *)buf->data) + offset;

```

fs/minix/v1_inode.c

Page 2/7

```

65:
66:     i->i_mode = ii->i_mode;
67:     i->i_uid = ii->i_uid;
68:     i->i_size = ii->i_size;
69:     i->i_atime = ii->i_time;
70:     i->i_ctime = ii->i_time;
71:     i->i_mtime = ii->i_time;
72:     i->i_gid = ii->i_gid;
73:     i->i_nlink = ii->i_nlinks;
74:     memcpy_b(i->u.minix.u.il_zone, ii->i_zone, sizeof(ii->i_zone));
75:     i->count = 1;
76:
77:     errno = 0;
78:     switch(i->i_mode & S_IFMT) {
79:         case S_IFCHR:
80:             i->fsop = &def_chr_fsop;
81:             i->rdev = ii->i_zone[0];
82:             break;
83:         case S_IFBLK:
84:             i->fsop = &def_blk_fsop;
85:             i->rdev = ii->i_zone[0];
86:             break;
87:         case S_IFIFO:
88:             i->fsop = &pipefs_fsop;
89:             /* it's a union so we need to clear pipefs_i */
90:             memset_b(&i->u.pipefs, NULL, sizeof(struct pipefs_inode)
);
91:             break;
92:         case S_IFDIR:
93:             i->fsop = &minix_dir_fsop;
94:             break;
95:         case S_IFREG:
96:             i->fsop = &minix_file_fsop;
97:             break;
98:         case S_IFLNK:
99:             i->fsop = &minix_symlink_fsop;
100:            break;
101:         case S_IFSOCK:
102:             i->fsop = NULL;
103:             break;
104:         default:
105:             printk("WARNING: %s(): invalid inode (%d) mode %o.\n", _
_FUNCTION__, i->inode, i->i_mode);
106:             errno = -ENOENT;
107:             break;
108:     }
109:
110:     brelse(buf);
111:     return errno;
112: }
113:
114: int v1_minix_write_inode(struct inode *i)
115: {
116:     __ino_t block;
117:     short int offset;
118:     struct minix_inode *ii;
119:     struct buffer *buf;
120:
121:     block = 1 + SUPERBLOCK + i->sb->u.minix.sb.s_imap_blocks + i->sb->u.mini
x.sb.s_zmap_blocks + (i->inode - 1) / MINIX_INODES_PER_BLOCK(i->sb);
122:
123:     if(!(buf = bread(i->dev, block, i->sb->s_blocksize))) {
124:         return -EIO;
125:     }
126:     offset = (i->inode - 1) % MINIX_INODES_PER_BLOCK(i->sb);
127:     ii = ((struct minix_inode *)buf->data) + offset;
128:

```

fs/minix/v1_inode.c

Page 3/7

```

129:         ii->i_mode = i->i_mode;
130:         ii->i_uid = i->i_uid;
131:         ii->i_size = i->i_size;
132:         ii->i_time = i->i_mtime;
133:         ii->i_gid = i->i_gid;
134:         ii->i_nlinks = i->i_nlink;
135:         if(S_ISCHR(i->i_mode) || S_ISBLK(i->i_mode)) {
136:             ii->i_zone[0] = i->rdev;
137:         } else {
138:             memcpy_b(ii->i_zone, i->u.minix.u.i1_zone, sizeof(i->u.minix.u.i
139:         }
140:         i->dirty = 0;
141:         bwrite(buf);
142:         return 0;
143:     }
144:
145: int v1_minix_ialloc(struct inode *i, int mode)
146: {
147:     __blk_t offset;
148:     int inode, errno;
149:     struct superblock *sb;
150:
151:     sb = i->sb;
152:     superblock_lock(sb);
153:
154:     offset = 1 + SUPERBLOCK;
155:
156:     if(!(inode = minix_find_first_zero(sb, offset, sb->u.minix.sb.s_ninodes,
offset + sb->u.minix.sb.s_imap_blocks))) {
157:         superblock_unlock(sb);
158:         return -ENOSPC;
159:     }
160:
161:     errno = minix_change_bit(SET_BIT, sb, offset, inode);
162:
163:     if(errno) {
164:         if(errno < 0) {
165:             printk("WARNING: %s(): unable to set inode %d.\n", __FUN
CTION__, inode);
166:             superblock_unlock(sb);
167:             return errno;
168:         } else {
169:             printk("WARNING: %s(): inode %d is already marked as use
d!\n", __FUNCTION__, inode);
170:         }
171:     }
172:
173:     i->inode = inode;
174:     i->i_atime = CURRENT_TIME;
175:     i->i_mtime = CURRENT_TIME;
176:     i->i_ctime = CURRENT_TIME;
177:     superblock_unlock(sb);
178:     return 0;
179: }
180:
181: void v1_minix_ifree(struct inode *i)
182: {
183:     int errno;
184:     struct superblock *sb;
185:
186:     minix_truncate(i, 0);
187:
188:     sb = i->sb;
189:     superblock_lock(sb);
190:
191:     errno = minix_change_bit(CLEAR_BIT, i->sb, 1 + SUPERBLOCK, i->inode);

```

fs/minix/v1_inode.c

Page 4/7

```

192:
193:     if(errno) {
194:         if(errno < 0) {
195:             printk("WARNING: %s(): unable to clear inode %d.\n", __F
UNCTION__, i->inode);
196:         } else {
197:             printk("WARNING: %s(): inode %d is already marked as fre
e!\n", __FUNCTION__, i->inode);
198:         }
199:     }
200:
201:     i->i_size = 0;
202:     i->i_mtime = CURRENT_TIME;
203:     i->i_ctime = CURRENT_TIME;
204:     i->dirty = 1;
205:     superblock_unlock(sb);
206: }
207:
208: int v1_minix_bmap(struct inode *i, __off_t offset, int mode)
209: {
210:     unsigned char level;
211:     __u16 *indblock, *dindblock;
212:     __blk_t block, iblock, dblock, newblock;
213:     int blksize;
214:     struct buffer *buf, *buf2, *buf3;
215:
216:     blksize = i->sb->s_blocksize;
217:     block = offset / blksize;
218:     level = 0;
219:
220:     if(block < MINIX_NDIR_BLOCKS) {
221:         level = MINIX_NDIR_BLOCKS - 1;
222:     } else {
223:         if(block < (BLOCKS_PER_IND_BLOCK(i->sb) + MINIX_NDIR_BLOCKS)) {
224:             level = MINIX_IND_BLOCK;
225:         } else {
226:             level = MINIX_DIND_BLOCK;
227:         }
228:         block -= MINIX_NDIR_BLOCKS;
229:     }
230:
231:     if(level < MINIX_NDIR_BLOCKS) {
232:         if(!i->u.minix.u.il_zone[block] && mode == FOR_WRITING) {
233:             if((newblock = minix_balloc(i->sb)) < 0) {
234:                 return -ENOSPC;
235:             }
236:             /* initialize the new block */
237:             if(!(buf = bread(i->dev, newblock, blksize))) {
238:                 minix_bfree(i->sb, newblock);
239:                 return -EIO;
240:             }
241:             memset_b(buf->data, 0, blksize);
242:             bwrite(buf);
243:             i->u.minix.u.il_zone[block] = newblock;
244:         }
245:         return i->u.minix.u.il_zone[block];
246:     }
247:
248:     if(!i->u.minix.u.il_zone[level]) {
249:         if(mode == FOR_WRITING) {
250:             if((newblock = minix_balloc(i->sb)) < 0) {
251:                 return -ENOSPC;
252:             }
253:             /* initialize the new block */
254:             if(!(buf = bread(i->dev, newblock, blksize))) {
255:                 minix_bfree(i->sb, newblock);
256:                 return -EIO;

```

fs/minix/v1_inode.c

Page 5/7

```

257:         }
258:         memset_b(buf->data, 0, blksize);
259:         bwrite(buf);
260:         i->u.minix.u.il_zone[level] = newblock;
261:     } else {
262:         return 0;
263:     }
264: }
265: if(!(buf = bread(i->dev, i->u.minix.u.il_zone[level], blksize))) {
266:     return -EIO;
267: }
268: indblock = (__u16 *)buf->data;
269: dblock = block - BLOCKS_PER_IND_BLOCK(i->sb);
270:
271: if(level == MINIX_DIND_BLOCK) {
272:     block = dblock / BLOCKS_PER_IND_BLOCK(i->sb);
273: }
274:
275: if(!indblock[block]) {
276:     if(mode == FOR_WRITING) {
277:         if((newblock = minix_balloc(i->sb)) < 0) {
278:             brelse(buf);
279:             return -ENOSPC;
280:         }
281:         /* initialize the new block */
282:         if(!(buf2 = bread(i->dev, newblock, blksize))) {
283:             minix_bfree(i->sb, newblock);
284:             brelse(buf);
285:             return -EIO;
286:         }
287:         memset_b(buf2->data, 0, blksize);
288:         bwrite(buf2);
289:         indblock[block] = newblock;
290:         if(level == MINIX_IND_BLOCK) {
291:             bwrite(buf);
292:             return newblock;
293:         }
294:         buf->flags |= (BUFFER_DIRTY | BUFFER_VALID);
295:     } else {
296:         brelse(buf);
297:         return 0;
298:     }
299: }
300: if(level == MINIX_IND_BLOCK) {
301:     newblock = indblock[block];
302:     brelse(buf);
303:     return newblock;
304: }
305:
306: iblock = block;
307: if(!(buf2 = bread(i->dev, indblock[iblock], blksize))) {
308:     printk("%s(): returning -EIO\n", __FUNCTION__);
309:     brelse(buf);
310:     return -EIO;
311: }
312: dindblock = (__u16 *)buf2->data;
313: block = dindblock[dblock - (iblock * BLOCKS_PER_IND_BLOCK(i->sb))];
314: if(!block && mode == FOR_WRITING) {
315:     if((newblock = minix_balloc(i->sb)) < 0) {
316:         brelse(buf);
317:         brelse(buf2);
318:         return -ENOSPC;
319:     }
320:     /* initialize the new block */
321:     if(!(buf3 = bread(i->dev, newblock, blksize))) {
322:         minix_bfree(i->sb, newblock);
323:         brelse(buf);

```

fs/minix/v1_inode.c

Page 6/7

```

324:             brelse(buf2);
325:             return -EIO;
326:         }
327:         memset_b(buf3->data, 0, blksize);
328:         bwrite(buf3);
329:         dindblock[dblock - (iblock * BLOCKS_PER_IND_BLOCK(i->sb))] = new
block;
330:         buf2->flags |= (BUFFER_DIRTY | BUFFER_VALID);
331:         block = newblock;
332:     }
333:     brelse(buf);
334:     brelse(buf2);
335:     return block;
336: }
337:
338: int v1_minix_truncate(struct inode *i, __off_t length)
339: {
340:     int n;
341:     __blk_t block, dblock;
342:     __u16 *zone;
343:     struct buffer *buf;
344:
345:     block = length / i->sb->s_blocksize;
346:
347:     if(!S_ISDIR(i->i_mode) && !S_ISREG(i->i_mode) && !S_ISLNK(i->i_mode)) {
348:         return -EINVAL;
349:     }
350:
351:     if(block < MINIX_NDIR_BLOCKS) {
352:         for(n = block; n < MINIX_NDIR_BLOCKS; n++) {
353:             if(i->u.minix.u.il_zone[n]) {
354:                 minix_bfree(i->sb, i->u.minix.u.il_zone[n]);
355:                 i->u.minix.u.il_zone[n] = 0;
356:             }
357:         }
358:         block = 0;
359:     }
360:
361:     if(!block || block < (BLOCKS_PER_IND_BLOCK(i->sb) + MINIX_NDIR_BLOCKS))
{
362:         if(block) {
363:             block -= MINIX_NDIR_BLOCKS;
364:         }
365:         if(i->u.minix.u.il_zone[MINIX_IND_BLOCK]) {
366:             free_zone(i, i->u.minix.u.il_zone[MINIX_IND_BLOCK], bloc
k);
367:             if(!block) {
368:                 minix_bfree(i->sb, i->u.minix.u.il_zone[MINIX_IN
D_BLOCK]);
369:                 i->u.minix.u.il_zone[MINIX_IND_BLOCK] = 0;
370:             }
371:         }
372:         block = 0;
373:     }
374:
375:     if(block) {
376:         block -= MINIX_NDIR_BLOCKS;
377:         block -= BLOCKS_PER_IND_BLOCK(i->sb);
378:     }
379:     if(i->u.minix.u.il_zone[MINIX_DIND_BLOCK]) {
380:         if(!(buf = bread(i->dev, i->u.minix.u.il_zone[MINIX_DIND_BLOCK],
i->sb->s_blocksize))) {
381:             printk("%s(): error reading block %d.\n", __FUNCTION__,
i->u.minix.u.il_zone[MINIX_DIND_BLOCK]);
382:         }
383:         zone = (__u16 *)buf->data;
384:         dblock = block % BLOCKS_PER_IND_BLOCK(i->sb);

```


fs/minix/v1_inode.c

Page 7/7

```
385:          for(n = block / BLOCKS_PER_IND_BLOCK(i->sb); n < BLOCKS_PER_IND_
BLOCK(i->sb); n++) {
386:              if(zone[n]) {
387:                  free_zone(i, zone[n], dblock);
388:                  if(!dblock) {
389:                      minix_bfree(i->sb, zone[n]);
390:                  }
391:              }
392:              dblock = 0;
393:          }
394:          bwrite(buf);
395:          if(!block) {
396:              minix_bfree(i->sb, i->u.minix.u.il_zone[MINIX_DIND_BLOCK
]);
397:              i->u.minix.u.il_zone[MINIX_DIND_BLOCK] = 0;
398:          }
399:      }
400:
401:      i->i_mtime = CURRENT_TIME;
402:      i->i_ctime = CURRENT_TIME;
403:      i->i_size = length;
404:      i->dirty = 1;
405:
406:      return 0;
407: }
```

fs/minix/v2_inode.c

Page 1/8

```

1: /*
2:  * fiwix/fs/minix/v2_inode.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/filesystems.h>
11: #include <fiwix/fs_minix.h>
12: #include <fiwix/fs_pipe.h>
13: #include <fiwix/statfs.h>
14: #include <fiwix/sleep.h>
15: #include <fiwix/stat.h>
16: #include <fiwix/sched.h>
17: #include <fiwix/buffer.h>
18: #include <fiwix/process.h>
19: #include <fiwix/errno.h>
20: #include <fiwix/stdio.h>
21: #include <fiwix/string.h>
22:
23: #define BLOCKS_PER_IND_BLOCK(sb)          (sb->s_blocksize / sizeof(__u32))
24: #define MINIX2_INODES_PER_BLOCK(sb)      (sb->s_blocksize / sizeof(struct minix2_
inode))
25:
26: #define MINIX_NDIR_BLOCKS                7
27: #define MINIX_IND_BLOCK                  MINIX_NDIR_BLOCKS
28: #define MINIX_DIND_BLOCK                  (MINIX_NDIR_BLOCKS + 1)
29: #define MINIX_TIND_BLOCK                  (MINIX_NDIR_BLOCKS + 2)
30:
31: static void free_zone(struct inode *i, int block, int offset)
32: {
33:     int n;
34:     struct buffer *buf;
35:     __u32 *zone;
36:
37:     if(!(buf = bread(i->dev, block, i->sb->s_blocksize))) {
38:         printk("WARNING: %s(): error reading block %d.\n", __FUNCTION__,
block);
39:         return;
40:     }
41:     zone = (__u32 *)buf->data;
42:     for(n = offset; n < BLOCKS_PER_IND_BLOCK(i->sb); n++) {
43:         if(zone[n]) {
44:             minix_bfree(i->sb, zone[n]);
45:             zone[n] = 0;
46:         }
47:     }
48:     bwrite(buf);
49: }
50:
51: int v2_minix_read_inode(struct inode *i)
52: {
53:     __ino_t block;
54:     short int offset;
55:     struct minix2_inode *ii;
56:     struct buffer *buf;
57:     int errno;
58:
59:     block = 1 + SUPERBLOCK + i->sb->u.minix.sb.s_imap_blocks + i->sb->u.mini
x.sb.s_zmap_blocks + (i->inode - 1) / MINIX2_INODES_PER_BLOCK(i->sb);
60:
61:     if(!(buf = bread(i->dev, block, i->sb->s_blocksize))) {
62:         return -EIO;
63:     }
64:     offset = (i->inode - 1) % MINIX2_INODES_PER_BLOCK(i->sb);

```

fs/minix/v2_inode.c

Page 2/8

```

65:         ii = ((struct minix2_inode *)buf->data) + offset;
66:
67:         i->i_mode = ii->i_mode;
68:         i->i_nlink = ii->i_nlink;
69:         i->i_uid = ii->i_uid;
70:         i->i_gid = ii->i_gid;
71:         i->i_size = ii->i_size;
72:         i->i_atime = ii->i_atime;
73:         i->i_mtime = ii->i_mtime;
74:         i->i_ctime = ii->i_ctime;
75:         memcpy_b(i->u.minix.u.i2_zone, ii->i_zone, sizeof(ii->i_zone));
76:         i->count = 1;
77:
78:         errno = 0;
79:         switch(i->i_mode & S_IFMT) {
80:             case S_IFCHR:
81:                 i->fsop = &def_chr_fsop;
82:                 i->rdev = ii->i_zone[0];
83:                 break;
84:             case S_IFBLK:
85:                 i->fsop = &def_blk_fsop;
86:                 i->rdev = ii->i_zone[0];
87:                 break;
88:             case S_IFIFO:
89:                 i->fsop = &pipefs_fsop;
90:                 /* it's a union so we need to clear pipefs_i */
91:                 memset_b(&i->u.pipefs, NULL, sizeof(struct pipefs_inode)
);
92:                 break;
93:             case S_IFDIR:
94:                 i->fsop = &minix_dir_fsop;
95:                 break;
96:             case S_IFREG:
97:                 i->fsop = &minix_file_fsop;
98:                 break;
99:             case S_IFLNK:
100:                i->fsop = &minix_symlink_fsop;
101:                break;
102:             case S_IFSOCK:
103:                i->fsop = NULL;
104:                break;
105:             default:
106:                printk("WARNING: %s(): invalid inode (%d) mode %o.\n", _
_FUNCTION__, i->inode, i->i_mode);
107:                errno = -ENOENT;
108:                break;
109:         }
110:
111:         brelse(buf);
112:         return errno;
113: }
114:
115: int v2_minix_write_inode(struct inode *i)
116: {
117:     __ino_t block;
118:     short int offset;
119:     struct minix2_inode *ii;
120:     struct buffer *buf;
121:
122:     block = 1 + SUPERBLOCK + i->sb->u.minix.sb.s_imap_blocks + i->sb->u.mini
x.sb.s_zmap_blocks + (i->inode - 1) / MINIX2_INODES_PER_BLOCK(i->sb);
123:
124:     if (!(buf = bread(i->dev, block, i->sb->s_blocksize))) {
125:         return -EIO;
126:     }
127:     offset = (i->inode - 1) % MINIX2_INODES_PER_BLOCK(i->sb);
128:     ii = ((struct minix2_inode *)buf->data) + offset;

```

fs/minix/v2_inode.c

Page 3/8

```

129:
130:     ii->i_mode = i->i_mode;
131:     ii->i_nlink = i->i_nlink;
132:     ii->i_uid = i->i_uid;
133:     ii->i_gid = i->i_gid;
134:     ii->i_size = i->i_size;
135:     ii->i_atime = i->i_atime;
136:     ii->i_mtime = i->i_mtime;
137:     ii->i_ctime = i->i_ctime;
138:     if(S_ISCHR(i->i_mode) || S_ISBLK(i->i_mode)) {
139:         ii->i_zone[0] = i->rdev;
140:     } else {
141:         memcpy_b(ii->i_zone, i->u.minix.u.i2_zone, sizeof(i->u.minix.u.i
2_zone));
142:     }
143:     i->dirty = 0;
144:     bwrite(buf);
145:     return 0;
146: }
147:
148: int v2_minix_ialloc(struct inode *i, int mode)
149: {
150:     __blk_t offset;
151:     int inode, errno;
152:     struct superblock *sb;
153:
154:     sb = i->sb;
155:     superblock_lock(sb);
156:
157:     offset = 1 + SUPERBLOCK;
158:
159:     if(!(inode = minix_find_first_zero(sb, offset, sb->u.minix.sb.s_ninodes,
offset + sb->u.minix.sb.s_imap_blocks))) {
160:         superblock_unlock(sb);
161:         return -ENOSPC;
162:     }
163:
164:     errno = minix_change_bit(SET_BIT, sb, offset, inode);
165:
166:     if(errno) {
167:         if(errno < 0) {
168:             printk("WARNING: %s(): unable to set inode %d.\n", __FUN
CTION__, inode);
169:             superblock_unlock(sb);
170:             return errno;
171:         } else {
172:             printk("WARNING: %s(): inode %d is already marked as use
d!\n", __FUNCTION__, inode);
173:         }
174:     }
175:
176:     i->inode = inode;
177:     i->i_atime = CURRENT_TIME;
178:     i->i_mtime = CURRENT_TIME;
179:     i->i_ctime = CURRENT_TIME;
180:     superblock_unlock(sb);
181:     return 0;
182: }
183:
184: void v2_minix_ifree(struct inode *i)
185: {
186:     int errno;
187:     struct superblock *sb;
188:
189:     minix_truncate(i, 0);
190:
191:     sb = i->sb;

```

fs/minix/v2_inode.c

Page 4/8

```

192:         superblock_lock(sb);
193:
194:         errno = minix_change_bit(CLEAR_BIT, i->sb, 1 + SUPERBLOCK, i->inode);
195:
196:         if(errno) {
197:             if(errno < 0) {
198:                 printk("WARNING: %s(): unable to clear inode %d.\n", __F
UNCTION__, i->inode);
199:             } else {
200:                 printk("WARNING: %s(): inode %d is already marked as fre
e!\n", __FUNCTION__, i->inode);
201:             }
202:         }
203:
204:         i->i_size = 0;
205:         i->i_mtime = CURRENT_TIME;
206:         i->i_ctime = CURRENT_TIME;
207:         i->dirty = 1;
208:         superblock_unlock(sb);
209:     }
210:
211: int v2_minix_bmap(struct inode *i, __off_t offset, int mode)
212: {
213:     unsigned char level;
214:     __u32 *indblock, *dindblock, *tindblock;
215:     __blk_t block, iblock, dblock, tblock, newblock;
216:     int blksize;
217:     struct buffer *buf, *buf2, *buf3, *buf4;
218:
219:     blksize = i->sb->s_blocksize;
220:     block = offset / blksize;
221:     level = 0;
222:     buf3 = NULL;    /* makes GCC happy */
223:
224:     if(block < MINIX_NDIR_BLOCKS) {
225:         level = MINIX_NDIR_BLOCKS - 1;
226:     } else {
227:         if(block < (BLOCKS_PER_IND_BLOCK(i->sb) + MINIX_NDIR_BLOCKS)) {
228:             level = MINIX_IND_BLOCK;
229:         } else if(block < ((BLOCKS_PER_IND_BLOCK(i->sb) * BLOCKS_PER_IND
_BLOCK(i->sb)) + BLOCKS_PER_IND_BLOCK(i->sb) + MINIX_NDIR_BLOCKS)) {
230:             level = MINIX_DIND_BLOCK;
231:         } else {
232:             level = MINIX_TIND_BLOCK;
233:         }
234:         block -= MINIX_NDIR_BLOCKS;
235:     }
236:
237:     if(level < MINIX_NDIR_BLOCKS) {
238:         if(!i->u.minix.u.i2_zone[block] && mode == FOR_WRITING) {
239:             if((newblock = minix_balloc(i->sb)) < 0) {
240:                 return -ENOSPC;
241:             }
242:             /* initialize the new block */
243:             if(!(buf = bread(i->dev, newblock, blksize))) {
244:                 minix_bfree(i->sb, newblock);
245:                 return -EIO;
246:             }
247:             memset_b(buf->data, 0, blksize);
248:             bwrite(buf);
249:             i->u.minix.u.i2_zone[block] = newblock;
250:         }
251:         return i->u.minix.u.i2_zone[block];
252:     }
253:
254:     if(!i->u.minix.u.i2_zone[level]) {
255:         if(mode == FOR_WRITING) {

```

fs/minix/v2_inode.c

Page 5/8

```

256:             if((newblock = minix_balloc(i->sb)) < 0) {
257:                 return -ENOSPC;
258:             }
259:             /* initialize the new block */
260:             if(!(buf = bread(i->dev, newblock, blksize))) {
261:                 minix_bfree(i->sb, newblock);
262:                 return -EIO;
263:             }
264:             memset_b(buf->data, 0, blksize);
265:             bwrite(buf);
266:             i->u.minix.u.i2_zone[level] = newblock;
267:         } else {
268:             return 0;
269:         }
270:     }
271:     if(!(buf = bread(i->dev, i->u.minix.u.i2_zone[level], blksize))) {
272:         return -EIO;
273:     }
274:     indblock = (__u32 *)buf->data;
275:     dblock = block - BLOCKS_PER_IND_BLOCK(i->sb);
276:     tblock = block - (BLOCKS_PER_IND_BLOCK(i->sb) * BLOCKS_PER_IND_BLOCK(i->
sb)) - BLOCKS_PER_IND_BLOCK(i->sb);
277:
278:     if(level == MINIX_DIND_BLOCK) {
279:         block = dblock / BLOCKS_PER_IND_BLOCK(i->sb);
280:     }
281:     if(level == MINIX_TIND_BLOCK) {
282:         block = tblock / (BLOCKS_PER_IND_BLOCK(i->sb) * BLOCKS_PER_IND_B
LOCK(i->sb));
283:     }
284:
285:     if(!indblock[block]) {
286:         if(mode == FOR_WRITING) {
287:             if((newblock = minix_balloc(i->sb)) < 0) {
288:                 brelse(buf);
289:                 return -ENOSPC;
290:             }
291:             /* initialize the new block */
292:             if(!(buf2 = bread(i->dev, newblock, blksize))) {
293:                 minix_bfree(i->sb, newblock);
294:                 brelse(buf);
295:                 return -EIO;
296:             }
297:             memset_b(buf2->data, 0, blksize);
298:             bwrite(buf2);
299:             indblock[block] = newblock;
300:             if(level == MINIX_IND_BLOCK) {
301:                 bwrite(buf);
302:                 return newblock;
303:             }
304:             buf->flags |= (BUFFER_DIRTY | BUFFER_VALID);
305:         } else {
306:             brelse(buf);
307:             return 0;
308:         }
309:     }
310:     if(level == MINIX_IND_BLOCK) {
311:         newblock = indblock[block];
312:         brelse(buf);
313:         return newblock;
314:     }
315:
316:     if(level == MINIX_TIND_BLOCK) {
317:         if(!(buf3 = bread(i->dev, indblock[block], blksize))) {
318:             printk("%s(): returning -EIO\n", __FUNCTION__);
319:             brelse(buf);
320:             return -EIO;

```

fs/minix/v2_inode.c

Page 6/8

```

321:         }
322:         tindblock = (__u32 *)buf3->data;
323:         block = tindblock[tblock / BLOCKS_PER_IND_BLOCK(i->sb)];
324:         if(!block) {
325:             if(mode == FOR_WRITING) {
326:                 if((newblock = minix_balloc(i->sb)) < 0) {
327:                     brelse(buf);
328:                     brelse(buf3);
329:                     return -ENOSPC;
330:                 }
331:                 /* initialize the new block */
332:                 if(!(buf4 = bread(i->dev, newblock, blksize))) {
333:                     minix_bfree(i->sb, newblock);
334:                     brelse(buf);
335:                     brelse(buf3);
336:                     return -EIO;
337:                 }
338:                 memset_b(buf4->data, 0, blksize);
339:                 bwrite(buf4);
340:                 tindblock[tblock / BLOCKS_PER_IND_BLOCK(i->sb)]
= newblock;
341:                 buf3->flags |= (BUFFER_DIRTY | BUFFER_VALID);
342:                 block = newblock;
343:             } else {
344:                 brelse(buf);
345:                 brelse(buf3);
346:                 return 0;
347:             }
348:         }
349:         dblock = tblock;
350:         iblock = tblock / BLOCKS_PER_IND_BLOCK(i->sb);
351:         if(!(buf2 = bread(i->dev, block, blksize))) {
352:             printk("%s(): returning -EIO\n", __FUNCTION__);
353:             brelse(buf);
354:             brelse(buf3);
355:             return -EIO;
356:         }
357:     } else {
358:         iblock = block;
359:         if(!(buf2 = bread(i->dev, indblock[iblock], blksize))) {
360:             printk("%s(): returning -EIO\n", __FUNCTION__);
361:             brelse(buf);
362:             return -EIO;
363:         }
364:     }
365:
366:     dindblock = (__u32 *)buf2->data;
367:     block = dindblock[dblock - (iblock * BLOCKS_PER_IND_BLOCK(i->sb))];
368:     if(!block && mode == FOR_WRITING) {
369:         if((newblock = minix_balloc(i->sb)) < 0) {
370:             brelse(buf);
371:             if(level == MINIX_TIND_BLOCK) {
372:                 brelse(buf3);
373:             }
374:             brelse(buf2);
375:             return -ENOSPC;
376:         }
377:         /* initialize the new block */
378:         if(!(buf4 = bread(i->dev, newblock, blksize))) {
379:             minix_bfree(i->sb, newblock);
380:             brelse(buf);
381:             if(level == MINIX_TIND_BLOCK) {
382:                 brelse(buf3);
383:             }
384:             brelse(buf2);
385:             return -EIO;
386:         }

```

fs/minix/v2_inode.c

Page 7/8

```

387:         memset_b(buf4->data, 0, blksize);
388:         bwrite(buf4);
389:         dindblock[dblock - (iblock * BLOCKS_PER_IND_BLOCK(i->sb))] = new
block;
390:         buf2->flags |= (BUFFER_DIRTY | BUFFER_VALID);
391:         block = newblock;
392:     }
393:     brelease(buf);
394:     if(level == MINIX_TIND_BLOCK) {
395:         brelease(buf3);
396:     }
397:     brelease(buf2);
398:     return block;
399: }
400:
401: int v2_minix_truncate(struct inode *i, __off_t length)
402: {
403:     int n;
404:     __blk_t block, dblock;
405:     __u32 *zone;
406:     struct buffer *buf;
407:
408:     block = length / i->sb->s_blocksize;
409:
410:     if(!S_ISDIR(i->i_mode) && !S_ISREG(i->i_mode) && !S_ISLNK(i->i_mode)) {
411:         return -EINVAL;
412:     }
413:
414:     if(block < MINIX_NDIR_BLOCKS) {
415:         for(n = block; n < MINIX_NDIR_BLOCKS; n++) {
416:             if(i->u.minix.u.i2_zone[n]) {
417:                 minix_bfree(i->sb, i->u.minix.u.i2_zone[n]);
418:                 i->u.minix.u.i2_zone[n] = 0;
419:             }
420:         }
421:         block = 0;
422:     }
423:
424:     if(!block || block < (BLOCKS_PER_IND_BLOCK(i->sb) + MINIX_NDIR_BLOCKS))
{
425:         if(block) {
426:             block -= MINIX_NDIR_BLOCKS;
427:         }
428:         if(i->u.minix.u.i2_zone[MINIX_IND_BLOCK]) {
429:             free_zone(i, i->u.minix.u.i2_zone[MINIX_IND_BLOCK], bloc
k);
430:
431:             if(!block) {
432:                 minix_bfree(i->sb, i->u.minix.u.i2_zone[MINIX_IN
D_BLOCK]);
433:                 i->u.minix.u.i2_zone[MINIX_IND_BLOCK] = 0;
434:             }
435:             block = 0;
436:         }
437:
438:         if(block) {
439:             block -= MINIX_NDIR_BLOCKS;
440:             block -= BLOCKS_PER_IND_BLOCK(i->sb);
441:         }
442:         if(i->u.minix.u.i2_zone[MINIX_DIND_BLOCK]) {
443:             if(!(buf = bread(i->dev, i->u.minix.u.i2_zone[MINIX_DIND_BLOCK],
i->sb->s_blocksize))) {
444:                 printk("%s(): error reading block %d.\n", __FUNCTION__,
i->u.minix.u.i2_zone[MINIX_DIND_BLOCK]);
445:             }
446:             zone = (__u32 *)buf->data;
447:             dblock = block % BLOCKS_PER_IND_BLOCK(i->sb);

```


fs/minix/v2_inode.c

Page 8/8

```
448:          for(n = block / BLOCKS_PER_IND_BLOCK(i->sb); n < BLOCKS_PER_IND_
BLOCK(i->sb); n++) {
449:              if(zone[n]) {
450:                  free_zone(i, zone[n], dblock);
451:                  if(!dblock) {
452:                      minix_bfree(i->sb, zone[n]);
453:                  }
454:              }
455:              dblock = 0;
456:          }
457:          bwrite(buf);
458:          if(!block) {
459:              minix_bfree(i->sb, i->u.minix.u.i2_zone[MINIX_DIND_BLOCK
]);
460:              i->u.minix.u.i2_zone[MINIX_DIND_BLOCK] = 0;
461:          }
462:      }
463:
464:      i->i_mtime = CURRENT_TIME;
465:      i->i_ctime = CURRENT_TIME;
466:      i->i_size = length;
467:      i->dirty = 1;
468:
469:      return 0;
470: }
```

fs/pipefs/fifo.c

Page 1/2

```

1: /*
2:  * fiwix/fs/pipefs/fifo.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/errno.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/filesystems.h>
12: #include <fiwix/fs_pipe.h>
13: #include <fiwix/stat.h>
14: #include <fiwix/mm.h>
15: #include <fiwix/sleep.h>
16: #include <fiwix/fcntl.h>
17: #include <fiwix/sched.h>
18: #include <fiwix/stdio.h>
19:
20: int fifo_open(struct inode *i, struct fd *fd_table)
21: {
22:     /* first open */
23:     if(i->count == 1) {
24:         if(!(i->u.pipefs.i_data = (void *)kmalloc())) {
25:             return -ENOMEM;
26:         }
27:         i->u.pipefs.i_readoff = 0;
28:         i->u.pipefs.i_writeoff = 0;
29:     }
30:
31:     if((fd_table->flags & O_ACCMODE) == O_RDONLY) {
32:         i->u.pipefs.i_readers++;
33:         wakeup(&pipefs_write);
34:         if(!(fd_table->flags & O_NONBLOCK)) {
35:             while(!i->u.pipefs.i_writers) {
36:                 if(sleep(&pipefs_read, PROC_INTERRUPTIBLE)) {
37:                     if(--i->u.pipefs.i_readers) {
38:                         wakeup(&pipefs_write);
39:                     }
40:                     return -EINTR;
41:                 }
42:             }
43:         }
44:     }
45:
46:     if((fd_table->flags & O_ACCMODE) == O_WRONLY) {
47:         if((fd_table->flags & O_NONBLOCK) && !i->u.pipefs.i_readers) {
48:             return -ENXIO;
49:         }
50:
51:         i->u.pipefs.i_writers++;
52:         wakeup(&pipefs_read);
53:         if(!(fd_table->flags & O_NONBLOCK)) {
54:             while(!i->u.pipefs.i_readers) {
55:                 if(sleep(&pipefs_write, PROC_INTERRUPTIBLE)) {
56:                     if(--i->u.pipefs.i_writers) {
57:                         wakeup(&pipefs_read);
58:                     }
59:                     return -EINTR;
60:                 }
61:             }
62:         }
63:     }
64:
65:     if((fd_table->flags & O_ACCMODE) == O_RDWR) {
66:         i->u.pipefs.i_readers++;
67:         i->u.pipefs.i_writers++;

```

fs/pipefs/fifo.c

Page 2/2

```
68:                wakeup(&pipefs_write);
69:                wakeup(&pipefs_read);
70:            }
71:
72:            return 0;
73: }
```

fs/pipefs/Makefile

Page 1/1

```
1: # fiwix/fs/pipefs/Makefile
2: #
3: # Copyright 2018, Jordi Sanfeliu. All rights reserved.
4: # Distributed under the terms of the Fiwix License.
5: #
6:
7: .S.o:
8:     $(CC) -traditional -I$(INCLUDE) -c -o $@ $<
9: .c.o:
10:     $(CC) $(CFLAGS) -c -o $@ $<
11:
12: OBJS = super.o fifo.o pipe.o
13:
14: pipefs: $(OBJS)
15:     $(LD) $(LDFLAGS) -r $(OBJS) -o pipefs.o
16:
17: clean:
18:     rm -f *.o
19:
```

fs/pipefs/pipe.c

Page 1/4

```

1: /*
2:  * fiwix/fs/pipefs/pipe.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/errno.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/filesystems.h>
12: #include <fiwix/fs_pipe.h>
13: #include <fiwix/stat.h>
14: #include <fiwix/fcntl.h>
15: #include <fiwix/ioctl.h>
16: #include <fiwix/sleep.h>
17: #include <fiwix/sched.h>
18: #include <fiwix/stdio.h>
19: #include <fiwix/string.h>
20:
21: static struct resource pipe_resource = { NULL, NULL };
22:
23: int pipefs_close(struct inode *i, struct fd *fd_table)
24: {
25:     if((fd_table->flags & O_ACCMODE) == O_RDONLY) {
26:         if(!--i->u.pipefs.i_readers) {
27:             wakeup(&do_select);
28:             wakeup(&pipefs_write);
29:         }
30:     }
31:     if((fd_table->flags & O_ACCMODE) == O_WRONLY) {
32:         if(!--i->u.pipefs.i_writers) {
33:             wakeup(&do_select);
34:             wakeup(&pipefs_read);
35:         }
36:     }
37:     if((fd_table->flags & O_ACCMODE) == O_RDWR) {
38:         if(!--i->u.pipefs.i_readers) {
39:             wakeup(&do_select);
40:             wakeup(&pipefs_write);
41:         }
42:         if(!--i->u.pipefs.i_writers) {
43:             wakeup(&do_select);
44:             wakeup(&pipefs_read);
45:         }
46:     }
47:     return 0;
48: }
49:
50: int pipefs_read(struct inode *i, struct fd *fd_table, char *buffer, __size_t count)
51: {
52:     __off_t bytes_read;
53:     __size_t n, limit;
54:     char *data;
55:
56:     bytes_read = 0;
57:     data = i->u.pipefs.i_data;
58:
59:     while(count) {
60:         if(i->u.pipefs.i_writeoff) {
61:             if(i->u.pipefs.i_readoff >= i->u.pipefs.i_writeoff) {
62:                 limit = PIPE_BUF - i->u.pipefs.i_readoff;
63:             } else {
64:                 limit = i->u.pipefs.i_writeoff - i->u.pipefs.i_readoff;
65:             }

```

fs/pipefs/pipe.c

Page 2/4

```

66:         } else {
67:             limit = PIPE_BUF - i->u.pipefs.i_readoff;
68:         }
69:         n = MIN(limit, count);
70:         if(i->i_size && n) {
71:             lock_resource(&pipe_resource);
72:             memcpy_b(buffer + bytes_read, data + i->u.pipefs.i_reado
ff, n);
73:             bytes_read += n;
74:             i->u.pipefs.i_readoff += n;
75:             i->i_size -= n;
76:             if(i->u.pipefs.i_writeoff >= PIPE_BUF) {
77:                 i->u.pipefs.i_writeoff = 0;
78:             }
79:             unlock_resource(&pipe_resource);
80:             wakeup(&do_select);
81:             wakeup(&pipefs_write);
82:             break;
83:         } else {
84:             if(i->u.pipefs.i_writers) {
85:                 if(fd_table->flags & O_NONBLOCK) {
86:                     return -EAGAIN;
87:                 }
88:                 if(sleep(&pipefs_read, PROC_INTERRUPTIBLE)) {
89:                     return -EINTR;
90:                 }
91:             } else {
92:                 if(i->i_size) {
93:                     if(i->u.pipefs.i_readoff >= PIPE_BUF) {
94:                         i->u.pipefs.i_readoff = 0;
95:                         continue;
96:                     }
97:                 }
98:                 break;
99:             }
100:         }
101:     }
102:     if(!i->i_size) {
103:         i->u.pipefs.i_readoff = 0;
104:         i->u.pipefs.i_writeoff = 0;
105:     }
106:     return bytes_read;
107: }
108:
109: int pipefs_write(struct inode *i, struct fd *fd_table, const char *buffer, __siz
e_t count)
110: {
111:     __off_t bytes_written;
112:     __size_t n;
113:     char *data;
114:     int limit;
115:
116:     bytes_written = 0;
117:     data = i->u.pipefs.i_data;
118:
119:     while(bytes_written < count) {
120:         /* if the read end closes then send signal and return */
121:         if(!i->u.pipefs.i_readers) {
122:             send_sig(current, SIGPIPE);
123:             return -EPIPE;
124:         }
125:
126:         if(i->u.pipefs.i_readoff) {
127:             if(i->u.pipefs.i_writeoff <= i->u.pipefs.i_readoff) {
128:                 limit = i->u.pipefs.i_readoff;
129:             } else {
130:                 limit = PIPE_BUF;

```

fs/pipefs/pipe.c

Page 3/4

```

131:         }
132:     } else {
133:         limit = PIPE_BUF;
134:     }
135:
136:     n = MIN((count - bytes_written), (limit - i->u.pipefs.i_writeoff
));
137:
138:     /*
139:      * POSIX requires that any write operation involving fewer than
140:      * PIPE_BUF bytes must be automatically executed and finished
141:      * without being interleaved with write operations of other
142:      * processes to the same pipe.
143:      */
144:     if(n && n <= PIPE_BUF) {
145:         lock_resource(&pipe_resource);
146:         memcpy_b(data + i->u.pipefs.i_writeoff, buffer + bytes_w
ritten, n);
147:         bytes_written += n;
148:         i->u.pipefs.i_writeoff += n;
149:         i->i_size += n;
150:         if(i->u.pipefs.i_readoff >= PIPE_BUF) {
151:             i->u.pipefs.i_readoff = 0;
152:         }
153:         unlock_resource(&pipe_resource);
154:         wakeup(&do_select);
155:         wakeup(&pipefs_read);
156:         continue;
157:     }
158:
159:     wakeup(&do_select);
160:     wakeup(&pipefs_read);
161:     if(!(fd_table->flags & O_NONBLOCK)) {
162:         if(sleep(&pipefs_write, PROC_INTERRUPTIBLE)) {
163:             return -EINTR;
164:         }
165:     } else {
166:         return -EAGAIN;
167:     }
168: }
169: return bytes_written;
170: }
171:
172: int pipefs_ioctl(struct inode *i, int cmd, unsigned long int arg)
173: {
174:     int errno;
175:
176:     switch(cmd) {
177:         case FIONREAD:
178:             if((errno = check_user_area(VERIFY_WRITE, (void *)arg, s
izeof(unsigned int)))) {
179:                 return errno;
180:             }
181:             memcpy_b((void *)arg, &i->i_size, sizeof(unsigned int));
182:             break;
183:         default:
184:             return -EINVAL;
185:     }
186:     return 0;
187: }
188:
189: int pipefs_lseek(struct inode *i, __off_t offset)
190: {
191:     return -ESPIPE;
192: }
193:
194: int pipefs_select(struct inode *i, int flag)

```

fs/pipefs/pipe.c

Page 4/4

```
195: {
196:     switch(flag) {
197:         case SEL_R:
198:             /*
199:              * if !i->i_size && !i->u.pipefs.i_writers
200:              * should also return 1?
201:              */
202:             if(i->i_size || !i->u.pipefs.i_writers) {
203:                 return 1;
204:             }
205:             break;
206:         case SEL_W:
207:             /*
208:              * if i->i_size == PIPE_BUF && !i->u.pipefs.i_readers
209:              * should also return 1?
210:              */
211:             if(i->i_size < PIPE_BUF || !i->u.pipefs.i_readers) {
212:                 return 1;
213:             }
214:             break;
215:     }
216:     return 0;
217: }
```


fs/pipefs/super.c

Page 1/2

```

1: /*
2:  * fiwix/fs/pipefs/super.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/errno.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/filesystems.h>
12: #include <fiwix/fs_pipe.h>
13: #include <fiwix/stat.h>
14: #include <fiwix/mm.h>
15: #include <fiwix/sched.h>
16: #include <fiwix/stdio.h>
17: #include <fiwix/string.h>
18:
19: static unsigned int i_counter;
20:
21: struct fs_operations pipefs_fsop = {
22:     FSOP_KERN_MOUNT,
23:     PIPE_DEV,
24:
25:     fifo_open,
26:     pipefs_close,
27:     pipefs_read,
28:     pipefs_write,
29:     pipefs_ioctl,
30:     pipefs_lseek,
31:     NULL,                /* readdir */
32:     NULL,                /* mmap */
33:     pipefs_select,
34:
35:     NULL,                /* readlink */
36:     NULL,                /* followlink */
37:     NULL,                /* bmap */
38:     NULL,                /* lookup */
39:     NULL,                /* rmdir */
40:     NULL,                /* link */
41:     NULL,                /* unlink */
42:     NULL,                /* symlink */
43:     NULL,                /* mkdir */
44:     NULL,                /* mknod */
45:     NULL,                /* truncate */
46:     NULL,                /* create */
47:     NULL,                /* rename */
48:
49:     NULL,                /* read_block */
50:     NULL,                /* write_block */
51:
52:     NULL,                /* read_inode */
53:     NULL,                /* write_inode */
54:     pipefs_ialloc,
55:     pipefs_ifree,
56:     NULL,                /* statfs */
57:     pipefs_read_superblock,
58:     NULL,                /* remount_fs */
59:     NULL,                /* write_superblock */
60:     NULL,                /* release_superblock */
61: };
62:
63: int pipefs_read_superblock(__dev_t dev, struct superblock *sb)
64: {
65:     superblock_lock(sb);
66:     sb->dev = dev;
67:     sb->fsop = &pipefs_fsop;

```

fs/pipefs/super.c

Page 2/2

```
68:         sb->s_blocksize = BLKSIZE_1K;
69:         i_counter = 0;
70:         superblock_unlock(sb);
71:         return 0;
72: }
73:
74: int pipefs_ialloc(struct inode *i, int mode)
75: {
76:     struct superblock *sb = i->sb;
77:
78:     superblock_lock(sb);
79:     i_counter++;
80:     superblock_unlock(sb);
81:
82:     i->i_mode = S_IFIFO;
83:     i->dev = i->rdev = sb->dev;
84:     i->fsop = &pipefs_fsop;
85:     i->inode = i_counter;
86:     i->count = 2;
87:     if(!(i->u.pipefs.i_data = (void *)kmalloc())) {
88:         return -ENOMEM;
89:     }
90:     i->u.pipefs.i_readoff = 0;
91:     i->u.pipefs.i_writeoff = 0;
92:     i->u.pipefs.i_readers = 1;
93:     i->u.pipefs.i_writers = 1;
94:     return 0;
95: }
96:
97: void pipefs_ifree(struct inode *i)
98: {
99:     if(!i->u.pipefs.i_readers && !i->u.pipefs.i_writers) {
100:         /*
101:          * We need to ask before to kfree() because this function is
102:          * also called to free removed (with sys_unlink) fifo files.
103:          */
104:         if(i->u.pipefs.i_data) {
105:             kfree((unsigned int)i->u.pipefs.i_data);
106:         }
107:     }
108: }
109:
110: int pipefs_init(void)
111: {
112:     return register_filesystem("pipefs", &pipefs_fsop);
113: }
```

fs/procfs/data.c

Page 1/14

```

1: /*
2:  * fiwix/fs/procfs/data.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/system.h>
10: #include <fiwix/types.h>
11: #include <fiwix/process.h>
12: #include <fiwix/cmos.h>
13: #include <fiwix/dma.h>
14: #include <fiwix/ide.h>
15: #include <fiwix/fs.h>
16: #include <fiwix/filesystems.h>
17: #include <fiwix/devices.h>
18: #include <fiwix/locks.h>
19: #include <fiwix/mm.h>
20: #include <fiwix/mman.h>
21: #include <fiwix/fs_proc.h>
22: #include <fiwix/cpu.h>
23: #include <fiwix/pic.h>
24: #include <fiwix/sched.h>
25: #include <fiwix/timer.h>
26: #include <fiwix/utsname.h>
27: #include <fiwix/version.h>
28: #include <fiwix/errno.h>
29: #include <fiwix/stdio.h>
30: #include <fiwix/string.h>
31:
32: #define FSHIFT16          16
33: #define FIXED16_1        (1 << FSHIFT16)
34: #define LOAD_INT(x)      ((x) >> FSHIFT16)
35: #define LOAD_FRAC(x)     LOAD_INT(((x) & (FIXED16_1 - 1)) * 100)
36:
37: static const char *pstate[] = {
38:     "? (unused!)",
39:     "R (running)",
40:     "S (sleeping)",
41:     "Z (zombie)",
42:     "T (stopped)",
43:     "D (idle)",
44: };
45:
46: /*
47:  * procfs root directory related functions
48:  * -----
49:  */
50: int data_proc_self(char *buffer, __pid_t pid)
51: {
52:     return sprintk(buffer, "%s", current->pidstr);
53: }
54:
55: int data_proc_cmdline(char *buffer, __pid_t pid)
56: {
57:     return sprintk(buffer, "%s\n", cmdline);
58: }
59:
60: int data_proc_cpuinfo(char *buffer, __pid_t pid)
61: {
62:     int size;
63:
64:     size = sprintk(buffer, "processor          : 0\n");
65:     size += sprintk(buffer + size, "cpu family          : %d86\n", cpu_table.family
ily <= 6 ? cpu_table.family : 6);
66:     if(cpu_table.model >= 0) {

```

fs/procfs/data.c

Page 2/14

```

67:             size += sprintf(buffer + size, "model           : %d\n", cpu_tab
le.model);
68:         } else {
69:             size += sprintf(buffer + size, "model           : unknown\n");
70:         }
71:
72:         if(cpu_table.vendor_id) {
73:             size += sprintf(buffer + size, "vendor_id        : %s\n", cpu_tab
le.vendor_id);
74:         }
75:         if(cpu_table.model_name) {
76:             size += sprintf(buffer + size, "model name       : %s\n", cpu_tab
le.model_name);
77:         }
78:         if(cpu_table.stepping >= 0) {
79:             size += sprintf(buffer + size, "stepping         : %d\n", cpu_tab
le.stepping);
80:         } else {
81:             size += sprintf(buffer + size, "stepping         : unknown\n");
82:         }
83:
84:         size += sprintf(buffer + size, "cpu MHz           : ");
85:         if(cpu_table.hz) {
86:             size += sprintf(buffer + size, "%d.%d\n", (cpu_table.hz / 100000
0), ((cpu_table.hz % 1000000) / 100000));
87:         } else {
88:             size += sprintf(buffer + size, "unknown\n");
89:         }
90:         if(cpu_table.cache) {
91:             size += sprintf(buffer + size, "cache size      : %s\n", cpu_tab
le.cache);
92:         }
93:         size += sprintf(buffer + size, "cpuid             : %s\n", cpu_table.has_cp
uid ? "yes" : "no");
94:         size += sprintf(buffer + size, "fpu               : %s\n", cpu_table.has_fp
u ? "yes" : "no");
95:         size += get_cpu_flags(buffer, size);
96:         return size;
97:     }
98:
99: int data_proc_devices(char *buffer, __pid_t pid)
100: {
101:     int n, size;
102:     struct device *d;
103:
104:     size = sprintf(buffer, "Character devices:\n");
105:     for(n = 0; n < NR_CHRDEV; n++) {
106:         d = chr_device_table[n];
107:         while(d) {
108:             size += sprintf(buffer + size, "%3d %s\n", d->major, d->
name);
109:             d = d->next;
110:         }
111:     }
112:
113:     size += sprintf(buffer + size, "\nBlock devices:\n");
114:     for(n = 0; n < NR_BLKDEV; n++) {
115:         d = blk_device_table[n];
116:         while(d) {
117:             size += sprintf(buffer + size, "%3d %s\n", d->major, d->
name);
118:             d = d->next;
119:         }
120:     }
121:     return size;
122: }
123:

```

fs/procfs/data.c

Page 3/14

```

124: int data_proc_dma(char *buffer, __pid_t pid)
125: {
126:     int n, size;
127:
128:     size = 0;
129:     for(n = 0; n < DMA_CHANNELS; n++) {
130:         if(dma_resources[n]) {
131:             size += sprintf(buffer + size, "%2d: %s\n", n, dma_resou
rces[n]);
132:         }
133:     }
134:     return size;
135: }
136:
137: int data_proc_filesystems(char *buffer, __pid_t pid)
138: {
139:     int n, size;
140:     int nodev;
141:
142:     size = 0;
143:     for(n = 0; n < NR_FILESYSTEMS; n++) {
144:         if(filesystems_table[n].name) {
145:             nodev = 0;
146:             if(filesystems_table[n].fsop->flags != FSOP_REQUIRES_DEV
) {
147:                 nodev = 1;
148:             }
149:             size += sprintf(buffer + size, "%s %s\n", nodev ? "nodev
" : "", filesystems_table[n].name);
150:         }
151:     }
152:     return size;
153: }
154:
155: int data_proc_interrupts(char *buffer, __pid_t pid)
156: {
157:     struct interrupt *irq;
158:     int n, size;
159:
160:     size = 0;
161:     for(n = 0; n < NR_IRQS; n++) {
162:         if((irq = irq_table[n])) {
163:             size += sprintf(buffer + size, "%3d: %9u %s", n, irq->ti
cks, irq->name);
164:             while((irq = irq->next)) {
165:                 size += sprintf(buffer + size, ",%s", irq->name)
;
166:             }
167:             size += sprintf(buffer + size, "\n");
168:         }
169:     }
170:     size += sprintf(buffer + size, "SPU: %9u %s\n", kstat.sirqs, "Spurious i
nterrupts");
171:     return size;
172: }
173:
174: int data_proc_loadavg(char *buffer, __pid_t pid)
175: {
176:     int a, b, c;
177:     int size;
178:     struct proc *p;
179:     int nrun = 0;
180:     int nprocs = 0;
181:
182:     a = avenrun[0] << (SI_LOAD_SHIFT - FSHIFT);
183:     b = avenrun[1] << (SI_LOAD_SHIFT - FSHIFT);
184:     c = avenrun[2] << (SI_LOAD_SHIFT - FSHIFT);

```

```

185:
186:     FOR_EACH_PROCESS(p) {
187:         nprocs++;
188:         if(p->state == PROC_RUNNING) {
189:             nrun++;
190:         }
191:         p = p->next;
192:     }
193:
194:     size = sprintf(buffer, "%d.%02d %d.%02d %d.%02d %d/%d %d\n", LOAD_INT(a)
, LOAD_FRAC(a), LOAD_INT(b), LOAD_FRAC(b), LOAD_INT(c), LOAD_FRAC(c), nrun, nprocs, las
tpid);
195:     return size;
196: }
197:
198: int data_proc_locks(char *buffer, __pid_t pid)
199: {
200:     int n, size;
201:     struct flock_file *ff;
202:
203:     size = 0;
204:
205:     for(n = 0; n < NR_FLOCKS; n++) {
206:         ff = &flock_file_table[n];
207:         if(ff->inode) {
208:             size += sprintf(buffer + size, "%d: FLOCK ADVISORY %s
", n + 1, ff->type & LOCK_SH ? "READ " : "WRITE");
209:             size += sprintf(buffer + size, "%d %x:%d:%d 0 EOF\n", ff
->proc->pid, MAJOR(ff->inode->dev), MINOR(ff->inode->dev), ff->inode->inode);
210:         }
211:     }
212:
213:     return size;
214: }
215:
216: int data_proc_meminfo(char *buffer, __pid_t pid)
217: {
218:     struct page *pg;
219:     int n, size;
220:
221:     kstat.shared = 0;
222:     for(n = 0; n < kstat.physical_pages; n++) {
223:         pg = &page_table[n];
224:         if(pg->flags & PAGE_RESERVED) {
225:             continue;
226:         }
227:         if(!pg->count) {
228:             continue;
229:         }
230:         kstat.shared += pg->count - 1;
231:     }
232:
233:     size = 0;
234:     size += sprintf(buffer + size, "          total:    used:    free:    shared
: buffers:  cached:\n");
235:     size += sprintf(buffer + size, "Mem:  %8u %8u %8u %8u %8u %8u\n", kstat.
total_mem_pages << PAGE_SHIFT, (kstat.total_mem_pages << PAGE_SHIFT) - (kstat.free_page
s << PAGE_SHIFT), kstat.free_pages << PAGE_SHIFT, kstat.shared * 1024, kstat.buffers *
1024, kstat.cached * 1024);
236:     size += sprintf(buffer + size, "Swap: %8u %8u %8u\n", 0, 0, 0);
237:     size += sprintf(buffer + size, "MemTotal: %9d kB\n", kstat.total_mem_pag
es << 2);
238:     size += sprintf(buffer + size, "MemFree:  %9d kB\n", kstat.free_pages <<
2);
239:     size += sprintf(buffer + size, "MemShared:%9d kB\n", kstat.shared);
240:     size += sprintf(buffer + size, "Buffers:  %9d kB\n", kstat.buffers);
241:     size += sprintf(buffer + size, "Cached:   %9d kB\n", kstat.cached);

```

fs/procfs/data.c

Page 5/14

```

242:         size += sprintf(buffer + size, "SwapTotal:%9d kB\n", 0);
243:         size += sprintf(buffer + size, "SwapFree: %9d kB\n", 0);
244:         size += sprintf(buffer + size, "Dirty:    %9d kB\n", kstat.dirty);
245:         return size;
246:     }
247:
248: int data_proc_mounts(char *buffer, __pid_t pid)
249: {
250:     int n, size;
251:     char *flag;
252:
253:     size = 0;
254:     for(n = 0; n < NR_MOUNT_POINTS; n++) {
255:         if(mount_table[n].used) {
256:             if(mount_table[n].fs->fsop->flags != FSOP_KERN_MOUNT) {
257:                 flag = "rw";
258:                 if(mount_table[n].sb.flags & MS_RDONLY) {
259:                     flag = "ro";
260:                 }
261:                 size += sprintf(buffer + size, "%s %s %s %s 0 0\
n", mount_table[n].devname, mount_table[n].dirname, mount_table[n].fs->name, flag);
262:             }
263:         }
264:     }
265:     return size;
266: }
267:
268: int data_proc_partitions(char *buffer, __pid_t pid)
269: {
270:     int n, ctrl, drv, size;
271:     int minor, major;
272:     unsigned int blocks;
273:     struct ide *ide;
274:     struct ide_drv *drive;
275:
276:     size = 0;
277:     size += sprintf(buffer + size, "major minor #blocks name\n\n");
278:
279:     for(ctrl = 0; ctrl < NR_IDE_CTRL; ctrl++) {
280:         ide = &ide_table[ctrl];
281:         for(drv = 0; drv < NR_IDE_DRVS; drv++) {
282:             drive = &ide->drive[drv];
283:             if(!drive->nr_sects) {
284:                 continue;
285:             }
286:             if(drive->flags & DEVICE_IS_DISK) {
287:                 major = (int)drive->major;
288:                 minor = (int)drive->minor_shift;
289:                 blocks = drive->nr_sects / 2;
290:                 size += sprintf(buffer + size, "%4d %4d %9d %s
\n", major, 0, blocks, drive->dev_name);
291:                 for(n = 0; n < NR_PARTITIONS; n++) {
292:                     if(drive->part_table[n].type) {
293:                         blocks = drive->part_table[n].nr
_sects / 2;
294:                         size += sprintf(buffer + size, "
%4d %4d %9u %s%d\n", major, (n + 1) << minor, blocks, drive->dev_name, n + 1);
295:                     }
296:                 }
297:             }
298:         }
299:     }
300:     return size;
301: }
302:
303: int data_proc_rtc(char *buffer, __pid_t pid)
304: {

```

fs/procfs/data.c

Page 6/14

```

305:         int size;
306:         short int sec, min, hour;
307:         short int day, month, year, century;
308:
309:         sec = cmos_read_date(CMOS_SEC);
310:         min = cmos_read_date(CMOS_MIN);
311:         hour = cmos_read_date(CMOS_HOUR);
312:         day = cmos_read_date(CMOS_DAY);
313:         month = cmos_read_date(CMOS_MONTH);
314:         year = cmos_read_date(CMOS_YEAR);
315:         century = cmos_read_date(CMOS_CENTURY);
316:         year += century * 100;
317:
318:         size = 0;
319:         size += sprintf(buffer + size, "rtc_time\t: %02d:%02d:%02d\n", hour, min
, sec);
320:         size += sprintf(buffer + size, "rtc_date\t: %02d-%02d-%02d\n", year, mon
th, day);
321:         sec = cmos_read_date(CMOS_ASEC);
322:         min = cmos_read_date(CMOS_AMIN);
323:         hour = cmos_read_date(CMOS_AHOUR);
324:         size += sprintf(buffer + size, "alarm\t\t: %02d:%02d:%02d\n", hour, min,
sec);
325:         size += sprintf(buffer + size, "DST_enable\t: %s\n", cmos_read(CMOS_STAT
B) & CMOS_STATB_DSE ? "yes" : "no");
326:         size += sprintf(buffer + size, "BCD\t\t: %s\n", cmos_read(CMOS_STATB) &
CMOS_STATB_DM ? "no" : "yes");
327:         size += sprintf(buffer + size, "24hr\t\t: %s\n", cmos_read(CMOS_STATB) &
CMOS_STATB_24H ? "yes" : "no");
328:         size += sprintf(buffer + size, "square_wave\t: %s\n", cmos_read(CMOS_STA
TB) & CMOS_STATB_SQWE ? "yes" : "no");
329:         size += sprintf(buffer + size, "alarm_IRQ\t: %s\n", cmos_read(CMOS_STATB
) & CMOS_STATB_AIE ? "yes" : "no");
330:         size += sprintf(buffer + size, "update_IRQ\t: %s\n", cmos_read(CMOS_STAT
B) & CMOS_STATB_UIE ? "yes" : "no");
331:         size += sprintf(buffer + size, "periodic_IRQ\t: %s\n", cmos_read(CMOS_ST
ATB) & CMOS_STATB_PIE ? "yes" : "no");
332:         size += sprintf(buffer + size, "periodic_freq\t: %s\n", (cmos_read(CMOS_
STATA) & CMOS_STATATA_IRQF) == 0x6 ? "1024" : "?");
333:         size += sprintf(buffer + size, "batt_status\t: %s\n", cmos_read(CMOS_STA
TD) & CMOS_STATD_VRT ? "okay" : "dead");
334:         return size;
335:     }
336:
337: int data_proc_stat(char *buffer, __pid_t pid)
338: {
339:     int n, size;
340:     unsigned int idle;
341:     struct interrupt *irq;
342:
343:     idle = kstat.ticks - (kstat.cpu_user + kstat.cpu_nice + kstat.cpu_system
);
344:     size = 0;
345:     size += sprintf(buffer + size, "cpu %d %d %d %d\n", kstat.cpu_user, ksta
t.cpu_nice, kstat.cpu_system, idle);
346:     size += sprintf(buffer + size, "disk 0 0 0 0\n");
347:     size += sprintf(buffer + size, "page 0 0\n");
348:     size += sprintf(buffer + size, "swap 0 0\n");
349:     size += sprintf(buffer + size, "intr %u", kstat.irqs);
350:     for(n = 0; n < NR_IRQS; n++) {
351:         irq = irq_table[n];
352:         if(irq) {
353:             size += sprintf(buffer + size, " %u", irq->ticks);
354:         }
355:     }
356:     size += sprintf(buffer + size, "\n");
357:     size += sprintf(buffer + size, "ctxt %u\n", kstat.ctxt);

```


fs/procfs/data.c

Page 7/14

```

358:         size += sprintf(buffer + size, "btime %d\n", kstat.boot_time);
359:         size += sprintf(buffer + size, "processes %d\n", kstat.processes);
360:         return size;
361:     }
362:
363: int data_proc_uptime(char *buffer, __pid_t pid)
364: {
365:     struct proc *p;
366:     unsigned long int idle;
367:
368:     p = &proc_table[IDLE];
369:     idle = tv2ticks(&p->usage.ru_utime);
370:     idle += tv2ticks(&p->usage.ru_stime);
371:     return sprintf(buffer, "%u.%02u %u.%02u\n", kstat.uptime, kstat.ticks %
HZ, idle / HZ, idle % HZ);
372: }
373:
374: int data_proc_fullversion(char *buffer, __pid_t pid)
375: {
376:     return sprintf(buffer, "Fiwix version %s %s\n", UTS_RELEASE, UTS_VERSION
);
377: }
378:
379: int data_proc_domainname(char *buffer, __pid_t pid)
380: {
381:     return sprintf(buffer, "%s\n", sys_utsname.domainname);
382: }
383:
384: int data_proc_filemax(char *buffer, __pid_t pid)
385: {
386:     return sprintf(buffer, "%d\n", NR_OPENS);
387: }
388:
389: int data_proc_filenr(char *buffer, __pid_t pid)
390: {
391:     int n, nr;
392:
393:     nr = 0;
394:     for(n = 1; n < NR_OPENS; n++) {
395:         if(fd_table[n].count != 0) {
396:             nr++;
397:         }
398:     }
399:     return sprintf(buffer, "%d\n", nr);
400: }
401:
402: int data_proc_hostname(char *buffer, __pid_t pid)
403: {
404:     return sprintf(buffer, "%s\n", sys_utsname.nodename);
405: }
406:
407: int data_proc_inodemax(char *buffer, __pid_t pid)
408: {
409:     return sprintf(buffer, "%d\n", inode_table_size / sizeof(struct inode));
410: }
411:
412: int data_proc_inodenr(char *buffer, __pid_t pid)
413: {
414:     return sprintf(buffer, "%d\n", (inode_table_size / sizeof(struct inode))
- kstat.free_inodes);
415: }
416:
417: int data_proc_osrelease(char *buffer, __pid_t pid)
418: {
419:     return sprintf(buffer, "%s\n", UTS_RELEASE);
420: }
421:

```

fs/procfs/data.c

Page 8/14

```

422: int data_proc_ostype(char *buffer, __pid_t pid)
423: {
424:     return sprintk(buffer, "%s\n", UTS_SYSNAME);
425: }
426:
427: int data_proc_version(char *buffer, __pid_t pid)
428: {
429:     return sprintk(buffer, "%s\n", UTS_VERSION);
430: }
431:
432:
433: /*
434:  * PID directory related functions
435:  * -----
436:  */
437: int data_proc_pid_cmdline(char *buffer, __pid_t pid)
438: {
439:     int n, size;
440:     char *arg;
441:     char **argv;
442:     unsigned int addr, offset;
443:     struct proc *p;
444:
445:     size = 0;
446:     if((p = get_proc_by_pid(pid)) {
447:         for(n = 0; n < p->argc && (p->argv + n); n++) {
448:             argv = p->argv + n;
449:             offset = (int)argv & ~PAGE_MASK;
450:             addr = get_mapped_addr(p, (int)argv) & PAGE_MASK;
451:             addr = P2V(addr);
452:             argv = (char **) (addr + offset);
453:             offset = (int)argv[0] & ~PAGE_MASK;
454:             addr = get_mapped_addr(p, (int)argv[0]) & PAGE_MASK;
455:             addr = P2V(addr);
456:             arg = (char *) (addr + offset);
457:             if(size + strlen(arg) < (PAGE_SIZE - 1)) {
458:                 size += sprintk(buffer + size, "%s", arg);
459:                 buffer[size++] = NULL;
460:             } else {
461:                 break;
462:             }
463:         }
464:     }
465:     return size;
466: }
467:
468: int data_proc_pid_cwd(char *buffer, __pid_t pid)
469: {
470:     int size;
471:     struct proc *p;
472:     struct inode *i;
473:
474:     size = 0;
475:     if((p = get_proc_by_pid(pid)) {
476:
477:         /* zombie processes don't have current working directory */
478:         if(!p->pwd) {
479:             return -ENOENT;
480:         }
481:
482:         i = p->pwd;
483:         size = sprintk(buffer, "[%02d%02d]:%d", MAJOR(i->rdev), MINOR(i-
>rdev), i->inode);
484:     }
485:     return size;
486: }
487:

```

fs/procfs/data.c

Page 9/14

```

488: int data_proc_pid_environ(char *buffer, __pid_t pid)
489: {
490:     int n, size;
491:     char *env;
492:     char **envp;
493:     unsigned int addr, offset;
494:     struct proc *p;
495:
496:     size = 0;
497:     if((p = get_proc_by_pid(pid)) {
498:         for(n = 0; n < p->envc && (p->envp + n); n++) {
499:             envp = p->envp + n;
500:             offset = (int)envp & ~PAGE_MASK;
501:             addr = get_mapped_addr(p, (int)envp) & PAGE_MASK;
502:             addr = P2V(addr);
503:             envp = (char **) (addr + offset);
504:             offset = (int)envp[0] & ~PAGE_MASK;
505:             addr = get_mapped_addr(p, (int)envp[0]) & PAGE_MASK;
506:             addr = P2V(addr);
507:             env = (char *) (addr + offset);
508:             if(size + strlen(env) < (PAGE_SIZE - 1)) {
509:                 size += sprintf(buffer + size, "%s", env);
510:                 buffer[size++] = NULL;
511:             } else {
512:                 break;
513:             }
514:         }
515:     }
516:     return size;
517: }
518:
519: int data_proc_pid_exe(char *buffer, __pid_t pid)
520: {
521:     int size;
522:     struct proc *p;
523:     struct inode *i;
524:
525:     size = 0;
526:     if((p = get_proc_by_pid(pid)) {
527:
528:         /* kernel and zombie processes are programless */
529:         if(!p->vma || !p->vma->inode) {
530:             return -ENOENT;
531:         }
532:
533:         i = p->vma->inode;
534:         size = sprintf(buffer, "[%02d%02d]:%d", MAJOR(i->rdev), MINOR(i-
>rdev), i->inode);
535:     }
536:     return size;
537: }
538:
539: int data_proc_pid_maps(char *buffer, __pid_t pid)
540: {
541:     unsigned int n;
542:     int size, len;
543:     __ino_t inode;
544:     int major, minor;
545:     char *section;
546:     char r, w, x, f;
547:     struct proc *p;
548:     struct vma *vma;
549:
550:     size = 0;
551:     if((p = get_proc_by_pid(pid)) {
552:         if(!p->vma) {
553:             return 0;

```

fs/procfs/data.c

Page 10/14

```

554:         }
555:         vma = p->vma;
556:         for(n = 0; n < VMA_REGIONS && vma->start; n++, vma++) {
557:             r = vma->prot & PROT_READ ? 'r' : '-';
558:             w = vma->prot & PROT_WRITE ? 'w' : '-';
559:             x = vma->prot & PROT_EXEC ? 'x' : '-';
560:             if(vma->flags & MAP_SHARED) {
561:                 f = 's';
562:             } else if(vma->flags & MAP_PRIVATE) {
563:                 f = 'p';
564:             } else {
565:                 f = '-';
566:             }
567:             switch(vma->s_type) {
568:                 case P_TEXT:      section = "text";
569:                                 break;
570:                 case P_DATA:      section = "data";
571:                                 break;
572:                 case P_BSS:       section = "bss";
573:                                 break;
574:                 case P_HEAP:      section = "heap";
575:                                 break;
576:                 case P_STACK:    section = "stack";
577:                                 break;
578:                 case P_MMAP:      section = "mmap";
579:                                 break;
580:                 default:
581:                     section = NULL;
582:                     break;
583:             }
584:             inode = major = minor = 0;
585:             if(vma->inode) {
586:                 inode = vma->inode->inode;
587:                 major = MAJOR(vma->inode->dev);
588:                 minor = MINOR(vma->inode->dev);
589:             }
590:             len = sprintf(buffer + size, "%08x-%08x %c%c%c%c %08x %0
2d:%02d %- 10u [%s]\n", vma->start, vma->end, r, w, x, f, vma->offset, major, minor, in
ode, section);
591:             size += len;
592:         }
593:     }
594:     return size;
595: }
596:
597: int data_proc_pid_mountinfo(char *buffer, __pid_t pid)
598: {
599:     int n, size;
600:     char *flag, *devname;
601:
602:     size = 0;
603:     for(n = 0; n < NR_MOUNT_POINTS; n++) {
604:         if(mount_table[n].used) {
605:             if(mount_table[n].fs->fsop->flags != FSOP_KERN_MOUNT) {
606:                 flag = "rw";
607:                 if(mount_table[n].sb.flags & MS_RDONLY) {
608:                     flag = "ro";
609:                 }
610:                 devname = mount_table[n].devname;
611:                 if(!strcmp(mount_table[n].devname, "/dev/root"))
612:                     devname = _rootdevname;
613:             }
614:             size += sprintf(buffer + size, "%d 0 %d:%d %s %s
%s - %s %s %s\n", n, MAJOR(mount_table[n].dev), MINOR(mount_table[n].dev), "/", mount_
table[n].dirname, flag, mount_table[n].fs->name, devname, flag);
615:         }

```

fs/procfs/data.c

Page 11/14

```

616:         }
617:     }
618:     return size;
619: }
620:
621: int data_proc_pid_root(char *buffer, __pid_t pid)
622: {
623:     int size;
624:     struct proc *p;
625:     struct inode *i;
626:
627:     size = 0;
628:     if((p = get_proc_by_pid(pid)) {
629:
630:         /* zombie processes don't have root directory */
631:         if(!p->root) {
632:             return -ENOENT;
633:         }
634:
635:         i = p->root;
636:         size = sprintf(buffer, "[%02d%02d]:%d", MAJOR(i->rdev), MINOR(i-
>rdev), i->inode);
637:     }
638:     return size;
639: }
640:
641: int data_proc_pid_stat(char *buffer, __pid_t pid)
642: {
643:     int n, size, vma_start, vma_end;
644:     unsigned int esp, eip;
645:     int signum, mask;
646:     __sigset_t sigignored, sigcaught;
647:     struct proc *p;
648:     struct sigcontext *sc;
649:     struct vma *vma;
650:     int text, data, stack, mmap;
651:
652:     size = text = data = stack = mmap = 0;
653:     if((p = get_proc_by_pid(pid)) {
654:         if(!p->vma) {
655:             return 0;
656:         }
657:         vma_start = p->vma[0].start;
658:         vma_end = p->vma[0].end;
659:
660:         vma = p->vma;
661:         for(n = 0; n < VMA_REGIONS && vma->start; n++, vma++) {
662:             switch(vma->s_type) {
663:                 case P_TEXT:
664:                     text += vma->end - vma->start;
665:                     break;
666:                 case P_HEAP:
667:                     data += vma->end - vma->start;
668:                     break;
669:                 case P_STACK:
670:                     stack += vma->end - vma->start;
671:                     break;
672:                 case P_MMAP:
673:                     mmap += vma->end - vma->start;
674:                     break;
675:             }
676:         }
677:
678:         sigignored = sigcaught = 0;
679:         for(signum = 0, mask = 1; signum < NSIG; signum++, mask <= 1) {
680:             if(p->sigaction[signum].sa_handler == SIG_IGN) {
681:                 sigignored |= mask;

```

fs/procfs/data.c

Page 12/14

```

682:         }
683:         if(p->sigaction[signal].sa_handler == SIG_DFL) {
684:             sigcaught |= mask;
685:         }
686:     }
687:
688:     esp = eip = 0;
689:     if(p->sp) {
690:         sc = (struct sigcontext *)p->sp;
691:         esp = sc->oldesp;
692:         eip = sc->eip;
693:     }
694:     size = sprintf(buffer, "%d (%s) %c %d %d %d %d %d %d %d %d %d
%u %u %u %u %d %d %d %d %d %u %u %u %u %u %u %u %d %d %u %u %u\n",
695:         p->pid,
696:         p->argv0,
697:         pstate[p->state][0],
698:         p->ppid, p->pgid, p->sid,
699:         p->ctty ? p->ctty->dev : 0,
700:         p->ctty ? p->ctty->pgid : - 1,
701:         0, /* flags */
702:         0, 0, 0, 0, /* minflt, cminflt, majflt, cmajflt */
703:         tv2ticks(&p->usage.ru_utime),
704:         tv2ticks(&p->usage.ru_stime),
705:         tv2ticks(&p->cusage.ru_utime),
706:         tv2ticks(&p->cusage.ru_stime),
707:         0, /* counter */
708:         0, /* priority */
709:         0, /* timeout */
710:         0, /* itrealvalue */
711:         p->start_time,
712:         text + data + stack + mmap,
713:         p->rss,
714:         0x7FFFFFFF, /* rlim */
715:         vma_start, /* startcode */
716:         vma_end, /* endcode */
717:         KERNEL_BASE_ADDR - 1, /* startstack */
718:         esp, /* kstkesp */
719:         eip, /* kstkeip */
720:         p->sigpending,
721:         p->sigblocked,
722:         sigignored,
723:         sigcaught,
724:         p->sleep_address
725:     );
726: }
727: return size;
728: }
729:
730: int data_proc_pid_statm(char *buffer, __pid_t pid)
731: {
732:     int n, size;
733:     struct proc *p;
734:     struct vma *vma;
735:     int text, data, stack, mmap;
736:
737:     size = text = data = stack = mmap = 0;
738:     if((p = get_proc_by_pid(pid))) {
739:         if(!p->vma) {
740:             return 0;
741:         }
742:         vma = p->vma;
743:         for(n = 0; n < VMA_REGIONS && vma->start; n++, vma++) {
744:             switch(vma->s_type) {
745:                 case P_TEXT:
746:                     text += vma->end - vma->start;
747:                     break;

```

fs/procfs/data.c

Page 13/14

```

748:         case P_HEAP:
749:             data += vma->end - vma->start;
750:             break;
751:         case P_STACK:
752:             stack += vma->end - vma->start;
753:             break;
754:         case P_MMAP:
755:             mmap += vma->end - vma->start;
756:             break;
757:     }
758: }
759:
760: size = sprintf(buffer, "%d", (text + data + stack + mmap) / PAGE
_SIZE);
761: size += sprintf(buffer + size, " %d", p->rss);
762: size += sprintf(buffer + size, " 0"); /* shared mappings */
763: size += sprintf(buffer + size, " %d", text / PAGE_SIZE);
764: size += sprintf(buffer + size, " 0");
765: size += sprintf(buffer + size, " %d", (data + stack) / PAGE_SIZE
);
766: size += sprintf(buffer + size, " 0\n");
767: }
768: return size;
769: }
770:
771: int data_proc_pid_status(char *buffer, __pid_t pid)
772: {
773:     int n, size;
774:     int signum, mask;
775:     __sigset_t sigignored, sigcaught;
776:     struct proc *p;
777:     struct vma *vma;
778:     int text, data, stack, mmap;
779:
780:     size = text = data = stack = mmap = 0;
781:     if((p = get_proc_by_pid(pid)) {
782:         if(!p->vma) {
783:             return 0;
784:         }
785:         vma = p->vma;
786:         for(n = 0; n < VMA_REGIONS && vma->start; n++, vma++) {
787:             switch(vma->s_type) {
788:                 case P_TEXT:
789:                     text += vma->end - vma->start;
790:                     break;
791:                 case P_HEAP:
792:                     data += vma->end - vma->start;
793:                     break;
794:                 case P_STACK:
795:                     stack += vma->end - vma->start;
796:                     break;
797:                 case P_MMAP:
798:                     mmap += vma->end - vma->start;
799:                     break;
800:             }
801:         }
802:
803:         size = sprintf(buffer, "Name:\t%s\n", p->argv0);
804:         size += sprintf(buffer + size, "State:\t%s\n", pstate[p->state])
;
805:         size += sprintf(buffer + size, "Pid:\t%d\n", p->pid);
806:         size += sprintf(buffer + size, "PPid:\t%d\n", p->ppid);
807:         size += sprintf(buffer + size, "Uid:\t%d\t%d\t%d\t-\n", p->uid,
p->euid, p->suid);
808:         size += sprintf(buffer + size, "Gid:\t%d\t%d\t%d\t-\n", p->gid,
p->egid, p->sgid);
809:         size += sprintf(buffer + size, "VmSize:\t%8d kB\n", (text + data

```

```
+ stack + mmap) / 1024);
810:         size += sprintf(buffer + size, "VmLck:\t%8d kB\n", 0);
811:         size += sprintf(buffer + size, "VmRSS:\t%8d kB\n", p->rss << 2);
812:         size += sprintf(buffer + size, "VmData:\t%8d kB\n", data / 1024)
;
813:         size += sprintf(buffer + size, "VmStk:\t%8d kB\n", stack / 1024)
;
814:         size += sprintf(buffer + size, "VmExe:\t%8d kB\n", text / 1024);
815:         size += sprintf(buffer + size, "VmLib:\t%8d kB\n", 0);
816:         size += sprintf(buffer + size, "SigPnd:\t%08x\n", p->sigpending)
;
817:         size += sprintf(buffer + size, "SigBlk:\t%08x\n", p->sigblocked)
;
818:         sigignored = sigcaught = 0;
819:         for(signum = 0, mask = 1; signum < NSIG; signum++, mask <=<= 1) {
820:             if(p->sigaction[signum].sa_handler == SIG_IGN) {
821:                 sigignored |= mask;
822:             }
823:             if(p->sigaction[signum].sa_handler == SIG_DFL) {
824:                 sigcaught |= mask;
825:             }
826:         }
827:         size += sprintf(buffer + size, "SigIgn:\t%08x\n", sigignored);
828:         size += sprintf(buffer + size, "SigCgt:\t%08x\n", sigcaught);
829:     }
830:     return size;
831: }
```


fs/procfs/dir.c

Page 1/4

```

1: /*
2:  * fiwix/fs/procfs/dir.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/errno.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/filesystems.h>
12: #include <fiwix/fs_proc.h>
13: #include <fiwix/dirent.h>
14: #include <fiwix/stat.h>
15: #include <fiwix/mm.h>
16: #include <fiwix/stdio.h>
17: #include <fiwix/string.h>
18:
19: struct fs_operations procfs_dir_fsop = {
20:     0,
21:     0,
22:
23:     procfs_dir_open,
24:     procfs_dir_close,
25:     procfs_dir_read,
26:     NULL, /* write */
27:     NULL, /* ioctl */
28:     NULL, /* lseek */
29:     procfs_dir_readdir,
30:     NULL, /* mmap */
31:     NULL, /* select */
32:
33:     NULL, /* readlink */
34:     NULL, /* followlink */
35:     procfs_bmap,
36:     procfs_lookup,
37:     NULL, /* rmdir */
38:     NULL, /* link */
39:     NULL, /* unlink */
40:     NULL, /* symlink */
41:     NULL, /* mkdir */
42:     NULL, /* mknod */
43:     NULL, /* truncate */
44:     NULL, /* create */
45:     NULL, /* rename */
46:
47:     NULL, /* read_block */
48:     NULL, /* write_block */
49:
50:     NULL, /* read_inode */
51:     NULL, /* write_inode */
52:     NULL, /* ialloc */
53:     NULL, /* ifree */
54:     NULL, /* statfs */
55:     NULL, /* read_superblock */
56:     NULL, /* remount_fs */
57:     NULL, /* write_superblock */
58:     NULL, /* release_superblock */
59: };
60:
61: static int proc_listdir(char *buffer, int count)
62: {
63:     int n;
64:     struct proc *p;
65:     struct procfs_dir_entry *pd;
66:     struct procfs_dir_entry d;
67:     int size;

```

fs/procfs/dir.c

Page 2/4

```

68:
69:     size = 0;
70:     pd = (struct procfs_dir_entry *)buffer;
71:
72:     FOR_EACH_PROCESS(p) {
73:         d.inode = PROC_PID_INO + (p->pid << 12);
74:         d.mode = S_IFDIR | S_IRUSR | S_IXUSR | S_IRGRP | S_IXGRP | S_IRO
TH | S_IXOTH;
75:         d.nlink = 1;
76:         d.lev = -1;
77:         d.name_len = 1;
78:         n = p->pid;
79:         while(n / 10) {
80:             n /= 10;
81:             d.name_len++;
82:         }
83:         d.name = p->pidstr;
84:         d.data_fn = NULL;
85:
86:         if(size + sizeof(struct procfs_dir_entry) > (count - 1)) {
87:             printk("WARNING: kmalloc() is limited to 4096 bytes.\n")
;
88:             break;
89:         }
90:
91:         size += sizeof(struct procfs_dir_entry);
92:         memcpy_b((void *)pd, (void *)&d, sizeof(struct procfs_dir_entry)
);
93:         pd++;
94:         p = p->next;
95:     }
96:     return size;
97: }
98:
99: /*
100: static int proc_listfd(struct inode *i, char *buffer, int count)
101: {
102:     int n;
103:     struct proc *p;
104:     struct procfs_dir_entry *pd;
105:     struct procfs_dir_entry d;
106:     int size;
107:
108:     size = 0;
109:     pd = (struct procfs_dir_entry *)buffer;
110:
111:     p = get_proc_by_pid((i->inode >> 12) & 0xFFFF);
112:     for(n = 0; n < OPEN_MAX; n++) {
113:         if(p->fd[n]) {
114:             d.inode = PROC_PID_INO + (p->pid << 12) + n;
115:             d.mode = S_IFREG | S_IRWXU;
116:             d.nlink = 1;
117:             d.lev = -1;
118:             d.name_len = sprintk(d.name, "%d", n);
119:             d.data_fn = NULL;
120:
121:             if(size + sizeof(struct procfs_dir_entry) > (count - 1))
{
122:                 printk("WARNING: kmalloc() is limited to 4096 by
tes.\n");
123:                 break;
124:             }
125:
126:             size += sizeof(struct procfs_dir_entry);
127:             memcpy_b((void *)pd, (void *)&d, sizeof(struct procfs_di
r_entry));
128:             pd++;

```

fs/procfs/dir.c

Page 3/4

```

129:         }
130:     }
131:     memset_b((void *)pd + size, NULL, sizeof(struct procfs_dir_entry));
132:     return size;
133: }
134: */
135:
136: static int dir_read(struct inode *i, struct fd *fd_table, char *buffer, __size_t
count)
137: {
138:     __off_t total_read;
139:     unsigned int bytes;
140:     int len, lev;
141:     char *buf;
142:
143:     if(!(buf = (void *)kmalloc())) {
144:         return -ENOMEM;
145:     }
146:
147:     /* create the list of directories for each process */
148:     len = 0;
149:     if(i->inode == PROC_ROOT_INO) {
150:         len = proc_listdir(buf, count);
151:     }
152:
153:     /* TODO: create the list of fds used for each process
154:     if((i->inode & 0xF0000FFF) == PROC_PID_FD) {
155:         len = proc_listfd(i, buf, count);
156:     }
157:     */
158:
159:     /* add the rest of static files in the main directory */
160:     lev = i->u.procfs.i_lev;
161:
162:     /* assigns the size of the level without the last entry (NULL) */
163:     bytes = sizeof(procfs_array[lev]) - sizeof(struct procfs_dir_entry);
164:
165:     if((len + bytes) > (count - 1)) {
166:         printk("WARNING: %s(): len (%d) > count (%d).\n", __FUNCTION__,
len, count);
167:         kfree((unsigned int)buf);
168:         return 0;
169:     }
170:     memcpy_b(buf + len, (char *)&procfs_array[lev], bytes);
171:     len += bytes;
172:     total_read = fd_table->offset = len;
173:     memcpy_b(buffer, buf, len);
174:     kfree((unsigned int)buf);
175:     return total_read;
176: }
177:
178: int procfs_dir_open(struct inode *i, struct fd *fd_table)
179: {
180:     fd_table->offset = 0;
181:     return 0;
182: }
183:
184: int procfs_dir_close(struct inode *i, struct fd *fd_table)
185: {
186:     return 0;
187: }
188:
189: int procfs_dir_read(struct inode *i, struct fd *fd_table, char *buffer, __size_t
count)
190: {
191:     return -EISDIR;
192: }

```

fs/procfs/dir.c

Page 4/4

```

193:
194: int procfs_dir_readdir(struct inode *i, struct fd *fd_table, struct dirent *dire
nt, unsigned int count)
195: {
196:     unsigned int offset, boffset, dirent_offset, doffset;
197:     int dirent_len;
198:     unsigned int total_read;
199:     struct procfs_dir_entry *d;
200:     int base_dirent_len;
201:     char *buffer;
202:     int lev;
203:
204:     if(!(buffer = (void *)kmalloc())) {
205:         return -ENOMEM;
206:     }
207:
208:     lev = i->u.procfs.i_lev;
209:     base_dirent_len = sizeof(dirent->d_ino) + sizeof(dirent->d_off) + sizeof
(dirent->d_reclen);
210:
211:     offset = fd_table->offset;
212:     boffset = dirent_offset = doffset = 0;
213:
214:     boffset = offset % PAGE_SIZE;
215:
216:     total_read = dir_read(i, fd_table, buffer, PAGE_SIZE);
217:     if((count = MIN(total_read, count)) == 0) {
218:         kfree((unsigned int)buffer);
219:         return dirent_offset;
220:     }
221:
222:     while(boffset < fd_table->offset) {
223:         d = (struct procfs_dir_entry *) (buffer + boffset);
224:         if(!d->inode) {
225:             break;
226:         }
227:         dirent_len = (base_dirent_len + (d->name_len + 1)) + 3;
228:         dirent_len &= ~3; /* round up */
229:         if((doffset + dirent_len) <= count) {
230:             boffset += sizeof(struct procfs_dir_entry);
231:             offset += sizeof(struct procfs_dir_entry);
232:             doffset += dirent_len;
233:             dirent->d_ino = d->inode;
234:             dirent->d_off = offset;
235:             dirent->d_reclen = dirent_len;
236:             memcpy_b(dirent->d_name, d->name, d->name_len);
237:             dirent->d_name[d->name_len] = NULL;
238:             dirent = (struct dirent *) ((char *)dirent + dirent_len);
239:             dirent_offset += dirent_len;
240:         } else {
241:             break;
242:         }
243:     }
244:     fd_table->offset = boffset;
245:     kfree((unsigned int)buffer);
246:     return dirent_offset;
247: }

```

fs/procfs/file.c

Page 1/2

```

1: /*
2:  * fiwix/fs/procfs/file.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/errno.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/filesystems.h>
12: #include <fiwix/fs_proc.h>
13: #include <fiwix/fcntl.h>
14: #include <fiwix/mm.h>
15: #include <fiwix/stdio.h>
16: #include <fiwix/string.h>
17:
18: struct fs_operations procfs_file_fsop = {
19:     0,
20:     0,
21:
22:     procfs_file_open,
23:     procfs_file_close,
24:     procfs_file_read,
25:     NULL, /* write */
26:     NULL, /* ioctl */
27:     procfs_file_lseek,
28:     NULL, /* readdir */
29:     NULL, /* mmap */
30:     NULL, /* select */
31:
32:     NULL, /* readlink */
33:     NULL, /* followlink */
34:     procfs_bmap,
35:     NULL, /* lookup */
36:     NULL, /* rmdir */
37:     NULL, /* link */
38:     NULL, /* unlink */
39:     NULL, /* symlink */
40:     NULL, /* mkdir */
41:     NULL, /* mknod */
42:     NULL, /* truncate */
43:     NULL, /* create */
44:     NULL, /* rename */
45:
46:     NULL, /* read_block */
47:     NULL, /* write_block */
48:
49:     NULL, /* read_inode */
50:     NULL, /* write_inode */
51:     NULL, /* ialloc */
52:     NULL, /* ifree */
53:     NULL, /* statfs */
54:     NULL, /* read_superblock */
55:     NULL, /* remount_fs */
56:     NULL, /* write_superblock */
57:     NULL, /* release_superblock */
58: };
59:
60: int procfs_file_open(struct inode *i, struct fd *fd_table)
61: {
62:     if(fd_table->flags & (O_WRONLY | O_RDWR | O_TRUNC | O_APPEND)) {
63:         return -EINVAL;
64:     }
65:     fd_table->offset = 0;
66:     return 0;
67: }

```

fs/procfs/file.c

Page 2/2

```

68:
69: int procfs_file_close(struct inode *i, struct fd *fd_table)
70: {
71:     return 0;
72: }
73:
74: int procfs_file_read(struct inode *i, struct fd *fd_table, char *buffer, __size_
t count)
75: {
76:     __off_t total_read;
77:     unsigned int boffset, bytes, size;
78:     int blksize;
79:     struct procfs_dir_entry *d;
80:     char *buf;
81:
82:     if(!(d = get_procfs_by_inode(i))) {
83:         return -EINVAL;
84:     }
85:     if(!d->data_fn) {
86:         return -EINVAL;
87:     }
88:     if(!(buf = (void *)kmalloc())) {
89:         return -ENOMEM;
90:     }
91:
92:     size = d->data_fn(buf, (i->inode >> 12) & 0xFFFF);
93:     blksize = i->sb->s_blocksize;
94:     if(fd_table->offset > size) {
95:         fd_table->offset = size;
96:     }
97:
98:     total_read = 0;
99:
100:    for(;;) {
101:        count = (fd_table->offset + count > size) ? size - fd_table->off
set : count;
102:        if(!count) {
103:            break;
104:        }
105:
106:        boffset = fd_table->offset % blksize;
107:        bytes = blksize - boffset;
108:        bytes = MIN(bytes, count);
109:        memcpy_b(buffer + total_read, buf + boffset, bytes);
110:        total_read += bytes;
111:        count -= bytes;
112:        fd_table->offset += bytes;
113:    }
114:
115:    kfree((unsigned int)buf);
116:    return total_read;
117: }
118:
119: int procfs_file_lseek(struct inode *i, __off_t offset)
120: {
121:     return offset;
122: }

```

fs/procfs/inode.c

Page 1/2

```

1: /*
2:  * fiwix/fs/procfs/inode.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/filesystems.h>
11: #include <fiwix/fs_proc.h>
12: #include <fiwix/statfs.h>
13: #include <fiwix/sleep.h>
14: #include <fiwix/stat.h>
15: #include <fiwix/sched.h>
16: #include <fiwix/mm.h>
17: #include <fiwix/process.h>
18: #include <fiwix/errno.h>
19: #include <fiwix/stdio.h>
20: #include <fiwix/string.h>
21:
22: int procfs_read_inode(struct inode *i)
23: {
24:     int lev;
25:     __mode_t mode;
26:     __nlink_t nlink;
27:     struct procfs_dir_entry *d;
28:
29:     if((i->inode & 0xF0000FFF) == PROC_PID_INO) { /* dynamic PID dir */
30:         mode = S_IFDIR | S_IRUSR | S_IXUSR | S_IRGRP | S_IXGRP | S_IROTH
| S_IXOTH;
31:         nlink = 3;
32:         lev = PROC_PID_LEV;
33:     } else {
34:         if(!(d = get_procfs_by_inode(i))) {
35:             return -ENOENT;
36:         }
37:         mode = d->mode;
38:         nlink = d->nlink;
39:         lev = d->lev;
40:     }
41:
42:     i->i_mode = mode;
43:     i->i_uid = 0;
44:     i->i_size = 0;
45:     i->i_atime = CURRENT_TIME;
46:     i->i_ctime = CURRENT_TIME;
47:     i->i_mtime = CURRENT_TIME;
48:     i->i_gid = 0;
49:     i->i_nlink = nlink;
50:     i->i_blocks = 0;
51:     i->i_flags = 0;
52:     i->locked = 1;
53:     i->dirty = 0;
54:     i->count = 1;
55:     i->u.procfs.i_lev = lev;
56:     switch(i->i_mode & S_IFMT) {
57:         case S_IFDIR:
58:             i->fsop = &procfs_dir_fsop;
59:             break;
60:         case S_IFREG:
61:             i->fsop = &procfs_file_fsop;
62:             break;
63:         case S_IFLNK:
64:             i->fsop = &procfs_symlink_fsop;
65:             break;
66:         default:

```

fs/procfs/inode.c

Page 2/2

```
67:                                PANIC("invalid inode (%d) mode %08o.\n", i->inode, i->i_
mode);
68:                                }
69:                                return 0;
70: }
71:
72: int procfs_bmap(struct inode *i, __off_t offset, int mode)
73: {
74:     return i->u.procfs.i_lev;
75: }
76:
77: void procfs_statfs(struct superblock *sb, struct statfs *statfsbuf)
78: {
79:     statfsbuf->f_type = PROC_SUPER_MAGIC;
80:     statfsbuf->f_bsize = sb->s_blocksize;
81:     statfsbuf->f_blocks = 0;
82:     statfsbuf->f_bfree = 0;
83:     statfsbuf->f_bavail = 0;
84:     statfsbuf->f_files = 0;
85:     statfsbuf->f_ffree = 0;
86:     /* statfsbuf->f_fsid = ? */
87:     statfsbuf->f_namelen = NAME_MAX;
88: }
```


fs/procfs/Makefile

Page 1/1

```
1: # fiwix/fs/procfs/Makefile
2: #
3: # Copyright 2018, Jordi Sanfeliu. All rights reserved.
4: # Distributed under the terms of the Fiwix License.
5: #
6:
7: .S.o:
8:     $(CC) -traditional -I$(INCLUDE) -c -o $@ $<
9: .c.o:
10:     $(CC) $(CFLAGS) -c -o $@ $<
11:
12: OBJS = super.o inode.o namei.o dir.o file.o symlink.o tree.o data.o
13:
14: procfs: $(OBJS)
15:     $(LD) $(LDFLAGS) -r $(OBJS) -o procfs.o
16:
17: clean:
18:     rm -f *.o
19:
```

fs/procfs/namei.c

Page 1/2

```

1: /*
2:  * fiwix/fs/procfs/namei.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/fs.h>
10: #include <fiwix/filesystems.h>
11: #include <fiwix/fs_proc.h>
12: #include <fiwix/process.h>
13: #include <fiwix/stat.h>
14: #include <fiwix/mm.h>
15: #include <fiwix/errno.h>
16: #include <fiwix/stdio.h>
17: #include <fiwix/string.h>
18:
19: int procfs_lookup(const char *name, struct inode *dir, struct inode **i_res)
20: {
21:     int len, lev;
22:     __ino_t inode;
23:     __pid_t pid;
24:     struct proc *p;
25:     struct procfs_dir_entry *pdirent;
26:
27:     pid = inode = 0;
28:     len = strlen(name);
29:     if((dir->inode & 0xF0000000) == PROC_PID_INO) {
30:         pid = (dir->inode >> 12) & 0xFFFF;
31:     }
32:
33:     lev = bmap(dir, 0, FOR_READING);
34:     pdirent = procfs_array[lev];
35:     while(pdirent->inode && !inode) {
36:         if(len == pdirent->name_len) {
37:             if(!(strcmp(pdirent->name, name))) {
38:                 inode = pdirent->inode;
39:                 if(pid) {
40:                     inode = (PROC_PID_INO + (pid << 12)) + (
inode & 0xFFF);
41:                     if(strcmp(".", name) == 0) {
42:                         inode = dir->inode;
43:                     }
44:                     if(strcmp("../", name) == 0) {
45:                         inode = pdirent->inode;
46:                     }
47:                 }
48:             }
49:         }
50:         if(inode) {
51:             /*
52:              * This prevents a deadlock in iget() when
53:              * trying to lock '.' when 'dir' is the same
54:              * directory (ls -lai <dir>).
55:              */
56:             if(inode == dir->inode) {
57:                 *i_res = dir;
58:                 return 0;
59:             }
60:
61:             if(!(*i_res = iget(dir->sb, inode))) {
62:                 return -EACCES;
63:             }
64:             iput(dir);
65:             return 0;
66:         }

```

fs/procfs/namei.c

Page 2/2

```
67:         pdirent++;
68:     }
69:
70:     FOR_EACH_PROCESS(p) {
71:         if(len == strlen(p->pidstr)) {
72:             if(!(strcmp(p->pidstr, name))) {
73:                 inode = PROC_PID_INO + (p->pid << 12);
74:             }
75:         }
76:         if(inode) {
77:             if(!(*i_res = iget(dir->sb, inode))) {
78:                 return -EACCES;
79:             }
80:             iput(dir);
81:             return 0;
82:         }
83:         p = p->next;
84:     }
85:     iput(dir);
86:     return -ENOENT;
87: }
```

fs/procfs/super.c

Page 1/2

```

1: /*
2:  * fiwix/fs/procfs/super.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/errno.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/filesystems.h>
12: #include <fiwix/fs_proc.h>
13: #include <fiwix/stat.h>
14: #include <fiwix/mm.h>
15: #include <fiwix/sched.h>
16: #include <fiwix/stdio.h>
17: #include <fiwix/string.h>
18:
19: struct fs_operations procfs_fsop = {
20:     0,
21:     PROC_DEV,
22:
23:     NULL, /* open */
24:     NULL, /* close */
25:     NULL, /* read */
26:     NULL, /* write */
27:     NULL, /* ioctl */
28:     NULL, /* lseek */
29:     NULL, /* readdir */
30:     NULL, /* mmap */
31:     NULL, /* select */
32:
33:     NULL, /* readlink */
34:     NULL, /* followlink */
35:     NULL, /* bmap */
36:     NULL, /* lookup */
37:     NULL, /* rmdir */
38:     NULL, /* link */
39:     NULL, /* unlink */
40:     NULL, /* symlink */
41:     NULL, /* mkdir */
42:     NULL, /* mknod */
43:     NULL, /* truncate */
44:     NULL, /* create */
45:     NULL, /* rename */
46:
47:     NULL, /* read_block */
48:     NULL, /* write_block */
49:
50:     procfs_read_inode,
51:     NULL, /* write_inode */
52:     NULL, /* ialloc */
53:     NULL, /* ifree */
54:     procfs_statfs,
55:     procfs_read_superblock,
56:     NULL, /* remount_fs */
57:     NULL, /* write_superblock */
58:     NULL, /* release_superblock */
59: };
60:
61: int procfs_read_superblock(__dev_t dev, struct superblock *sb)
62: {
63:     superblock_lock(sb);
64:     sb->dev = dev;
65:     sb->fsop = &procfs_fsop;
66:     sb->s_blocksize = PAGE_SIZE;
67:

```

fs/procfs/super.c

Page 2/2

```
68:         if(!(sb->root = iget(sb, PROC_ROOT_INO))) {
69:             printk("WARNING: %s(): unable to get root inode.\n", __FUNCTION_
_);
70:             superblock_unlock(sb);
71:             return -EINVAL;
72:         }
73:         sb->root->u.procfs.i_lev = 0;
74:
75:         superblock_unlock(sb);
76:         return 0;
77:     }
78:
79: int procfs_init(void)
80: {
81:     return register_filesystem("proc", &procfs_fsop);
82: }
```

fs/procfs/symlink.c

Page 1/3

```

1:  /*
2:  *  fiwix/fs/procfs/symlink.c
3:  *
4:  *  Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  *  Distributed under the terms of the Fiwix License.
6:  */
7:
8:  #include <fiwix/types.h>
9:  #include <fiwix/errno.h>
10: #include <fiwix/buffer.h>
11: #include <fiwix/fs.h>
12: #include <fiwix/filesystems.h>
13: #include <fiwix/fs_proc.h>
14: #include <fiwix/mm.h>
15: #include <fiwix/stat.h>
16: #include <fiwix/stdio.h>
17: #include <fiwix/string.h>
18:
19: struct fs_operations procfs_symlink_fsop = {
20:     0,
21:     0,
22:
23:     NULL,          /* open */
24:     NULL,          /* close */
25:     NULL,          /* read */
26:     NULL,          /* write */
27:     NULL,          /* ioctl */
28:     NULL,          /* lseek */
29:     NULL,          /* readdir */
30:     NULL,          /* mmap */
31:     NULL,          /* select */
32:
33:     procfs_readlink,
34:     procfs_followlink,
35:     NULL,          /* bmap */
36:     NULL,          /* lookup */
37:     NULL,          /* rmdir */
38:     NULL,          /* link */
39:     NULL,          /* unlink */
40:     NULL,          /* symlink */
41:     NULL,          /* mkdir */
42:     NULL,          /* mknod */
43:     NULL,          /* truncate */
44:     NULL,          /* create */
45:     NULL,          /* rename */
46:
47:     NULL,          /* read_block */
48:     NULL,          /* write_block */
49:
50:     NULL,          /* read_inode */
51:     NULL,          /* write_inode */
52:     NULL,          /* ialloc */
53:     NULL,          /* ifree */
54:     NULL,          /* statfs */
55:     NULL,          /* read_superblock */
56:     NULL,          /* remount_fs */
57:     NULL,          /* write_superblock */
58:     NULL,          /* release_superblock */
59: };
60:
61: int procfs_readlink(struct inode *i, char *buffer, __size_t count)
62: {
63:     int size_read;
64:     struct procfs_dir_entry *d;
65:     char *buf;
66:
67:     if (!(d = get_procfs_by_inode(i))) {

```

fs/procfs/symlink.c

Page 2/3

```

68:         return -EINVAL;
69:     }
70:
71:     if(!d->data_fn) {
72:         return -EINVAL;
73:     }
74:
75:     if(!(buf = (char *)kmalloc())) {
76:         return -ENOMEM;
77:     }
78:
79:     if((size_read = d->data_fn(buf, (i->inode >> 12) & 0xFFFF)) > 0) {
80:         if(size_read > count) {
81:             size_read = count;
82:         }
83:         memcpy_b(buffer, buf, size_read);
84:     }
85:     kfree((unsigned int)buf);
86:     return size_read;
87: }
88:
89: int procfs_followlink(struct inode *dir, struct inode *i, struct inode **i_res)
90: {
91:     __ino_t errno;
92:     __pid_t pid;
93:     struct proc *p;
94:
95:     if(!i) {
96:         return -ENOENT;
97:     }
98:     if(!(S_ISLNK(i->i_mode))) {
99:         printk("%s(): Oops, inode '%d' is not a symlink (!?)\n", __FUNC
TION_, i->inode);
100:         return 0;
101:     }
102:
103:     p = NULL;
104:     if((pid = (i->inode >> 12) & 0xFFFF)) {
105:         if(!(p = get_proc_by_pid(pid))) {
106:             return -ENOENT;
107:         }
108:     }
109:
110:     /* FIXME!
111:     if(p && p->root) {
112:         printk("(pid %d) p->root->inode = %d (count = %d)\n", p->pid, p-
>root->inode, p->root->count);
113:     }
114:     */
115:
116:     switch(i->inode & 0xF0000FFF) {
117:         case PROC_PID_CWD:
118:             if(!p->pwd) {
119:                 return -ENOENT;
120:             }
121:             *i_res = p->pwd;
122:             p->pwd->count++;
123:             iput(i);
124:             break;
125:         case PROC_PID_EXE:
126:             if(!p->vma || !p->vma->inode) {
127:                 return -ENOENT;
128:             }
129:             *i_res = p->vma->inode;
130:             p->vma->inode->count++;
131:             iput(i);
132:             break;

```

fs/procfs/symlink.c

Page 3/3

```
133:         case PROC_PID_ROOT:
134:             if(!p->root) {
135:                 return -ENOENT;
136:             }
137:             *i_res = p->root;
138:             p->root->count++;
139:             iput(i);
140:             break;
141:         default:
142:             iput(i);
143:             if((errno = parse_namei(current->pidstr, dir, i_res, NUL
L, FOLLOW_LINKS))) {
144:                 return errno;
145:             }
146:         }
147:     return 0;
148: }
```


fs/procfs/tree.c

Page 1/2

```

1: /*
2:  * fiwix/fs/procfs/tree.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/types.h>
9: #include <fiwix/stat.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/fs_proc.h>
12: #include <fiwix/errno.h>
13: #include <fiwix/stdio.h>
14: #include <fiwix/string.h>
15:
16: #define DIR      S_IFDIR | S_IRUSR | S_IXUSR | S_IRGRP | S_IXGRP | S_IROTH | \
17:                S_IXOTH /* dr-xr-xr-x */
18: #define REG      S_IFREG | S_IRUSR | S_IRGRP | S_IROTH /* -r--r--r-- */
19: #define REGUSR   S_IFREG | S_IRUSR /* -r----- */
20: #define LNK      S_IFLNK | S_IRWXU | S_IRWXG | S_IRWXO /* lrwxrwxrwx */
21: #define LNKPID   S_IFLNK | S_IRWXU /* lrwx----- */
22:
23: /*
24:  * WARNING: every time a new entry is added to this array you must also change
25:  * the PROC_ARRAY_ENTRIES value defined in fs_proc.h.
26:  */
27: struct procfs_dir_entry procfs_array[PROC_ARRAY_ENTRIES + 1] = {
28:     { /* [0] / */
29:         { 1, DIR, 2, 0, 1, ".", NULL },
30:         { 2, DIR, 2, 0, 2, "..", NULL },
31:         { 3, DIR, 3, 3, 3, "sys", NULL },
32:         { 4, REG, 1, 0, 7, "cmdline", data_proc_cmdline },
33:         { 5, REG, 1, 0, 7, "cpuinfo", data_proc_cpuinfo },
34:         { 6, REG, 1, 0, 7, "devices", data_proc_devices },
35:         { 7, REG, 1, 0, 3, "dma", data_proc_dma },
36:         { 8, REG, 1, 0, 11, "filesystems", data_proc_filesystems },
37:         { 9, REG, 1, 0, 10, "interrupts", data_proc_interrupts },
38:         { 10, REG, 1, 0, 7, "loadavg", data_proc_loadavg },
39:         { 11, REG, 1, 0, 5, "locks", data_proc_locks },
40:         { 12, REG, 1, 0, 7, "meminfo", data_proc_meminfo },
41:         { 13, REG, 1, 0, 6, "mounts", data_proc_mounts },
42:         { 14, REG, 1, 0, 10, "partitions", data_proc_partitions },
43:         { 15, REG, 1, 0, 3, "rtc", data_proc_rtc },
44:         { 16, LNK, 1, 0, 4, "self", data_proc_self },
45:         { 17, REG, 1, 0, 4, "stat", data_proc_stat },
46:         { 18, REG, 1, 0, 6, "uptime", data_proc_uptime },
47:         { 19, REG, 1, 0, 7, "version", data_proc_fullversion },
48:         { 0, 0, 0, 0, 0, NULL, NULL }
49:     },
50:     { /* [1] /PID/ */
51:         { 1000, DIR, 2, 1, 1, ".", NULL },
52:         { 1, DIR, 2, 0, 2, "..", NULL },
53:         /* { PROC_PID_FD, DIR, 2, 2, 2, "fd", data_proc_pid_fd }, */
54:         { PROC_PID_CMDLINE, REG, 1, 1, 7, "cmdline", data_proc_pid_cmdline
55:     },
56:     { PROC_PID_CWD, LNKPID, 1, 1, 3, "cwd", data_proc_pid_cwd },
57:     { PROC_PID_ENVIRON, REGUSR, 1, 1, 7, "environ", data_proc_pid_environ
58: },
59:     { PROC_PID_EXE, LNKPID, 1, 1, 3, "exe", data_proc_pid_exe },
60:     { PROC_PID_MAPS, REG, 1, 1, 4, "maps", data_proc_pid_maps },
61:     { PROC_PID_MOUNTINFO, REG, 1, 1, 9, "mountinfo", data_proc_pid_mountinf
62: },
63:     { PROC_PID_ROOT, LNKPID, 1, 1, 4, "root", data_proc_pid_root },
64:     { PROC_PID_STAT, REG, 1, 1, 4, "stat", data_proc_pid_stat },
65:     { PROC_PID_STATM, REG, 1, 1, 5, "statm", data_proc_pid_statm },
66:     { PROC_PID_STATUS, REG, 1, 1, 6, "status", data_proc_pid_status }
67: },

```

fs/procfs/tree.c

Page 2/2

```

64:     { 0, 0, 0, 0, 0, NULL, NULL }
65: },
66:
67: {
68: },
69:
70: { /* [3] /sys/ */
71:     { 3,     DIR,  2, 3, 1,  ".",      NULL },
72:     { 1,     DIR,  2, 0, 2,  "..",     NULL },
73:     { 2001,  DIR,  2, 4, 6,  "kernel", NULL },
74:     { 0, 0, 0, 0, 0, NULL, NULL }
75: },
76: { /* [4] /sys/kernel/ */
77:     { 2001,  DIR,  2, 4, 1,  ".",      NULL },
78:     { 3,     DIR,  2, 3, 2,  "..",     NULL },
79:     { 3001,  REG,  1, 4, 10, "domainname", data_proc_domainname },
80:     { 3002,  REG,  1, 4, 8,  "file-max",  data_proc_filemax },
81:     { 3003,  REG,  1, 4, 7,  "file-nr",   data_proc_filenr },
82:     { 3004,  REG,  1, 4, 8,  "hostname",  data_proc_hostname },
83:     { 3005,  REG,  1, 4, 9,  "inode-max", data_proc_inodemax },
84:     { 3006,  REG,  1, 4, 8,  "inode-nr",  data_proc_inodenr },
85:     { 3007,  REG,  1, 4, 9,  "osrelease", data_proc_osrelease },
86:     { 3008,  REG,  1, 4, 6,  "ostype",    data_proc_ostype },
87:     { 3009,  REG,  1, 4, 7,  "version",   data_proc_version },
88:     { 0, 0, 0, 0, 0, NULL, NULL }
89: }
90: };
91:
92: struct procfs_dir_entry * get_procfs_by_inode(struct inode *i)
93: {
94:     __ino_t inode;
95:     int n, lev;
96:     struct procfs_dir_entry *d;
97:
98:     inode = i->inode;
99:     for(lev = 0; procfs_array[lev]; lev++) {
100:         if(lev == PROC_PID_LEV) { /* PID entries */
101:             if((i->inode & 0xF0000000) == PROC_PID_INO) {
102:                 inode = i->inode & 0xF0000FFF;
103:             }
104:         }
105:         d = procfs_array[lev];
106:         for(n = 0; n < PROC_ARRAY_ENTRIES && d->inode; n++) {
107:             if(d->inode == inode) {
108:                 return d;
109:             }
110:             d++;
111:         }
112:     }
113:     return NULL;
114: }
115: }

```

mm/alloc.c

Page 1/1

```
1: /*
2:  * fiwix/mm/alloc.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/mm.h>
9: #include <fiwix/stdio.h>
10: #include <fiwix/string.h>
11:
12: /*
13:  * The implementation of kernel memory allocation is extremely simple, it works
14:  * with a granularity of PAGE_SIZE (4096 bytes). There is indeed a lot of room
15:  * for improvements here.
16:  */
17: unsigned int kmalloc(void)
18: {
19:     struct page *pg;
20:     unsigned int addr;
21:
22:     if((pg = get_free_page())) {
23:         addr = pg->page << PAGE_SHIFT;
24:         return P2V(addr);
25:     }
26:
27:     /* out of memory! */
28:     return 0;
29: }
30:
31: void kfree(unsigned int addr)
32: {
33:     addr = V2P(addr);
34:     release_page(addr >> PAGE_SHIFT);
35: }
```

mm/bios_map.c

Page 1/3

```

1: /*
2:  * fiwix/mm/bios_map.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/mm.h>
10: #include <fiwix/bios.h>
11: #include <fiwix/stdio.h>
12: #include <fiwix/string.h>
13:
14: static char *bios_mem_type[] = {
15:     NULL,
16:     "available",
17:     "reserved",
18:     "ACPI Reclaim",
19:     "ACPI NVS",
20:     "unusable",
21:     "disabled"
22: };
23:
24: /* check if an specific address is available in the BIOS memory map */
25: int addr_in_bios_map(unsigned int addr)
26: {
27:     int n, retval;
28:     struct bios_mem_map *bmm;
29:
30:     retval = 0;
31:     bmm = &bios_mem_map[0];
32:
33:     for(n = 0; n < NR_BIOS_MM_ENT; n++, bmm++) {
34:         if(bmm->to && bmm->type == MULTIBOOT_MEMORY_AVAILABLE) {
35:             if(addr >= bmm->from && addr < (bmm->to & PAGE_MASK)) {
36:                 retval = 1;
37:             }
38:         }
39:     }
40:
41:     /* this second pass is necessary because the array is not sorted */
42:     bmm = &bios_mem_map[0];
43:     for(n = 0; n < NR_BIOS_MM_ENT; n++, bmm++) {
44:         if(bmm->to && bmm->type == MULTIBOOT_MEMORY_RESERVED) {
45:             if(addr >= bmm->from && addr < (bmm->to & PAGE_MASK)) {
46:                 retval = 0;
47:             }
48:         }
49:     }
50:
51:     return retval; /* not in BIOS map or not available (reserved, ...) */
52: }
53:
54: void bios_map_add(unsigned long int from, unsigned long int to, int from_type, i
nt to_type)
55: {
56:     int n;
57:
58:     for(n = 0; n < NR_BIOS_MM_ENT; n++) {
59:         if(!bios_mem_map[n].type) {
60:             break;
61:         }
62:     }
63:
64:     if(from_type == to_type) {
65:         printk("memory 0x%08X%08X-0x%08X%08X %s\n",
66:             0, from,

```

mm/bios_map.c

Page 2/3

```

67:             0, to,
68:             bios_mem_type[to_type]
69:         );
70:     } else {
71:         printk("memory    0x%08X%08X-0x%08X%08X %s -> %s\n",
72:             0, from,
73:             0, to,
74:             bios_mem_type[from_type],
75:             bios_mem_type[to_type]
76:         );
77:     }
78:     bios_mem_map[n].from = from;
79:     bios_mem_map[n].to = to;
80:     bios_mem_map[n].type = to_type;
81: }
82:
83: void bios_map_init(struct multiboot_mmap_entry *bmmmap_addr, unsigned long int bm
map_length)
84: {
85:     struct multiboot_mmap_entry *bmmmap;
86:     unsigned int from_high, from_low, to_high, to_low;
87:     unsigned long long to;
88:     int n;
89:
90:     bmmmap = bmmmap_addr;
91:     kstat.physical_pages = 0;
92:
93:     if(bmmmap) {
94:         n = 0;
95:
96:         while((unsigned int)bmmmap < (unsigned int)bmmmap_addr + bmmmap_len
gth) {
97:             from_high = (unsigned int)(bmmmap->addr >> 32);
98:             from_low = (unsigned int)(bmmmap->addr & 0xFFFFFFFF);
99:             to = bmmmap->addr + bmmmap->len;
100:             to_high = (unsigned int)(to >> 32);
101:             to_low = (unsigned int)(to & 0xFFFFFFFF);
102:             printk("%s    0x%08X%08X-0x%08X%08X %s\n",
103:                 n ? "    " : "memory",
104:                 from_high,
105:                 from_low,
106:                 to_high,
107:                 to_low,
108:                 bios_mem_type[(int)bmmmap->type]
109:             );
110:             /* only memory addresses below 4GB are counted */
111:             if(!from_high && !to_high) {
112:                 if(n < NR_BIOS_MM_ENT && bmmmap->len) {
113:                     bios_mem_map[n].from = from_low;
114:                     bios_mem_map[n].to = to_low;
115:                     bios_mem_map[n].type = (int)bmmmap->type;
116:                     if(bios_mem_map[n].type == MULTIBOOT_MEM
ORY_AVAILABLE) {
117:                         from_low = bios_mem_map[n].from
& PAGE_MASK;
118:                         to_low = bios_mem_map[n].to & PA
GE_MASK;
119:
120:                         /* the first MB is not counted h
ere */
121:                         if(from_low >= 0x100000) {
122:                             kstat.physical_pages +=
(to_low - from_low) / PAGE_SIZE;
123:                         }
124:                     }
125:                     n++;
126:                 }

```

mm/bios_map.c

Page 3/3

```
127:                }
128:                bmmmap = (struct multiboot_mmap_entry *)((unsigned int)bm
map + bmmmap->size + sizeof(bmmmap->size));
129:                }
130:                kstat.physical_pages += (1024 >> 2);    /* add the first MB as a
whole */
131:                if(kstat.physical_pages > (0x40000000 >> PAGE_SHIFT)) {
132:                    printk("WARNING: detected a total of %dMB of available m
emory below 4GB.\n", (kstat.physical_pages << 2) / 1024);
133:                }
134:            } else {
135:                printk("WARNING: your BIOS has not provided a memory map.\n");
136:                bios_mem_map[0].from = 0;
137:                bios_mem_map[0].to = _memsize * 1024;
138:                bios_mem_map[0].type = MULTIBOOT_MEMORY_AVAILABLE;
139:                bios_mem_map[1].from = 0x00100000;
140:                bios_mem_map[1].to = (_extmemsize + 1024) * 1024;
141:                bios_mem_map[1].type = MULTIBOOT_MEMORY_AVAILABLE;
142:                kstat.physical_pages = (_extmemsize + 1024) >> 2;
143:            }
144:
145:            /*
146:             * This truncates to 1GB since it's the maximum physical memory
147:             * currently supported.
148:             */
149:            if(kstat.physical_pages > (0x40000000 >> PAGE_SHIFT)) {
150:                kstat.physical_pages = (0x40000000 >> PAGE_SHIFT);
151:                printk("WARNING: only up to 1GB of physical memory will be used.
\n");
152:            }
153: }
```

mm/fault.c

Page 1/5

```

1: /*
2:  * fiwix/mm/fault.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/sigcontext.h>
10: #include <fiwix/asm.h>
11: #include <fiwix/mm.h>
12: #include <fiwix/process.h>
13: #include <fiwix/traps.h>
14: #include <fiwix/sched.h>
15: #include <fiwix/fs.h>
16: #include <fiwix/mman.h>
17: #include <fiwix/errno.h>
18: #include <fiwix/stdio.h>
19: #include <fiwix/string.h>
20:
21: /* send the SIGSEGV signal to the ofending process */
22: static void send_sigsegv(struct sigcontext *sc)
23: {
24: #if defined(CONFIG_VERBOSE_SEGFAULTS)
25:     dump_registers(14, sc);
26:     printk("Memory map:\n");
27:     show_vma_regions(current);
28: #endif
29:     send_sig(current, SIGSEGV);
30: }
31:
32: static int page_protection_violation(struct vma *vma, unsigned int cr2, struct s
igcontext *sc)
33: {
34:     unsigned int *pgdir;
35:     unsigned int *pgtbl;
36:     unsigned int pde, pte, addr;
37:     struct page *pg;
38:     int page;
39:
40:     pde = GET_PGDIR(cr2);
41:     pte = GET_PGTBL(cr2);
42:     pgdir = (unsigned int *)P2V(current->tss.cr3);
43:     pgtbl = (unsigned int *)P2V((pgdir[pde] & PAGE_MASK));
44:     page = (pgtbl[pte] & PAGE_MASK) >> PAGE_SHIFT;
45:
46:     pg = &page_table[page];
47:
48:     /* Copy On Write feature */
49:     if(pg->count > 1) {
50:         /* a page not marked as copy-on-write means it's read-only */
51:         if(!(pg->flags & PAGE_COW)) {
52:             printk("Oops!, page %d NOT marked for CoW.\n", pg->page)
;
53:             send_sigsegv(sc);
54:             return 0;
55:         }
56:         if(!(addr = kmalloc())) {
57:             printk("%s(): not enough memory!\n", __FUNCTION__);
58:             return 1;
59:         }
60:         current->rss++;
61:         memcpy_b((void *)addr, (void *)P2V((page << PAGE_SHIFT)), PAGE_S
IZE);
62:         pgtbl[pte] = V2P(addr) | PAGE_PRESENT | PAGE_RW | PAGE_USER;
63:         kfree(P2V((page << PAGE_SHIFT)));
64:         current->rss--;

```

mm/fault.c

Page 2/5

```

65:             invalidate_tlb();
66:             return 0;
67:         } else {
68:             /* last page of Copy On Write procedure */
69:             if(pg->count == 1) {
70:                 /* a page not marked as copy-on-write means it's read-on
ly */
71:                 if(!(pg->flags & PAGE_COW)) {
72:                     printk("Oops!, last page %d NOT marked for CoW.\
n", pg->page);
73:                     send_sigsegv(sc);
74:                     return 0;
75:                 }
76:                 pgtbl[pte] = (page << PAGE_SHIFT) | PAGE_PRESENT | PAGE_
RW | PAGE_USER;
77:                 invalidate_tlb();
78:                 return 0;
79:             }
80:         }
81:         printk("WARNING: %s(): page %d with pg->count = 0!\n", __FUNCTION__, pg-
>page);
82:         return 1;
83:     }
84: }
85: static int page_not_present(struct vma *vma, unsigned int cr2, struct sigcontext
*sc)
86: {
87:     unsigned int addr, file_offset;
88:     struct page *pg;
89:
90:     if(!vma) {
91:         if(cr2 >= (sc->oldesp - 32) && cr2 < KERNEL_BASE_ADDR) {
92:             if(!(vma = find_vma_region(KERNEL_BASE_ADDR - 1))) {
93:                 printk("WARNING: %s(): process %d doesn't have a
n stack region in vma!\n", __FUNCTION__, current->pid);
94:                 send_sigsegv(sc);
95:                 return 0;
96:             } else {
97:                 /* assuming stack will never reach heap */
98:                 vma->start = cr2;
99:                 vma->start = vma->start & PAGE_MASK;
100:            }
101:        }
102:    }
103:
104:    /* if still a non-valid vma is found then kill the process! */
105:    if(!vma || vma->prot == PROT_NONE) {
106:        send_sigsegv(sc);
107:        return 0;
108:    }
109:
110:    /* fill the page with its corresponding file content */
111:    if(vma->inode) {
112:        file_offset = (cr2 & PAGE_MASK) - vma->start + vma->offset;
113:        file_offset &= PAGE_MASK;
114:        pg = NULL;
115:
116:        if(!(vma->prot & PROT_WRITE) || vma->flags & MAP_SHARED) {
117:            /* check if it's already in cache */
118:            if((pg = search_page_hash(vma->inode, file_offset))) {
119:                if(!map_page(current, cr2, (unsigned int)V2P(pg-
>data), vma->prot)) {
120:                    printk("%s(): Oops, map_page() returned
0!\n", __FUNCTION__);
121:                    return 1;
122:                }
123:                addr = (unsigned int)pg->data;

```


mm/fault.c

Page 3/5

```

124:             }
125:         }
126:         if(!pg) {
127:             if(!(addr = map_page(current, cr2, 0, vma->prot))) {
128:                 printk("%s(): Oops, map_page() returned 0!\n", _
_FUNCTION__);
129:                 return 1;
130:             }
131:             pg = &page_table[V2P(addr) >> PAGE_SHIFT];
132:             if(bread_page(pg, vma->inode, file_offset, vma->prot, vm
a->flags)) {
133:                 unmap_page(cr2);
134:                 return 1;
135:             }
136:             current->usage.ru_majflt++;
137:         }
138:     } else {
139:         current->usage.ru_minflt++;
140:         addr = 0;
141:     }
142:
143:     if(vma->flags & ZERO_PAGE) {
144:         if(!addr) {
145:             if(!(addr = map_page(current, cr2, 0, vma->prot))) {
146:                 printk("%s(): Oops, map_page() returned 0!\n", _
_FUNCTION__);
147:                 return 1;
148:             }
149:         }
150:         memset_b((void *) (addr & PAGE_MASK), NULL, PAGE_SIZE);
151:     }
152:
153:     return 0;
154: }
155:
156: /*
157:  * Exception 0xE: Page Fault
158:  *
159:  *           +-----+-----+-----+-----+-----+-----+
160:  *           | user | kernel | PV   | PF   | read | write |
161:  * +-----+-----+-----+-----+-----+-----+
162:  * | the page | U1   | K1   | U1 K1 | U1 K1 | K1   |
163:  * | has      | U2   | K2   | U2   | K2   | U2 K2 |
164:  * | a vma region | U3   |     |     | U3   | U3   |
165:  * +-----+-----+-----+-----+-----+-----+
166:  * | the page | U1   | K1   | U1 K1 | K1   | U1 K1 |
167:  * | doesn't have | U2   |     |     | U2   | U2   |
168:  * | a vma region |     |     |     |     | U2   |
169:  * +-----+-----+-----+-----+-----+-----+
170:  *
171:  * U1 - vma + user + PV + read
172:  *       (vma page in user-mode, page-violation during read)
173:  *       U1.1) if flags match                -> Demand paging
174:  *       U1.2) if flags don't match          -> SIGSEV
175:  *
176:  * U2 - vma + user + PV + write
177:  *       (vma page in user-mode, page-violation during write)
178:  *       U2.1) if flags match                -> Copy-On-Write
179:  *       U2.2) if flags don't match          -> SIGSEGV
180:  *
181:  * U3 - vma + user + PF + (read | write)     -> Demand paging
182:  *       (vma page in user-mode, page-fault during read or write)
183:  *
184:  * K1 - vma + kernel + PV + (read | write)   -> PANIC
185:  *       (vma page in kernel-mode, page-violation during read or write)
186:  * K2 - vma + kernel + PF + (read | write)   -> Demand paging (mmap)
187:  *       (vma page in kernel-mode, page-fault during read or write)

```

mm/fault.c

Page 4/5

```

188:  *
189:  * -----
190:  *
191:  * U1 - !vma + user + PV + (read | write)      -> SIGSEGV
192:  *       (!vma page in user-mode, page-violation during read or write)
193:  * U2 - !vma + user + PF + (read | write)      -> STACK grows
194:  *       (!vma page in user-mode, page-fault during read or write)
195:  *
196:  * K1 - !vma + kernel + (PV | PF) + (read | write)  -> PANIC
197:  *       (!vma page in kernel-mode, page-fault or page-violation during read
198:  *       or write)
199:  */
200: void do_page_fault(unsigned int trap, struct sigcontext *sc)
201: {
202:     unsigned int cr2;
203:     struct vma *vma;
204:
205:     GET_CR2(cr2);
206:     if((vma = find_vma_region(cr2))) {
207:
208:         /* in user mode */
209:         if(sc->err & PFAULT_U) {
210:             if(sc->err & PFAULT_V) {          /* violation */
211:                 if(sc->err & PFAULT_W) {
212:                     if((page_protection_violation(vma, cr2,
sc))) {
213:                         send_sig(current, SIGKILL);
214:                     }
215:                     return;
216:                 }
217:                 send_sigsegv(sc);
218:             } else {                          /* page not present */
219:                 if((page_not_present(vma, cr2, sc))) {
220:                     send_sig(current, SIGKILL);
221:                 }
222:             }
223:             return;
224:
225:             /* in kernel mode */
226:         } else {
227:             /*
228:              * WP bit marks the order: first check if the page is
229:              * present, then check for protection violation.
230:              */
231:             if(!(sc->err & PFAULT_V)) {      /* page not present */
232:                 if((page_not_present(vma, cr2, sc))) {
233:                     send_sig(current, SIGKILL);
234:                     printk("%s(): kernel was unable to read
a page of process '%s' (pid %d).\n", __FUNCTION__, current->argv0, current->pid);
235:                 }
236:                 return;
237:             }
238:             if(sc->err & PFAULT_W) {        /* copy-on-write? */
239:                 if((page_protection_violation(vma, cr2, sc))) {
240:                     send_sig(current, SIGKILL);
241:                     printk("%s(): kernel was unable to write
a page of process '%s' (pid %d).\n", __FUNCTION__, current->argv0, current->pid);
242:                 }
243:                 return;
244:             }
245:         }
246:     } else {
247:         /* in user mode */
248:         if(sc->err & PFAULT_U) {
249:             if(sc->err & PFAULT_V) {          /* violation */
250:                 send_sigsegv(sc);
251:             } else {                          /* stack? */

```

mm/fault.c

Page 5/5

```
252:                                if((page_not_present(vma, cr2, sc)) {
253:                                    send_sig(current, SIGKILL);
254:                                }
255:                                }
256:                                return;
257:                                }
258:                                }
259:
260:                                dump_registers(trap, sc);
261:                                show_vma_regions(current);
262:                                PANIC("\n");
263: }
```

mm/Makefile

Page 1/1

```
1: # fiwix/mm/Makefile
2: #
3: # Copyright 2018, Jordi Sanfeliu. All rights reserved.
4: # Distributed under the terms of the Fiwix License.
5: #
6:
7: .S.o:
8:     $(CC) -traditional -I$(INCLUDE) -c -o $@ $<
9: .c.o:
10:     $(CC) $(CFLAGS) -c -o $@ $<
11:
12: OBJS = bios_map.o memory.o page.o alloc.o fault.o mmap.o swapper.o
13:
14: mm:     $(OBJS)
15:         $(LD) $(LDFLAGS) -r $(OBJS) -o mm.o
16:
17: clean:
18:         rm -f *.o
19:
```

mm/memory.c

Page 1/8

```

1: /*
2:  * fiwix/mm/memory.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/asm.h>
10: #include <fiwix/mm.h>
11: #include <fiwix/mman.h>
12: #include <fiwix/bios.h>
13: #include <fiwix/ramdisk.h>
14: #include <fiwix/process.h>
15: #include <fiwix/buffer.h>
16: #include <fiwix/fs.h>
17: #include <fiwix/filesystems.h>
18: #include <fiwix/stdio.h>
19: #include <fiwix/string.h>
20:
21: #define KERNEL_TEXT_SIZE      ((int)_etext - (KERNEL_BASE_ADDR + KERNEL_ENTRY_
ADDR))
22: #define KERNEL_DATA_SIZE      ((int)_edata - (int)_etext)
23: #define KERNEL_BSS_SIZE       ((int)_end - (int)_edata)
24:
25: #define PGDIR_4MB_ADDR        0x50000
26:
27: unsigned int *kpage_dir;
28: unsigned int *kpage_table;
29:
30: unsigned int _last_data_addr;
31:
32: unsigned int proc_table_size = 0;
33: struct proc *proc_table;
34:
35: unsigned int buffer_table_size = 0;
36: unsigned int buffer_hash_table_size = 0;
37: struct buffer *buffer_table;
38: struct buffer **buffer_hash_table;
39:
40: unsigned int inode_table_size = 0;
41: unsigned int inode_hash_table_size = 0;
42: struct inode *inode_table;
43: struct inode **inode_hash_table;
44:
45: unsigned int fd_table_size = 0;
46: struct fd *fd_table;
47:
48: unsigned int mount_table_size = 0;
49: struct mount *mount_table;
50:
51: unsigned int page_table_size = 0;
52: unsigned int page_hash_table_size = 0;
53: struct page *page_table;
54: struct page **page_hash_table;
55:
56: static void map_kaddr(unsigned int from, unsigned int to, int flags)
57: {
58:     unsigned int n;
59:     unsigned int *pgtbl;
60:     unsigned int pde, pte;
61:
62:     for(n = from >> PAGE_SHIFT; n < (to >> PAGE_SHIFT); n++) {
63:         pde = GET_PGDIR(n << PAGE_SHIFT);
64:         pte = GET_PGTBL(n << PAGE_SHIFT);
65:         if(!(kpage_dir[pde] & ~PAGE_MASK)) {
66:             unsigned int addr;

```

mm/memory.c

Page 2/8

```

67:         addr = _last_data_addr;
68:         _last_data_addr += PAGE_SIZE;
69:         kpage_dir[pde] = addr | flags;
70:         memset_b((void *)addr, NULL, PAGE_SIZE);
71:     }
72:     pgtbl = (unsigned int *) (kpage_dir[pde] & PAGE_MASK);
73:     pgtbl[pte] = (n << PAGE_SHIFT) | flags;
74: }
75: }
76:
77: void bss_init(void)
78: {
79:     memset_b((void *) ((int)_edata), NULL, KERNEL_BSS_SIZE);
80: }
81:
82: /*
83:  * This function creates a minimal Page Directory covering only the first 4MB
84:  * of physical memory. Just enough to boot the kernel.
85:  * (it returns the address to be used by the CR3 register)
86:  */
87: unsigned int setup_minmem(void)
88: {
89:     int n;
90:     unsigned int addr;
91:     short int pd, mb4;
92:
93:     mb4 = 1;          /* 4MB units */
94:     addr = KERNEL_BASE_ADDR + PGDIR_4MB_ADDR;
95:
96:     kpage_dir = (unsigned int *)addr;
97:     memset_b(kpage_dir, NULL, PAGE_SIZE);
98:
99:     addr += PAGE_SIZE;
100:    kpage_table = (unsigned int *)addr;
101:    memset_b(kpage_table, NULL, PAGE_SIZE * mb4);
102:
103:    for(n = 0; n < (1024 * mb4); n++) {
104:        kpage_table[n] = (n << PAGE_SHIFT) | PAGE_PRESENT | PAGE_RW;
105:        if(!(n % 1024)) {
106:            pd = n / 1024;
107:            kpage_dir[pd] = (unsigned int) (addr + (PAGE_SIZE * pd) +
0x40000000) | PAGE_PRESENT | PAGE_RW;
108:            kpage_dir[GET_PGDIR(KERNEL_BASE_ADDR) + pd] = (unsigned
int) (addr + (PAGE_SIZE * pd) + 0x40000000) | PAGE_PRESENT | PAGE_RW;
109:        }
110:    }
111:    return (unsigned int)kpage_dir + 0x40000000;
112: }
113:
114: /* returns the mapped address of a virtual address */
115: unsigned int get_mapped_addr(struct proc *p, unsigned int addr)
116: {
117:     unsigned int *pgdir, *pgtbl;
118:     unsigned int pde, pte;
119:
120:     pgdir = (unsigned int *)P2V(p->tss.cr3);
121:     pde = GET_PGDIR(addr);
122:     pte = GET_PGTBL(addr);
123:     pgtbl = (unsigned int *)P2V((pgdir[pde] & PAGE_MASK));
124:     return pgtbl[pte];
125: }
126:
127: int clone_pages(struct proc *child)
128: {
129:     unsigned int *src_pgdir, *dst_pgdir;
130:     unsigned int *src_pgtbl, *dst_pgtbl;
131:     unsigned int pde, pte;

```

mm/memory.c

Page 3/8

```

132:     unsigned int p_addr, c_addr;
133:     int n, n2, pages;
134:     struct page *pg;
135:     struct vma *vma;
136:
137:     src_pgdir = (unsigned int *)P2V(current->tss.cr3);
138:     dst_pgdir = (unsigned int *)P2V(child->tss.cr3);
139:     vma = current->vma;
140:
141:     for(n = 0, pages = 0; n < VMA_REGIONS && vma->start; n++, vma++) {
142:         for(n2 = vma->start; n2 < vma->end; n2 += PAGE_SIZE) {
143:             if(vma->flags & MAP_SHARED) {
144:                 continue;
145:             }
146:             pde = GET_PGDIR(n2);
147:             pte = GET_PGTBL(n2);
148:             if(src_pgdir[pde] & PAGE_PRESENT) {
149:                 src_pgtbl = (unsigned int *)P2V((src_pgdir[pde]
& PAGE_MASK));
150:                 if(!(dst_pgdir[pde] & PAGE_PRESENT)) {
151:                     if(!(c_addr = kmalloc())) {
152:                         printk("%s(): returning 0!\n", __
_FUNCTION__);
153:                         return 0;
154:                     }
155:                     current->rss++;
156:                     pages++;
157:                     dst_pgdir[pde] = V2P(c_addr) | PAGE_PRES
ENT | PAGE_RW | PAGE_USER;
158:                     memset_b((void *)c_addr, NULL, PAGE_SIZE
);
159:                 }
160:                 dst_pgtbl = (unsigned int *)P2V((dst_pgdir[pde]
& PAGE_MASK));
161:                 if(src_pgtbl[pte] & PAGE_PRESENT) {
162:                     p_addr = src_pgtbl[pte] >> PAGE_SHIFT;
163:                     pg = &page_table[p_addr];
164:                     if(pg->flags & PAGE_RESERVED) {
165:                         continue;
166:                     }
167:                     src_pgtbl[pte] &= ~PAGE_RW;
168:                     /* mark write-only pages as copy-on-writ
e */
169:                     if(vma->prot & PROT_WRITE) {
170:                         pg->flags |= PAGE_COW;
171:                     }
172:                     dst_pgtbl[pte] = src_pgtbl[pte];
173:                     if(!is_valid_page((dst_pgtbl[pte] & PAGE
_MASK) >> PAGE_SHIFT)) {
174:                         PANIC("%s: missing page %d durin
g copy-on-write process.\n", __FUNCTION__, (dst_pgtbl[pte] & PAGE_MASK) >> PAGE_SHIFT);
175:                     }
176:                     pg = &page_table[(dst_pgtbl[pte] & PAGE_
MASK) >> PAGE_SHIFT];
177:                     pg->count++;
178:                 }
179:             }
180:         }
181:     }
182:     return pages;
183: }
184:
185: int free_page_tables(struct proc *p)
186: {
187:     unsigned int *pgdir;
188:     int n, count;
189:

```

mm/memory.c

Page 4/8

```

190:     pgdir = (unsigned int *)P2V(p->tss.cr3);
191:     for(n = 0, count = 0; n < PD_ENTRIES; n++) {
192:         if((pgdir[n] & (PAGE_PRESENT | PAGE_RW | PAGE_USER)) == (PAGE_PR
ESENT | PAGE_RW | PAGE_USER)) {
193:             kfree(P2V(pgdir[n]) & PAGE_MASK);
194:             pgdir[n] = NULL;
195:             count++;
196:         }
197:     }
198:     return count;
199: }
200:
201: unsigned int map_page(struct proc *p, unsigned int vaddr, unsigned int addr, uns
igned int prot)
202: {
203:     unsigned int *pgdir, *pgtbl;
204:     unsigned int newaddr;
205:     int pde, pte;
206:
207:     pgdir = (unsigned int *)P2V(p->tss.cr3);
208:     pde = GET_PGDIR(vaddr);
209:     pte = GET_PGTBL(vaddr);
210:
211:     if(!(pgdir[pde] & PAGE_PRESENT)) {          /* allocating page table */
212:         if(!(newaddr = kmalloc())) {
213:             return 0;
214:         }
215:         p->rss++;
216:         pgdir[pde] = V2P(newaddr) | PAGE_PRESENT | PAGE_RW | PAGE_USER;
217:         memset_b((void *)newaddr, NULL, PAGE_SIZE);
218:     }
219:     pgtbl = (unsigned int *)P2V((pgdir[pde] & PAGE_MASK));
220:     if(!(pgtbl[pte] & PAGE_PRESENT)) {          /* allocating page */
221:         if(!addr) {
222:             if(!(addr = kmalloc())) {
223:                 return 0;
224:             }
225:             addr = V2P(addr);
226:             p->rss++;
227:         }
228:         pgtbl[pte] = addr | PAGE_PRESENT | PAGE_USER;
229:     }
230:     if(prot & PROT_WRITE) {
231:         pgtbl[pte] |= PAGE_RW;
232:     }
233:     return P2V(addr);
234: }
235:
236: int unmap_page(unsigned int vaddr)
237: {
238:     unsigned int *pgdir, *pgtbl;
239:     unsigned int addr;
240:     int pde, pte;
241:
242:     pgdir = (unsigned int *)P2V(current->tss.cr3);
243:     pde = GET_PGDIR(vaddr);
244:     pte = GET_PGTBL(vaddr);
245:     if(!(pgdir[pde] & PAGE_PRESENT)) {
246:         printk("WARNING: %s(): trying to unmap an unallocated pde '0x%08
x'\n", __FUNCTION__, vaddr);
247:         return 1;
248:     }
249:
250:     pgtbl = (unsigned int *)P2V((pgdir[pde] & PAGE_MASK));
251:     if(!(pgtbl[pte] & PAGE_PRESENT)) {
252:         printk("WARNING: %s(): trying to unmap an unallocated page '0x%0
8x'\n", __FUNCTION__, vaddr);

```


mm/memory.c

Page 5/8

```

253:             return 1;
254:         }
255:
256:         addr = pgtbl[pte] & PAGE_MASK;
257:         pgtbl[pte] = NULL;
258:         kfree(P2V(addr));
259:         current->rss--;
260:         return 0;
261:     }
262:
263: void mem_init(void)
264: {
265:     unsigned int sizek;
266:     unsigned int physical_page_tables;
267:     unsigned int physical_memory;
268:     int n, pages;
269:
270:     physical_page_tables = (kstat.physical_pages / 1024) + ((kstat.physical_
pages % 1024) ? 1 : 0);
271:     physical_memory = (kstat.physical_pages << PAGE_SHIFT); /* in bytes */
272:
273:     /* Page Directory will be aligned to the next page */
274:     _last_data_addr = PAGE_ALIGN(_last_data_addr);
275:     kpage_dir = (unsigned int *)_last_data_addr;
276:     memset_b(kpage_dir, NULL, PAGE_SIZE);
277:     _last_data_addr += PAGE_SIZE;
278:
279:     /* Page Tables */
280:     kpage_table = (unsigned int *)_last_data_addr;
281:     memset_b(kpage_table, NULL, physical_page_tables * PAGE_SIZE);
282:     _last_data_addr += physical_page_tables * PAGE_SIZE;
283:
284:     /* Page Directory and Page Tables initialization */
285:     for(n = 0; n < kstat.physical_pages; n++) {
286:         kpage_table[n] = (n << PAGE_SHIFT) | PAGE_PRESENT | PAGE_RW;
287:         if(!(n % 1024)) {
288:             kpage_dir[GET_PGDIR(KERNEL_BASE_ADDR) + (n / 1024)] = (u
signed int)&kpage_table[n] | PAGE_PRESENT | PAGE_RW;
289:         }
290:     }
291:
292:     map_kaddr(KERNEL_ENTRY_ADDR, _last_data_addr, PAGE_PRESENT | PAGE_RW);
293:
294:     /*
295:      * FIXME: this is ugly!
296:      * It should go in console_init() once we have a proper kernel memory/pa
ge management.
297:      * Then map_kaddr will be a public function (not static).
298:      */
299:     if(video.flags & VPF_VGA) {
300:         map_kaddr(0xA0000, 0xA0000 + video.memsize, PAGE_PRESENT | PAGE_
RW);
301:     };
302:     if(video.flags & VPF_VESAFB) {
303:         map_kaddr((unsigned int)video.address, (unsigned int)video.address + video.memsize, PAGE_PRESENT | PAGE_RW);
304:     }
305:     /* printk("_last_data_addr = 0x%08x-0x%08x (kernel)\n", KERNEL_ENTRY_ADDR,
_last_data_addr); */
306:     activate_kpage_dir();
307:
308:     /* since Page Directory is now activated we can use virtual addresses */
309:     _last_data_addr = P2V(_last_data_addr);
310:
311:
312:     /* reserve memory space for proc_table[NR_PROCS] */
313:     proc_table_size = PAGE_ALIGN(sizeof(struct proc) * NR_PROCS);

```

mm/memory.c

Page 6/8

```

314:         if(!addr_in_bios_map(V2P(_last_data_addr) + proc_table_size)) {
315:             PANIC("Not enough memory for proc_table.\n");
316:         }
317: /*      printk("_last_data_addr = 0x%08x-0x%08x (proc_table)\n", _last_data_addr
, _last_data_addr + proc_table_size); */
318:     proc_table = (struct proc *)_last_data_addr;
319:     _last_data_addr += proc_table_size;
320:
321:
322:     /* reserve memory space for buffer_table */
323:     buffer_table_size = (kstat.physical_pages * BUFFER_PERCENTAGE) / 100;
324:     buffer_table_size *= sizeof(struct buffer);
325:     pages = buffer_table_size >> PAGE_SHIFT;
326:     buffer_table_size = !pages ? 4096 : pages << PAGE_SHIFT;
327: /*      printk("_last_data_addr = 0x%08x-0x%08x (buffer_table)\n", _last_data_ad
dr, _last_data_addr + buffer_table_size); */
328:     if(!addr_in_bios_map(V2P(_last_data_addr) + buffer_table_size)) {
329:         PANIC("Not enough memory for buffer_table.\n");
330:     }
331:     buffer_table = (struct buffer *)_last_data_addr;
332:     _last_data_addr += buffer_table_size;
333:
334:
335:     /* reserve memory space for buffer_hash_table */
336:     n = (buffer_table_size / sizeof(struct buffer) * BUFFER_HASH_PERCENTAGE)
/ 100;
337:     n = MAX(n, 10); /* 10 buffer hashes as minimum */
338:     /* buffer_hash_table is an array of pointers */
339:     pages = ((n * sizeof(unsigned int)) / PAGE_SIZE) + 1;
340:     buffer_hash_table_size = pages << PAGE_SHIFT;
341: /*      printk("_last_data_addr = 0x%08x-0x%08x (buffer_hash_table)\n", _last_da
ta_addr, _last_data_addr + buffer_hash_table_size); */
342:     if(!addr_in_bios_map(V2P(_last_data_addr) + buffer_hash_table_size)) {
343:         PANIC("Not enough memory for buffer_hash_table.\n");
344:     }
345:     buffer_hash_table = (struct buffer **)_last_data_addr;
346:     _last_data_addr += buffer_hash_table_size;
347:
348:
349:     /* reserve memory space for inode_table */
350:     sizek = physical_memory / 1024; /* this helps to avoid overflow */
351:     inode_table_size = (sizek * INODE_PERCENTAGE) / 100;
352:     inode_table_size *= 1024;
353:     pages = inode_table_size >> PAGE_SHIFT;
354:     inode_table_size = pages << PAGE_SHIFT;
355: /*      printk("_last_data_addr = 0x%08x-0x%08x (inode_table)\n", _last_data_add
r, _last_data_addr + inode_table_size); */
356:     if(!addr_in_bios_map(V2P(_last_data_addr) + inode_table_size)) {
357:         PANIC("Not enough memory for inode_table.\n");
358:     }
359:     inode_table = (struct inode *)_last_data_addr;
360:     _last_data_addr += inode_table_size;
361:
362:
363:     /* reserve memory space for inode_hash_table */
364:     n = ((inode_table_size / sizeof(struct inode)) * INODE_HASH_PERCENTAGE)
/ 100;
365:     n = MAX(n, 10); /* 10 inode hash buckets as minimum */
366:     /* inode_hash_table is an array of pointers */
367:     pages = ((n * sizeof(unsigned int)) / PAGE_SIZE) + 1;
368:     inode_hash_table_size = pages << PAGE_SHIFT;
369: /*      printk("_last_data_addr = 0x%08x-0x%08x (inode_hash_table)\n", _last_dat
a_addr, _last_data_addr + inode_hash_table_size); */
370:     if(!addr_in_bios_map(V2P(_last_data_addr) + inode_hash_table_size)) {
371:         PANIC("Not enough memory for inode_hash_table.\n");
372:     }
373:     inode_hash_table = (struct inode **)_last_data_addr;

```

mm/memory.c

Page 7/8

```

374:     _last_data_addr += inode_hash_table_size;
375:
376:
377:     /* reserve memory space for fd_table[NR_OPENS] */
378:     fd_table_size = PAGE_ALIGN(sizeof(struct fd) * NR_OPENS);
379: /*     printk("_last_data_addr = 0x%08x-0x%08x (fd_table)\n", _last_data_addr,
_last_data_addr + fd_table_size); */
380:     if(!addr_in_bios_map(V2P(_last_data_addr) + fd_table_size)) {
381:         PANIC("Not enough memory for fd_table.\n");
382:     }
383:     fd_table = (struct fd *)_last_data_addr;
384:     _last_data_addr += fd_table_size;
385:
386:
387:     /* reserve memory space for mount_table[NR_MOUNT_POINTS] */
388:     mount_table_size = PAGE_ALIGN(sizeof(struct mount) * NR_MOUNT_POINTS);
389: /*     printk("_last_data_addr = 0x%08x-0x%08x (mount_table)\n", _last_data_add
r, _last_data_addr + mount_table_size); */
390:     if(!addr_in_bios_map(V2P(_last_data_addr) + mount_table_size)) {
391:         PANIC("Not enough memory for mount_table.\n");
392:     }
393:     mount_table = (struct mount *)_last_data_addr;
394:     _last_data_addr += mount_table_size;
395:
396:
397:     /* reserve memory space for RAMdisk(s) */
398:     if(!_noramdisk) {
399:         if(!_ramdisksize) {
400:             _ramdisksize = RAMDISK_SIZE;
401:         }
402:         /*
403:          * If the 'initrd=' parameter was supplied, then the first
404:          * ramdisk device was already assigned to the initial ramdisk
405:          * image.
406:          */
407:         if(ramdisk_table[0].addr) {
408:             n = 1;
409:         } else {
410:             n = 0;
411:         }
412:         for(; n < RAMDISK_MINORS; n++) {
413:             if(!addr_in_bios_map(V2P(_last_data_addr) + (_ramdisksiz
e * 1024))) {
414:                 printk("WARNING: RAMdisk device disabled (not en
ough physical memory).\n");
415:                 _noramdisk = 1;
416:                 break;
417:             }
418: /*     printk("_last_data_addr = 0x%08x-0x%08x (/dev/ram%d)\n",
_last_data_addr, _last_data_addr + (_ramdisksize * 1024), n); */
419:             ramdisk_table[n].addr = (char *)_last_data_addr;
420:             _last_data_addr += _ramdisksize * 1024;
421:         }
422:     }
423:
424:     /*
425:      * FIXME: this is ugly!
426:      * It should go in console_init() once we have a proper kernel memory/pa
ge management.
427:      */
428:     #include <fiwix/console.h>
429:     for(n = 1; n <= NR_VCONSOLES; n++) {
430:         vc_screen[n] = (short int *)_last_data_addr;
431:         _last_data_addr += (video.columns * video.lines * 2);
432:     }
433:     /*
434:      * FIXME: this is ugly!

```

mm/memory.c

Page 8/8

```

435:      * It should go in console_init() once we have a proper kernel memory/pa
ge management.
436:      */
437:      vcbuf = (short int *)_last_data_addr;
438:      _last_data_addr += (video.columns * video.lines * SCREENS_LOG * 2 * size
of(short int));
439:
440:
441:      /* the last one must be the page_table structure */
442:      page_hash_table_size = 1 * PAGE_SIZE; /* only 1 page size */
443:      if(!addr_in_bios_map(V2P(_last_data_addr) + page_hash_table_size)) {
444:          PANIC("Not enough memory for page_hash_table.\n");
445:      }
446:      page_hash_table = (struct page **)_last_data_addr;
447: /*      printk("_last_data_addr = 0x%08x-0x%08x (page_hash_table)\n", _last_data
_addr, _last_data_addr + page_hash_table_size); */
448:      _last_data_addr += page_hash_table_size;
449:
450:      page_table_size = PAGE_ALIGN(kstat.physical_pages * sizeof(struct page))
;
451:      if(!addr_in_bios_map(V2P(_last_data_addr) + page_table_size)) {
452:          PANIC("Not enough memory for page_table.\n");
453:      }
454:      page_table = (struct page *)_last_data_addr;
455: /*      printk("page_table_size = %d\n", page_table_size); */
456: /*      printk("_last_data_addr = 0x%08x-0x%08x (page_table)\n", _last_data_addr
, _last_data_addr + page_table_size); */
457:      _last_data_addr += page_table_size;
458:
459:      page_init(kstat.physical_pages);
460: }
461:
462: void mem_stats(void)
463: {
464:     printk("\n");
465:     printk("memory: total=%dKB, user=%dKB, kernel=%dKB, reserved=%dKB\n", ks
tat.physical_pages << 2, kstat.total_mem_pages << 2, kstat.kernel_reserved, kstat.physi
cal_reserved);
466:     printk("kernel: text=%dKB, data=%dKB, bss=%dKB, i/o buffers=%d (%dKB)\n"
, KERNEL_TEXT_SIZE / 1024, KERNEL_DATA_SIZE / 1024, KERNEL_BSS_SIZE / 1024, buffer_tabl
e_size / sizeof(struct buffer), (buffer_table_size + buffer_hash_table_size) / 1024);
467:     printk("\tinodes=%d (%dKB)\n\n", inode_table_size / sizeof(struct inode)
, (inode_table_size + inode_hash_table_size) / 1024);
468: }

```

mm/mmap.c

Page 1/8

```

1: /*
2:  * fiwix/mm/mmap.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/mm.h>
10: #include <fiwix/fs.h>
11: #include <fiwix/fcntl.h>
12: #include <fiwix/stat.h>
13: #include <fiwix/process.h>
14: #include <fiwix/mman.h>
15: #include <fiwix/errno.h>
16: #include <fiwix/stdio.h>
17: #include <fiwix/string.h>
18:
19: void show_vma_regions(struct proc *p)
20: {
21:     __ino_t inode;
22:     int major, minor;
23:     char *section;
24:     char r, w, x, f;
25:     struct vma *vma;
26:     unsigned int n;
27:     int count;
28:
29:     vma = p->vma;
30:     printk("num  address range          flag offset      dev  inode      mod
section cnt\n");
31:     printk("-----\n");
32:     for(n = 0; n < VMA_REGIONS && vma->start; n++, vma++) {
33:         r = vma->prot & PROT_READ ? 'r' : '-';
34:         w = vma->prot & PROT_WRITE ? 'w' : '-';
35:         x = vma->prot & PROT_EXEC ? 'x' : '-';
36:         if(vma->flags & MAP_SHARED) {
37:             f = 's';
38:         } else if(vma->flags & MAP_PRIVATE) {
39:             f = 'p';
40:         } else {
41:             f = '-';
42:         }
43:         switch(vma->s_type) {
44:             case P_TEXT:    section = "text ";
45:                             break;
46:             case P_DATA:    section = "data ";
47:                             break;
48:             case P_BSS:     section = "bss  ";
49:                             break;
50:             case P_HEAP:    section = "heap ";
51:                             break;
52:             case P_STACK:   section = "stack";
53:                             break;
54:             case P_MMAP:    section = "mmap ";
55:                             break;
56:             default:
57:                 section = NULL;
58:                 break;
59:         }
60:         inode = major = minor = count = 0;
61:         if(vma->inode) {
62:             inode = vma->inode->inode;
63:             major = MAJOR(vma->inode->dev);
64:             minor = MINOR(vma->inode->dev);
65:             count = vma->inode->count;

```

mm/mmap.c

Page 2/8

```

66:         }
67:         printk("[%02d] 0x%08x-0x%08x %c%c%c%c 0x%08x %02d:%02d %- 10u <%=
d> [%s] (%d)\n", n, vma->start, vma->end, r, w, x, f, vma->offset, major, minor, inode
, vma->o_mode, section, count);
68:     }
69:     if(!n) {
70:         printk("[no vma regions]\n");
71:     }
72: }
73:
74: static struct vma * get_new_vma_region(void)
75: {
76:     unsigned int n;
77:     struct vma *vma;
78:
79:     vma = current->vma;
80:
81:     for(n = 0; n < VMA_REGIONS; n++, vma++) {
82:         if(!vma->start && !vma->end) {
83:             return vma;
84:         }
85:     }
86:     return NULL;
87: }
88:
89: /*
90:  * This sorts regions (in ascending order), merging equal regions and keeping
91:  * the unused ones at the end of the array.
92:  */
93: static void sort_vma(void)
94: {
95:     unsigned int n, n2, needs_sort;
96:     struct vma *vma, tmp;
97:
98:     vma = current->vma;
99:
100:    do {
101:        needs_sort = 0;
102:        for(n = 0, n2 = 1; n2 < VMA_REGIONS; n++, n2++) {
103:            if(vma[n].end && vma[n2].start) {
104:                if((vma[n].end == vma[n2].start) &&
105:                    (vma[n].prot == vma[n2].prot) &&
106:                    (vma[n].flags == vma[n2].flags) &&
107:                    (vma[n].offset == vma[n2].offset) &&
108:                    (vma[n].s_type == vma[n2].s_type) &&
109:                    (vma[n].inode == vma[n2].inode)) {
110:                    vma[n].end = vma[n2].end;
111:                    memset_b(&vma[n2], NULL, sizeof(struct v
ma));
112:                    needs_sort++;
113:                }
114:            }
115:            if((vma[n2].start && (vma[n].start > vma[n2].start)) ||
(!vma[n].start && vma[n2].start)) {
116:                memcpy_b(&tmp, &vma[n], sizeof(struct vma));
117:                memcpy_b(&vma[n], &vma[n2], sizeof(struct vma));
118:                memcpy_b(&vma[n2], &tmp, sizeof(struct vma));
119:                needs_sort++;
120:            }
121:        }
122:    } while(needs_sort);
123: }
124:
125: /*
126:  * This function removes all redundant entries.
127:  *
128:  * for example, if for any reason the map looks like this:

```

mm/mmap.c

Page 3/8

```

129: * [01] 0x0808e984-0x08092000 rw-p 0x00000000 0
130: * [02] 0x0808f000-0x0808ffff rw-p 0x000c0000 4066
131: *
132: * this function converts it to this:
133: * [01] 0x0808e984-0x0808f000 rw-p 0x00000000 0
134: * [02] 0x0808f000-0x0808ffff rw-p 0x000c0000 4066
135: * [03] 0x08090000-0x08092000 rw-p 0x00000000 0
136: */
137: static int optimize_vma(void)
138: {
139:     unsigned int n, needs_sort;
140:     struct vma *vma, *prev, *new;
141:
142:     for(;;) {
143:         needs_sort = 0;
144:         prev = new = NULL;
145:         vma = current->vma;
146:         for(n = 0; n < VMA_REGIONS && vma->start; n++, vma++) {
147:             if(!prev) {
148:                 prev = vma;
149:                 continue;
150:             }
151:             if((vma->start < prev->end)) {
152:                 if(!(new = get_new_vma_region())) {
153:                     printk("WARNING: %s(): unable to get a f
ree vma region.\n", __FUNCTION__);
154:                     return -ENOMEM;
155:                 }
156:                 new->start = vma->end;
157:                 new->end = prev->end;
158:                 new->prot = prev->prot;
159:                 new->flags = prev->flags;
160:                 new->offset = prev->offset;
161:                 new->s_type = prev->s_type;
162:                 new->inode = prev->inode;
163:                 new->o_mode = prev->o_mode;
164:                 prev->end = vma->start;
165:                 needs_sort++;
166:                 if(prev->start == prev->end) {
167:                     memset_b(prev, NULL, sizeof(struct vma))
;
168:                 }
169:                 if(new->start == new->end) {
170:                     memset_b(new, NULL, sizeof(struct vma));
171:                 }
172:                 break;
173:             }
174:             prev = vma;
175:         }
176:         if(!needs_sort) {
177:             break;
178:         }
179:         sort_vma();
180:     }
181:
182:     return 0;
183: }
184:
185: /* return the first free address that matches with the size of length */
186: static unsigned int get_unmapped_vma_region(unsigned int length)
187: {
188:     unsigned int n, addr;
189:     struct vma *vma;
190:
191:     if(!length) {
192:         return 0;
193:     }

```

mm/mmap.c

Page 4/8

```

194:
195:     addr = MMAP_START;
196:     vma = current->vma;
197:
198:     for(n = 0; n < VMA_REGIONS && vma->start; n++, vma++) {
199:         if(vma->start < MMAP_START) {
200:             continue;
201:         }
202:         if(vma->start - addr >= length) {
203:             return PAGE_ALIGN(addr);
204:         }
205:         addr = PAGE_ALIGN(vma->end);
206:     }
207:     return 0;
208: }
209:
210: static void free_vma_pages(unsigned int start, __size_t length, struct vma *vma)
211: {
212:     unsigned int n, addr;
213:     unsigned int *pgdir, *pgtbl;
214:     unsigned int pde, pte;
215:     struct page *pg;
216:     int page;
217:
218:     pgdir = (unsigned int *)P2V(current->tss.cr3);
219:     pgtbl = NULL;
220:
221:     for(n = 0; n < (length / PAGE_SIZE); n++) {
222:         pde = GET_PGDIR(start + (n * PAGE_SIZE));
223:         pte = GET_PGTBL(start + (n * PAGE_SIZE));
224:         if(pgdir[pde] & PAGE_PRESENT) {
225:             pgtbl = (unsigned int *)P2V((pgdir[pde] & PAGE_MASK));
226:             if(pgtbl[pte] & PAGE_PRESENT) {
227:                 /* make sure to not free reserved pages */
228:                 page = pgtbl[pte] >> PAGE_SHIFT;
229:                 pg = &page_table[page];
230:                 if(pg->flags & PAGE_RESERVED) {
231:                     continue;
232:                 }
233:
234:                 if(vma->prot & PROT_WRITE && vma->flags & MAP_SH
ARED) {
235:                     addr = start - vma->start + vma->offset;
236:                     write_page(pg, vma->inode, addr, length)
;
237:                 }
238:
239:                 kfree(P2V(pgtbl[pte]) & PAGE_MASK);
240:                 current->rss--;
241:                 pgtbl[pte] = NULL;
242:
243:                 /* check if a page table can be freed */
244:                 for(pte = 0; pte < PT_ENTRIES; pte++) {
245:                     if(pgtbl[pte] & PAGE_MASK) {
246:                         break;
247:                     }
248:                 }
249:                 if(pte == PT_ENTRIES) {
250:                     kfree((unsigned int)pgtbl & PAGE_MASK);
251:                     current->rss--;
252:                     pgdir[pde] = NULL;
253:                 }
254:             }
255:         }
256:     }
257: }
258:

```


mm/mmap.c

Page 5/8

```

259: static int free_vma_region(struct vma *vma, unsigned int start, __ssize_t length
)
260: {
261:     struct vma *new;
262:
263:     if(!(new = get_new_vma_region())) {
264:         printk("WARNING: %s(): unable to get a free vma region.\n", __FU
NCTION__);
265:         return -ENOMEM;
266:     }
267:
268:     new->start = start + length;
269:     new->end = vma->end;
270:     new->prot = vma->prot;
271:     new->flags = vma->flags;
272:     new->offset = vma->offset;
273:     new->s_type = vma->s_type;
274:     new->inode = vma->inode;
275:     new->o_mode = vma->o_mode;
276:
277:     vma->end = start;
278:
279:     if(vma->start == vma->end) {
280:         if(vma->inode) {
281:             iput(vma->inode);
282:         }
283:         memset_b(vma, NULL, sizeof(struct vma));
284:     }
285:     if(new->start == new->end) {
286:         memset_b(new, NULL, sizeof(struct vma));
287:     }
288:     return 0;
289: }
290:
291: void release_binary(void)
292: {
293:     unsigned int n;
294:     struct vma *vma;
295:
296:     vma = current->vma;
297:
298:     for(n = 0; n < VMA_REGIONS && vma->start; n++, vma++) {
299:         free_vma_pages(vma->start, vma->end - vma->start, vma);
300:         free_vma_region(vma, vma->start, vma->end - vma->start);
301:     }
302:     sort_vma();
303:     optimize_vma();
304:     invalidate_tlb();
305: }
306:
307: struct vma * find_vma_region(unsigned int addr)
308: {
309:     unsigned int n;
310:     struct vma *vma;
311:
312:     if(!addr) {
313:         return NULL;
314:     }
315:
316:     addr &= PAGE_MASK;
317:     vma = current->vma;
318:
319:     for(n = 0; n < VMA_REGIONS && vma->start; n++, vma++) {
320:         if((addr >= vma->start) && (addr < vma->end)) {
321:             return vma;
322:         }
323:     }

```

mm/mmap.c

Page 6/8

```

324:         return NULL;
325:     }
326:
327: int expand_heap(unsigned int new)
328: {
329:     unsigned int n;
330:     struct vma *vma, *heap;
331:
332:     vma = current->vma;
333:     heap = NULL;
334:
335:     for(n = 0; n < VMA_REGIONS && vma->start; n++, vma++) {
336:         /* make sure the new heap won't overlap the next region */
337:         if(heap && new < vma->start) {
338:             heap->end = new;
339:             return 0;
340:         } else {
341:             heap = NULL;    /* was a bad candidate */
342:         }
343:         if(!heap && vma->s_type == P_HEAP) {
344:             heap = vma;    /* possible candidate */
345:             continue;
346:         }
347:     }
348:
349:     /* out of memory! */
350:     return 1;
351: }
352:
353: int do_mmap(struct inode *i, unsigned int start, unsigned int length, unsigned i
nt prot, unsigned int flags, unsigned int offset, char type, char mode)
354: {
355:     struct vma *vma;
356:     int errno;
357:
358:     if(!(length = PAGE_ALIGN(length))) {
359:         return start;
360:     }
361:
362:     if(start > KERNEL_BASE_ADDR || start + length > KERNEL_BASE_ADDR) {
363:         return -EINVAL;
364:     }
365:
366:     /* file mapping */
367:     if(i) {
368:         if(!S_ISREG(i->i_mode) && !S_ISCHR(i->i_mode)) {
369:             return -ENODEV;
370:         }
371:
372:         /*
373:          * The file shall have been opened with read permission,
374:          * regardless of the protection options specified.
375:          * IEEE Std 1003.1, 2004 Edition.
376:          */
377:         if(mode == O_WRONLY) {
378:             return -EACCES;
379:         }
380:         switch(flags & MAP_TYPE) {
381:             case MAP_SHARED:
382:                 if(prot & PROT_WRITE) {
383:                     if(!(mode & (O_WRONLY | O_RDWR))) {
384:                         return -EACCES;
385:                     }
386:                 }
387:                 break;
388:             case MAP_PRIVATE:
389:                 break;

```

mm/mmap.c

Page 7/8

```

390:             default:
391:                 return -EINVAL;
392:         }
393:         i->count++;
394:
395:         /* anonymous mapping */
396:     } else {
397:         if((flags & MAP_TYPE) != MAP_PRIVATE) {
398:             return -EINVAL;
399:         }
400:
401:         /* anonymous objects must be filled with zeros */
402:         flags |= ZERO_PAGE;
403:     }
404:
405:     if(flags & MAP_FIXED) {
406:         if(start & ~PAGE_MASK) {
407:             return -EINVAL;
408:         }
409:     } else {
410:         start = get_unmapped_vma_region(length);
411:         if(!start) {
412:             printk("WARNING: %s(): unable to get an unmapped vma reg
ion.\n", __FUNCTION__);
413:             return -ENOMEM;
414:         }
415:     }
416:
417:     if(!(vma = get_new_vma_region())) {
418:         printk("WARNING: %s(): unable to get a free vma region.\n", __FU
NCTION__);
419:         return -ENOMEM;
420:     }
421:
422:     vma->start = start;
423:     vma->end = start + length;
424:     vma->prot = prot;
425:     vma->flags = flags;
426:     vma->offset = offset;
427:     vma->s_type = type;
428:     vma->inode = i;
429:     vma->o_mode = mode;
430:
431:     if(i && i->fsop->mmap) {
432:         if((errno = i->fsop->mmap(i, vma))) {
433:             int errno2;
434:
435:             if((errno2 = free_vma_region(vma, start, length))) {
436:                 return errno2;
437:             }
438:             sort_vma();
439:             if((errno2 = optimize_vma())) {
440:                 return errno2;
441:             }
442:             return errno;
443:         }
444:     }
445:
446:     sort_vma();
447:     if((errno = optimize_vma())) {
448:         return errno;
449:     }
450:     return start;
451: }
452:
453: int do_munmap(unsigned int addr, __size_t length)
454: {

```

mm/mmap.c

Page 8/8

```

455:     struct vma *vma;
456:     unsigned int size;
457:     int errno;
458:
459:     if((addr & ~PAGE_MASK) || length < 0) {
460:         return -EINVAL;
461:     }
462:
463:     length = PAGE_ALIGN(length);
464:
465:     while(length) {
466:         if((vma = find_vma_region(addr)) {
467:             if((addr + length) > vma->end) {
468:                 size = vma->end - addr;
469:             } else {
470:                 size = length;
471:             }
472:
473:             free_vma_pages(addr, size, vma);
474:             invalidate_tlb();
475:             if((errno = free_vma_region(vma, addr, size)) {
476:                 return errno;
477:             }
478:             sort_vma();
479:             if((errno = optimize_vma())) {
480:                 return errno;
481:             }
482:             length -= size;
483:             addr += size;
484:         } else {
485:             break;
486:         }
487:     }
488:
489:     return 0;
490: }
491:
492: int do_mprotect(struct vma *vma, unsigned int addr, __size_t length, int prot)
493: {
494:     struct vma *new;
495:     int errno;
496:
497:     if(!(new = get_new_vma_region())) {
498:         printk("WARNING: %s(): unable to get a free vma region.\n", __FU
NCTION__);
499:         return -ENOMEM;
500:     }
501:
502:     new->start = addr;
503:     new->end = addr + length;
504:     new->prot = prot;
505:     new->flags = vma->flags;
506:     new->offset = vma->offset;
507:     new->s_type = vma->s_type;
508:     new->inode = vma->inode;
509:     new->o_mode = vma->o_mode;
510:
511:     sort_vma();
512:     if((errno = optimize_vma())) {
513:         return errno;
514:     }
515:     return 0;
516: }

```

mm/page.c

Page 1/7

```

1: /*
2:  * fiwix/mm/page.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: /*
9:  * page.c implements a cache with a free list as a doubly circular linked
10:  * list and a chained hash table with doubly linked lists.
11:  *
12:  * hash table
13:  * +-----+ +-----+ +-----+ +-----+
14:  * | index | |prev|data|next| |prev|data|next| |prev|data|next|
15:  * |   0   --> | / |   |   |   | <--- |   |   |   | <--- |   |   |   |
16:  * +-----+ +-----+ +-----+ +-----+
17:  * +-----+ +-----+ +-----+ +-----+
18:  * | index | |prev|data|next| |prev|data|next| |prev|data|next|
19:  * |   1   --> | / |   |   |   | <--- |   |   |   | <--- |   |   |   |
20:  * +-----+ +-----+ +-----+ +-----+
21:  *                                     (page)      (page)      (page)
22:  *      ...
23:  */
24:
25: #include <fiwix/asm.h>
26: #include <fiwix/kernel.h>
27: #include <fiwix/mm.h>
28: #include <fiwix/mman.h>
29: #include <fiwix/bios.h>
30: #include <fiwix/sleep.h>
31: #include <fiwix/sched.h>
32: #include <fiwix/devices.h>
33: #include <fiwix/buffer.h>
34: #include <fiwix/errno.h>
35: #include <fiwix/stdio.h>
36: #include <fiwix/string.h>
37:
38: #define PAGE_HASH(inode, offset)      (((__ino_t)(inode) ^ (__off_t)(offset))
% (NR_PAGE_HASH))
39: #define NR_PAGES      (page_table_size / sizeof(struct page))
40: #define NR_PAGE_HASH  (page_hash_table_size / sizeof(unsigned int))
41:
42: struct page *page_table;          /* page pool */
43: struct page *page_head;          /* page pool head */
44: struct page **page_hash_table;
45:
46: static void insert_to_hash(struct page *pg)
47: {
48:     struct page **h;
49:     int i;
50:
51:     i = PAGE_HASH(pg->inode, pg->offset);
52:     h = &page_hash_table[i];
53:
54:     if(!*h) {
55:         *h = pg;
56:         (*h)->prev_hash = (*h)->next_hash = NULL;
57:     } else {
58:         pg->prev_hash = NULL;
59:         pg->next_hash = *h;
60:         (*h)->prev_hash = pg;
61:         *h = pg;
62:     }
63:     kstat.cached += (PAGE_SIZE / 1024);
64: }
65:
66: static void remove_from_hash(struct page *pg)

```

mm/page.c

Page 2/7

```

67: {
68:     struct page **h;
69:     int i;
70:
71:     if(!pg->inode) {
72:         return;
73:     }
74:
75:     i = PAGE_HASH(pg->inode, pg->offset);
76:     h = &page_hash_table[i];
77:
78:     while(*h) {
79:         if(*h == pg) {
80:             if((*h)->next_hash) {
81:                 (*h)->next_hash->prev_hash = (*h)->prev_hash;
82:             }
83:             if((*h)->prev_hash) {
84:                 (*h)->prev_hash->next_hash = (*h)->next_hash;
85:             }
86:             if(h == &page_hash_table[i]) {
87:                 *h = (*h)->next_hash;
88:             }
89:             kstat.cached -= (PAGE_SIZE / 1024);
90:             break;
91:         }
92:         h = &(*h)->next_hash;
93:     }
94: }
95:
96: static void insert_on_free_list(struct page *pg)
97: {
98:     if(!page_head) {
99:         pg->prev_free = pg->next_free = pg;
100:        page_head = pg;
101:    } else {
102:        pg->next_free = page_head;
103:        pg->prev_free = page_head->prev_free;
104:        page_head->prev_free->next_free = pg;
105:        page_head->prev_free = pg;
106:    }
107:
108:    kstat.free_pages++;
109: }
110:
111: static void remove_from_free_list(struct page *pg)
112: {
113:     if(!kstat.free_pages) {
114:         return;
115:     }
116:
117:    pg->prev_free->next_free = pg->next_free;
118:    pg->next_free->prev_free = pg->prev_free;
119:    kstat.free_pages--;
120:    if(pg == page_head) {
121:        page_head = pg->next_free;
122:    }
123:
124:    if(!kstat.free_pages) {
125:        page_head = NULL;
126:    }
127: }
128:
129: void page_lock(struct page *pg)
130: {
131:     unsigned long int flags;
132:
133:     for(;;) {

```

mm/page.c

Page 3/7

```

134:         SAVE_FLAGS(flags); CLI();
135:         if(pg->flags & PAGE_LOCKED) {
136:             RESTORE_FLAGS(flags);
137:             sleep(&pg, PROC_UNINTERRUPTIBLE);
138:         } else {
139:             break;
140:         }
141:     }
142:     pg->flags |= PAGE_LOCKED;
143:     RESTORE_FLAGS(flags);
144: }
145:
146: void page_unlock(struct page *pg)
147: {
148:     unsigned long int flags;
149:
150:     SAVE_FLAGS(flags); CLI();
151:     pg->flags &= ~PAGE_LOCKED;
152:     wakeup(pg);
153:     RESTORE_FLAGS(flags);
154: }
155:
156: struct page * get_free_page(void)
157: {
158:     unsigned long int flags;
159:     struct page *pg;
160:
161:     /* if no more pages on free list */
162:     if(!kstat.free_pages) {
163:         /* reclaim some memory from buffer cache */
164:         wakeup(&kswapd);
165:         sleep(&get_free_page, PROC_UNINTERRUPTIBLE);
166:
167:         if(!kstat.free_pages) {
168:             /* definitely out of memory! (no more pages) */
169:             printk("%s(): pid %d ran out of memory. OOM killer neede
d!\n", __FUNCTION__, current->pid);
170:             return NULL;
171:         }
172:     }
173:
174:     SAVE_FLAGS(flags); CLI();
175:
176:     pg = page_head;
177:     remove_from_free_list(pg);
178:     remove_from_hash(pg); /* remove it from its old hash */
179:     pg->count = 1;
180:     pg->inode = 0;
181:     pg->offset = 0;
182:     pg->dev = 0;
183:
184:     RESTORE_FLAGS(flags);
185:     return pg;
186: }
187:
188: struct page * search_page_hash(struct inode *inode, __off_t offset)
189: {
190:     struct page *pg;
191:     int i;
192:
193:     i = PAGE_HASH(inode->inode, offset);
194:     pg = page_hash_table[i];
195:
196:     while(pg) {
197:         if(pg->inode == inode->inode && pg->offset == offset && pg->dev
== inode->dev) {
198:             if(!pg->count) {

```

mm/page.c

Page 4/7

```

199:                                     remove_from_free_list(pg);
200:                                     }
201:                                     pg->count++;
202:                                     return pg;
203:                                     }
204:                                     pg = pg->next_hash;
205:     }
206:
207:     return NULL;
208: }
209:
210: void release_page(int page)
211: {
212:     unsigned long int flags;
213:     struct page *pg;
214:
215:     if(!is_valid_page(page)) {
216:         PANIC("Unexpected inconsistency in hash_table. Missing page %d (
0x%x).\n", page, page);
217:     }
218:
219:     pg = &page_table[page];
220:
221:     if(!pg->count) {
222:         printk("WARNING: %s(): trying to free an already freed page (%d)
!\n", __FUNCTION__, pg->page);
223:         return;
224:     }
225:
226:     if(--pg->count > 0) {
227:         return;
228:     }
229:
230:     SAVE_FLAGS(flags); CLI();
231:
232:     insert_on_free_list(pg);
233:
234:     /* if page is not cached then place it at the head of the free list */
235:     if(!pg->inode) {
236:         page_head = pg;
237:     }
238:
239:     RESTORE_FLAGS(flags);
240:
241:     /*
242:      * We need to wait for free pages to be greater than NR_BUF_RECLAIM,
243:      * otherwise get_free_pages() could run out of pages again, and it
244:      * would think that 'definitely there are no more free pages', killing
245:      * the current process prematurely.
246:      */
247:     if(kstat.free_pages > NR_BUF_RECLAIM) {
248:         wakeup(&get_free_page);
249:     }
250: }
251:
252: int is_valid_page(int page)
253: {
254:     return (page >= 0 && page < NR_PAGES);
255: }
256:
257: void update_page_cache(struct inode *i, __off_t offset, const char *buf, int cou
nt)
258: {
259:     __off_t poffset;
260:     struct page *pg;
261:     int bytes;
262:

```



```

263:     poffset = offset % PAGE_SIZE;
264:     offset &= PAGE_MASK;
265:     bytes = PAGE_SIZE - poffset;
266:
267:     if(count) {
268:         bytes = MIN(bytes, count);
269:         if((pg = search_page_hash(i, offset))) {
270:             page_lock(pg);
271:             memcpy_b(pg->data + poffset, buf, bytes);
272:             page_unlock(pg);
273:             release_page(pg->page);
274:         }
275:     }
276: }
277:
278: int write_page(struct page *pg, struct inode *i, __off_t offset, unsigned int length)
279: {
280:     struct fd fd_table;
281:     unsigned int size;
282:     int errno;
283:
284:     size = MIN(i->i_size, length);
285:     fd_table.inode = i;
286:     fd_table.flags = 0;
287:     fd_table.count = 0;
288:     fd_table.offset = offset;
289:     if(i->fsop && i->fsop->write) {
290:         errno = i->fsop->write(i, &fd_table, pg->data, size);
291:     } else {
292:         errno = -EINVAL;
293:     }
294:
295:     return errno;
296: }
297:
298: int bread_page(struct page *pg, struct inode *i, __off_t offset, char prot, char flags)
299: {
300:     __blk_t block;
301:     __off_t size_read;
302:     int blksize;
303:     struct buffer *buf;
304:
305:     blksize = i->sb->s_blocksize;
306:     size_read = 0;
307:
308:     while(size_read < PAGE_SIZE) {
309:         if((block = bmap(i, offset + size_read, FOR_READING)) < 0) {
310:             return 1;
311:         }
312:         if(block) {
313:             if(!(buf = bread(i->dev, block, blksize))) {
314:                 return 1;
315:             }
316:             memcpy_b(pg->data + size_read, buf->data, blksize);
317:             brelse(buf);
318:         } else {
319:             /* fill the hole with zeros */
320:             memset_b(pg->data + size_read, 0, blksize);
321:         }
322:         size_read += blksize;
323:     }
324:
325:     /* cache any read-only or public (shared) pages */
326:     if(!(prot & PROT_WRITE) || flags & MAP_SHARED) {
327:         pg->inode = i->inode;

```

mm/page.c

Page 6/7

```

328:         pg->offset = offset;
329:         pg->dev = i->dev;
330:         insert_to_hash(pg);
331:     }
332:
333:     return 0;
334: }
335:
336: int file_read(struct inode *i, struct fd *fd_table, char *buffer, __size_t count
)
337: {
338:     __off_t total_read;
339:     unsigned int addr, poffset, bytes;
340:     struct page *pg;
341:
342:     inode_lock(i);
343:
344:     if(fd_table->offset > i->i_size) {
345:         fd_table->offset = i->i_size;
346:     }
347:
348:     total_read = 0;
349:
350:     for(;;) {
351:         count = (fd_table->offset + count > i->i_size) ? i->i_size - fd_
table->offset : count;
352:         if(!count) {
353:             break;
354:         }
355:
356:         poffset = fd_table->offset % PAGE_SIZE;
357:         if(!(pg = search_page_hash(i, fd_table->offset & PAGE_MASK))) {
358:             if(!(addr = kmalloc())) {
359:                 inode_unlock(i);
360:                 printk("%s(): returning -ENOMEM\n", __FUNCTION__
);
361:                 return -ENOMEM;
362:             }
363:             pg = &page_table[V2P(addr) >> PAGE_SHIFT];
364:             if(bread_page(pg, i, fd_table->offset & PAGE_MASK, 0, MA
P_SHARED)) {
365:                 kfree(addr);
366:                 inode_unlock(i);
367:                 printk("%s(): returning -EIO\n", __FUNCTION__);
368:                 return -EIO;
369:             }
370:         } else {
371:             addr = (unsigned int)pg->data;
372:         }
373:
374:         page_lock(pg);
375:         bytes = PAGE_SIZE - poffset;
376:         bytes = MIN(bytes, count);
377:         memcpy_b(buffer + total_read, pg->data + poffset, bytes);
378:         total_read += bytes;
379:         count -= bytes;
380:         fd_table->offset += bytes;
381:         kfree(addr);
382:         page_unlock(pg);
383:     }
384:
385:     inode_unlock(i);
386:     return total_read;
387: }
388:
389: void page_init(int pages)
390: {

```

mm/page.c

Page 7/7

```
391:     struct page *pg;
392:     unsigned int n, addr;
393:
394:     memset_b(page_table, NULL, page_table_size);
395:     memset_b(page_hash_table, NULL, page_hash_table_size);
396:
397:     for(n = 0; n < pages; n++) {
398:         pg = &page_table[n];
399:         pg->page = n;
400:
401:         addr = n << PAGE_SHIFT;
402:         if(addr >= KERNEL_ENTRY_ADDR && addr < V2P(_last_data_addr)) {
403:             pg->flags = PAGE_RESERVED;
404:             kstat.kernel_reserved++;
405:             continue;
406:         }
407:
408:         /*
409:          * Some memory addresses are reserved, like the memory between
410:          * 0xA0000 and 0xFFFFF and other addresses, mostly used by the
411:          * VGA graphics adapter and BIOS.
412:          */
413:         if(!addr_in_bios_map(addr)) {
414:             pg->flags = PAGE_RESERVED;
415:             kstat.physical_reserved++;
416:             continue;
417:         }
418:
419:         pg->data = (char *)P2V(addr);
420:         insert_on_free_list(pg);
421:     }
422:     kstat.total_mem_pages = kstat.free_pages;
423:     kstat.kernel_reserved <<= 2;
424:     kstat.physical_reserved <<= 2;
425: }
```

mm/swapper.c

Page 1/1

```

1: /*
2:  * fiwix/mm/swapper.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/asm.h>
9: #include <fiwix/kernel.h>
10: #include <fiwix/limits.h>
11: #include <fiwix/process.h>
12: #include <fiwix/sleep.h>
13: #include <fiwix/sched.h>
14: #include <fiwix/tty.h>
15: #include <fiwix/memdev.h>
16: #include <fiwix/serial.h>
17: #include <fiwix/lp.h>
18: #include <fiwix/ramdisk.h>
19: #include <fiwix/floppy.h>
20: #include <fiwix/ide.h>
21: #include <fiwix/buffer.h>
22: #include <fiwix/mm.h>
23: #include <fiwix/fs.h>
24: #include <fiwix/locks.h>
25: #include <fiwix/filesystems.h>
26: #include <fiwix/stdio.h>
27:
28: /* kswapd continues the kernel initialization */
29: int kswapd(void)
30: {
31:     STI();
32:
33:     /* char devices */
34:     memdev_init();
35:     serial_init();
36:     lp_init();
37:
38:     /* block devices */
39:     ramdisk_init();
40:     floppy_init();
41:     ide_init();
42:
43:     /* data structures */
44:     sleep_init();
45:     buffer_init();
46:     sched_init();
47:     mount_init();
48:     inode_init();
49:     fd_init();
50:     flock_init();
51:
52:     mem_stats();
53:     fs_init();
54:     mount_root();
55:     init_init();
56:
57:     for(;;) {
58:         sleep(&kswapd, PROC_UNINTERRUPTIBLE);
59:         if(reclaim_buffers()) {
60:             continue;
61:         }
62:         printk("WARNING: %s(): out of memory and swapping is not impleme
nted yet, sorry.\n", __FUNCTION__);
63:         wakeup(&get_free_page);
64:     }
65: }

```

lib/ctype.c

```

1: /*
2:  * fiwix/lib/ctype.c
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/ctype.h>
9:
10: unsigned char _ctype[] = {
11:     0,
12:     _C,          /* ^@  0x00 (NUL '\0') */
13:     _C,          /* ^A  0x01 (SOH) */
14:     _C,          /* ^B  0x02 (STX) */
15:     _C,          /* ^C  0x03 (ETX) */
16:     _C,          /* ^D  0x04 (EOT) */
17:     _C,          /* ^E  0x05 (ENQ) */
18:     _C,          /* ^F  0x06 (ACK) */
19:     _C,          /* ^G  0x07 (BEL '\a') */
20:     _C,          /* ^H  0x08 (BS '\b') */
21:     _C   _S,    /* ^I  0x09 (HT '\t') */
22:     _C   _S,    /* ^J  0x0A (LF '\n') */
23:     _C   _S,    /* ^K  0x0B (VT '\v') */
24:     _C   _S,    /* ^L  0x0C (FF '\f') */
25:     _C   _S,    /* ^M  0x0D (CR '\r') */
26:     _C,          /* ^N  0x0E (SO) */
27:     _C,          /* ^O  0x0F (SI) */
28:     _C,          /* ^P  0x10 (DLE) */
29:     _C,          /* ^Q  0x11 (DC1) */
30:     _C,          /* ^R  0x12 (DC2) */
31:     _C,          /* ^S  0x13 (DC3) */
32:     _C,          /* ^T  0x14 (DC4) */
33:     _C,          /* ^U  0x15 (NAK) */
34:     _C,          /* ^V  0x16 (SYN) */
35:     _C,          /* ^W  0x17 (ETB) */
36:     _C,          /* ^X  0x18 (CAN) */
37:     _C,          /* ^Y  0x19 (EM) */
38:     _C,          /* ^Z  0x1A (SUB) */
39:     _C,          /* ^[  0x1B (ESC) */
40:     _C,          /* ^\  0x1C (FS) */
41:     _C,          /* ^]  0x1D (GS) */
42:     _C,          /* ^^  0x1E (RS) */
43:     _C,          /* ^_  0x1F (US) */
44:     _S,          /* ' '  0x20 */
45:     _P,          /* '! ' 0x21 */
46:     _P,          /* '" ' 0x22 */
47:     _P,          /* '# ' 0x23 */
48:     _P,          /* '$ ' 0x24 */
49:     _P,          /* '% ' 0x25 */
50:     _P,          /* '& ' 0x26 */
51:     _P,          /* '' ' 0x27 */
52:     _P,          /* '(' ' 0x28 */
53:     _P,          /* ')' ' 0x29 */
54:     _P,          /* '* ' 0x2A */
55:     _P,          /* '+' ' 0x2B */
56:     _P,          /* ',' ' 0x2C */
57:     _P,          /* '-' ' 0x2D */
58:     _P,          /* '.' ' 0x2E */
59:     _P,          /* '/' ' 0x2F */
60:     _N,          /* '0'  0x30 */
61:     _N,          /* '1'  0x31 */
62:     _N,          /* '2'  0x32 */
63:     _N,          /* '3'  0x33 */
64:     _N,          /* '4'  0x34 */
65:     _N,          /* '5'  0x35 */
66:     _N,          /* '6'  0x36 */
67:     _N,          /* '7'  0x37 */

```

lib/ctype.c

Page 2/3

```

68:      _N,          /* '8' 0x38 */
69:      _N,          /* '9' 0x39 */
70:      _P,          /* ':' 0x3A */
71:      _P,          /* ';' 0x3B */
72:      _P,          /* '<' 0x3C */
73:      _P,          /* '=' 0x3D */
74:      _P,          /* '>' 0x3E */
75:      _P,          /* '?' 0x3F */
76:      _P,          /* '@' 0x40 */
77:      _U,  _X,     /* 'A' 0x41 */
78:      _U,  _X,     /* 'B' 0x42 */
79:      _U,  _X,     /* 'C' 0x43 */
80:      _U,  _X,     /* 'D' 0x44 */
81:      _U,  _X,     /* 'E' 0x45 */
82:      _U,  _X,     /* 'F' 0x46 */
83:      _U,          /* 'G' 0x47 */
84:      _U,          /* 'H' 0x48 */
85:      _U,          /* 'I' 0x49 */
86:      _U,          /* 'J' 0x4A */
87:      _U,          /* 'K' 0x4B */
88:      _U,          /* 'L' 0x4C */
89:      _U,          /* 'M' 0x4D */
90:      _U,          /* 'N' 0x4E */
91:      _U,          /* 'O' 0x4F */
92:      _U,          /* 'P' 0x50 */
93:      _U,          /* 'Q' 0x51 */
94:      _U,          /* 'R' 0x52 */
95:      _U,          /* 'S' 0x53 */
96:      _U,          /* 'T' 0x54 */
97:      _U,          /* 'U' 0x55 */
98:      _U,          /* 'V' 0x56 */
99:      _U,          /* 'W' 0x57 */
100:     _U,          /* 'X' 0x58 */
101:     _U,          /* 'Y' 0x59 */
102:     _U,          /* 'Z' 0x5A */
103:     _P,          /* '[' 0x5B */
104:     _P,          /* '\' 0x5C */
105:     _P,          /* ']' 0x5D */
106:     _P,          /* '^' 0x5E */
107:     _P,          /* '_' 0x5F */
108:     _P,          /* '`' 0x60 */
109:     _L,  _X,     /* 'a' 0x61 */
110:     _L,  _X,     /* 'b' 0x62 */
111:     _L,  _X,     /* 'c' 0x63 */
112:     _L,  _X,     /* 'd' 0x64 */
113:     _L,  _X,     /* 'e' 0x65 */
114:     _L,  _X,     /* 'f' 0x66 */
115:     _L,          /* 'g' 0x67 */
116:     _L,          /* 'h' 0x68 */
117:     _L,          /* 'i' 0x69 */
118:     _L,          /* 'j' 0x6A */
119:     _L,          /* 'k' 0x6B */
120:     _L,          /* 'l' 0x6C */
121:     _L,          /* 'm' 0x6D */
122:     _L,          /* 'n' 0x6E */
123:     _L,          /* 'o' 0x6F */
124:     _L,          /* 'p' 0x70 */
125:     _L,          /* 'q' 0x71 */
126:     _L,          /* 'r' 0x72 */
127:     _L,          /* 's' 0x73 */
128:     _L,          /* 't' 0x74 */
129:     _L,          /* 'u' 0x75 */
130:     _L,          /* 'v' 0x76 */
131:     _L,          /* 'w' 0x77 */
132:     _L,          /* 'x' 0x78 */
133:     _L,          /* 'y' 0x79 */
134:     _L,          /* 'z' 0x7A */

```

lib/ctype.c

Page 3/3

```
135:         _P,           /* '{' 0x7B */
136:         _P,           /* '|' 0x7C */
137:         _P,           /* '}' 0x7D */
138:         _P,           /* '~' 0x7E */
139:         _C,           /* DEL 0x7F */
140:     };
```

lib/Makefile

Page 1/1

```
1: # fiwix/lib/Makefile
2: #
3: # Copyright 2018, Jordi Sanfeliu. All rights reserved.
4: # Distributed under the terms of the Fiwix License.
5: #
6:
7: .S.o:
8:     $(CC) -traditional -I$(INCLUDE) -c -o $@ $<
9: .c.o:
10:     $(CC) $(CFLAGS) -c -o $@ $<
11:
12: OBJS = ctype.o strings.o printk.o
13:
14: lib:     $(OBJS)
15:           $(LD) $(LDFLAGS) -r $(OBJS) -o lib.o
16:
17: clean:
18:           rm -f *.o
19:
```


lib/printk.c

Page 1/6

```

1: /*
2:  * fiwix/lib/printk.c
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #include <fiwix/kernel.h>
9: #include <fiwix/tty.h>
10: #include <fiwix/stdio.h>
11: #include <fiwix/string.h>
12: #include <fiwix/stdarg.h>
13:
14: #define LOG_BUF_LEN      4096
15: #define MAX_BUF          1024    /* printk() and sprintf() size limit */
16:
17: static char log_buf[LOG_BUF_LEN];
18: static unsigned int log_count;
19:
20: static void puts(char *buffer)
21: {
22:     struct tty *tty;
23:     unsigned short int count;
24:     char *b;
25:
26:     tty = get_tty(_syscondev);
27:     count = strlen(buffer);
28:     b = buffer;
29:
30:     while(count-- > 0) {
31:         if(!tty) {
32:             if(log_count < LOG_BUF_LEN) {
33:                 log_buf[log_count++] = *(b++);
34:             }
35:         } else {
36:             tty_queue_putchar(tty, &tty->write_q, *(b++));
37:
38:             /* kernel messages must be shown immediately */
39:             tty->output(tty);
40:         }
41:     }
42: }
43:
44: /*
45:  * format identifiers
46:  * -----
47:  *      %d      decimal conversion
48:  *      %u      unsigned decimal conversion
49:  *      %x      hexadecimal conversion (lower case)
50:  *      %X      hexadecimal conversion (upper case)
51:  *      %b      binary conversion
52:  *      %o      octal conversion
53:  *      %c      character
54:  *      %s      string
55:  *
56:  * flags
57:  * -----
58:  *      0      result is padded with zeros (e.g.: '%06d')
59:  *              (maximum value is 32)
60:  *      blank  result is padded with spaces (e.g.: '% 6d')
61:  *              (maximum value is 32)
62:  *      -      the numeric result is left-justified
63:  *              (default is right-justified)
64:  */
65: static void do_printk(char *buffer, const char *format, va_list args)
66: {
67:     char sw_neg, in_identifer, n_pad, lf;

```

lib/printk.c

Page 2/6

```

68:     char ch_pad, basecase, c;
69:     char str[] = {
70:         NULL, NULL, NULL, NULL, NULL, NULL, NULL,
71:         NULL, NULL, NULL, NULL, NULL, NULL, NULL,
72:         NULL, NULL, NULL, NULL, NULL, NULL, NULL,
73:         NULL, NULL, NULL, NULL, NULL, NULL, NULL,
74:         NULL
75:     };
76:     char nullstr[7] = { '<', 'N', 'U', 'L', 'L', '>', '\0' };
77:     char *ptr_s, *p;
78:     int num, count;
79:     char simplechar;
80:     unsigned int unum, digit;
81:
82:     sw_neg = in_identifier = n_pad = lf = 0;
83:     count = 0;
84:     basecase = 'A';
85:     ch_pad = ' ';
86:     p = NULL;
87:
88:     /* assumes buffer has a maximum size of MAX_BUF */
89:     while((c = *(format++)) && count < MAX_BUF) {
90:         if((c != '%') && !in_identifier) {
91:             *(buffer++) = c;
92:             memset_b(str, NULL, 32);
93:         } else {
94:             in_identifier = 1;
95:             switch(c = *(format)) {
96:                 case 'd':
97:                     num = va_arg(args, int);
98:                     if(num < 0) {
99:                         num *= -1;
100:                        sw_neg = 1;
101:                    }
102:                    ptr_s = str;
103:                    do {
104:                        *(ptr_s++) = '0' + (num % 10);
105:                    } while(num /= 10);
106:                    if(lf) {
107:                        p = ptr_s;
108:                    } else {
109:                        while(*ptr_s) {
110:                            ptr_s++;
111:                        }
112:                    }
113:                    if(sw_neg) {
114:                        sw_neg = 0;
115:                        *(ptr_s++) = '-';
116:                    }
117:                    do {
118:                        *(buffer++) = *(--ptr_s);
119:                        count++;
120:                    } while(ptr_s != str && count < MAX_BUF)
;
121:                    if(lf) {
122:                        while(*p && count < MAX_BUF) {
123:                            *(buffer++) = *(p++);
124:                            count++;
125:                        }
126:                    }
127:                    format++;
128:                    ch_pad = ' ';
129:                    n_pad = 0;
130:                    in_identifier = 0;
131:                    lf = 0;
132:                    break;
133:

```

lib/printk.c

Page 3/6

```

134:         case 'u':
135:             unum = va_arg(args, unsigned int);
136:             ptr_s = str;
137:             do {
138:                 *(ptr_s++) = '0' + (unum % 10);
139:             } while(unum /= 10);
140:             if(lf) {
141:                 p = ptr_s;
142:             } else {
143:                 while(*ptr_s) {
144:                     ptr_s++;
145:                 }
146:             }
147:             do {
148:                 *(buffer++) = *--ptr_s;
149:                 count++;
150:             } while(ptr_s != str && count < MAX_BUF)
;
151:             if(lf) {
152:                 while(*p && count < MAX_BUF) {
153:                     *(buffer++) = *(p++);
154:                     count++;
155:                 }
156:             }
157:             format++;
158:             ch_pad = ' ';
159:             n_pad = 0;
160:             in_identifier = 0;
161:             lf = 0;
162:             break;
163:
164:         case 'x':
165:             basecase = 'a';
166:         case 'X':
167:             unum = va_arg(args, unsigned int);
168:             ptr_s = str;
169:             do {
170:                 *(ptr_s++) = (digit = (unum & 0x
OF)) > 9 ? basecase + digit - 10 : '0' + digit;
171:             } while(unum /= 16);
172:             if(lf) {
173:                 p = ptr_s;
174:             } else {
175:                 while(*ptr_s) {
176:                     ptr_s++;
177:                 }
178:             }
179:             do {
180:                 *(buffer++) = *--ptr_s;
181:                 count++;
182:             } while(ptr_s != str && count < MAX_BUF)
;
183:             if(lf) {
184:                 while(*p && count < MAX_BUF) {
185:                     *(buffer++) = *(p++);
186:                     count++;
187:                 }
188:             }
189:             format++;
190:             ch_pad = ' ';
191:             n_pad = 0;
192:             in_identifier = 0;
193:             lf = 0;
194:             break;
195:
196:         case 'b':
197:             num = va_arg(args, unsigned int);

```

lib/printk.c

Page 4/6

```

198:         if(num < 0) {
199:             num *= -1;
200:         }
201:         ptr_s = str;
202:         do {
203:             *(ptr_s++) = '0' + (num % 2);
204:         } while(num /= 2);
205:         if(lf) {
206:             p = ptr_s;
207:         } else {
208:             while(*ptr_s) {
209:                 ptr_s++;
210:             }
211:         }
212:         do {
213:             *(buffer++) = *(--ptr_s);
214:             count++;
215:         } while(ptr_s != str && count < MAX_BUF)
;
216:         if(lf) {
217:             while(*p && count < MAX_BUF) {
218:                 *(buffer++) = *(p++);
219:                 count++;
220:             }
221:         }
222:         format++;
223:         ch_pad = ' ';
224:         n_pad = 0;
225:         in_identifier = 0;
226:         lf = 0;
227:         break;
228:
229:     case 'o':
230:         num = va_arg(args, unsigned int);
231:         if(num < 0) {
232:             num *= -1;
233:         }
234:         ptr_s = str;
235:         do {
236:             *(ptr_s++) = '0' + (num % 8);
237:         } while(num /= 8);
238:         if(lf) {
239:             p = ptr_s;
240:         } else {
241:             while(*ptr_s) {
242:                 ptr_s++;
243:             }
244:         }
245:         do {
246:             *(buffer++) = *(--ptr_s);
247:             count++;
248:         } while(ptr_s != str && count < MAX_BUF)
;
249:         if(lf) {
250:             while(*p && count < MAX_BUF) {
251:                 *(buffer++) = *(p++);
252:                 count++;
253:             }
254:         }
255:         format++;
256:         ch_pad = ' ';
257:         n_pad = 0;
258:         in_identifier = 0;
259:         lf = 0;
260:         break;
261:
262:     case 'c':

```

lib/printk.c

Page 5/6

```

263:         simplechar = va_arg(args, int);
264:         *(buffer++) = simplechar;
265:         format++;
266:         in_identifier = 0;
267:         lf = 0;
268:         break;
269:
270:     case 's':
271:         num = 0;
272:         ptr_s = va_arg(args, char *);
273:         if(n_pad) {
274:             num = n_pad - strlen(ptr_s);
275:             if(num < 0) {
276:                 num *= -1;
277:             }
278:         }
279:         /* if it's a NULL then show "<NULL>" */
280:         if(ptr_s == NULL) {
281:             ptr_s = (char *)nullstr;
282:         }
283:         while((c = *(ptr_s++)) && count < MAX_BU
F) {
284:             *(buffer++) = c;
285:             count++;
286:         }
287:         while(num-- && count < MAX_BUF) {
288:             *(buffer++) = ' ';
289:             count++;
290:         }
291:         format++;
292:         n_pad = 0;
293:         in_identifier = 0;
294:         lf = 0;
295:         break;
296:
297:     case ' ':
298:         ch_pad = ' ';
299:         break;
300:
301:     case '0':
302:         if(!n_pad) {
303:             ch_pad = '0';
304:         }
305:     case '1':
306:     case '2':
307:     case '3':
308:     case '4':
309:     case '5':
310:     case '6':
311:     case '7':
312:     case '8':
313:     case '9':
314:         n_pad = !n_pad ? c - '0' : ((n_pad * 10)
+ (c - '0'));
315:         n_pad = n_pad > 32 ? 32 : n_pad;
316:         for(unum = 0; unum < n_pad; unum++) {
317:             str[unum] = ch_pad;
318:         }
319:         break;
320:
321:     case '-':
322:         lf = 1;
323:         break;
324:     case '%':
325:         *(buffer++) = c;
326:         format++;
327:         in_identifier = 0;

```

lib/printk.c

Page 6/6

```
328:                                     break;
329:                                     }
330:                                 }
331:                                 count++;
332:                             }
333:                             *buffer = NULL;
334:     }
335:
336: void register_console(void (*fn)(char *, unsigned int))
337: {
338:     (*fn)(log_buf, log_count);
339: }
340:
341: void printk(const char *format, ...)
342: {
343:     va_list args;
344:     char buffer[MAX_BUF];
345:
346:     va_start(args, format);
347:     do_printk(buffer, format, args);
348:     puts(buffer);
349:     va_end(args);
350: }
351:
352: int sprintk(char *buffer, const char *format, ...)
353: {
354:     va_list args;
355:
356:     va_start(args, format);
357:     do_printk(buffer, format, args);
358:     va_end(args);
359:     return strlen(buffer);
360: }
```

lib/strings.c

Page 1/5

```

1:  /*
2:  *  fiwix/lib/strings.c
3:  *
4:  *  Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  *  Distributed under the terms of the Fiwix License.
6:  */
7:
8:  #include <fiwix/types.h>
9:  #include <fiwix/tty.h>
10: #include <fiwix/mm.h>
11: #include <fiwix/stdio.h>
12: #include <fiwix/string.h>
13:
14: /* convert from big-endian to little-endian (word swap) */
15: void swap_asc_word(char *str, int len)
16: {
17:     int n, n2;
18:     short int *ptr;
19:     char *buf;
20:
21:     if(!(buf = (void *)kmalloc())) {
22:         return;
23:     }
24:
25:     ptr = (short int *)str;
26:
27:     for(n = 0, n2 = 0; n < len; n++) {
28:         buf[n2++] = *ptr >> 8;
29:         buf[n2++] = *ptr & 0xFF;
30:         ptr++;
31:     }
32:     for(n = len - 1; n > 0; n--) {
33:         if(buf[n] == NULL || buf[n] == ' ') {
34:             buf[n] = NULL;
35:         } else {
36:             break;
37:         }
38:     }
39:     memcpy_b(str, buf, len);
40:     kfree((unsigned int)buf);
41: }
42:
43: int strcmp(const char *str1, const char *str2)
44: {
45:     while(*str1) {
46:         if(*str1 != *str2) {
47:             return 1;
48:         }
49:         str1++;
50:         str2++;
51:     }
52:     if(!(*str2)) {
53:         return 0;
54:     }
55:     return 1;
56: }
57:
58: int strncmp(const char *str1, const char *str2, __ssize_t n)
59: {
60:     while(n > 0) {
61:         if(*str1 != *str2) {
62:             return 1;
63:         }
64:         str1++;
65:         str2++;
66:         n--;
67:     }

```

lib/strings.c

Page 2/5

```
68:         return 0;
69:     }
70:
71:     char *strcpy(char *dest, const char *src)
72:     {
73:         if(!dest || !src) {
74:             return NULL;
75:         }
76:
77:         while(*src) {
78:             *dest = *src;
79:             dest++;
80:             src++;
81:         }
82:         *dest = NULL;          /* NULL-terminated */
83:         return dest;
84:     }
85:
86:     void strncpy(char *dest, const char *src, int len)
87:     {
88:         if(!dest || !src) {
89:             return;
90:         }
91:
92:         while((*src) && len) {
93:             *dest = *src;
94:             dest++;
95:             src++;
96:             len--;
97:         }
98:         *dest = NULL;        /* NULL-terminated */
99:     }
100:
101:     char *strcat(char *dest, const char *src)
102:     {
103:         char *orig;
104:
105:         orig = dest;
106:         while(*dest) {
107:             dest++;
108:         }
109:         while(*src) {
110:             *dest = *src;
111:             dest++;
112:             src++;
113:         }
114:         *dest = NULL;
115:         return orig;
116:     }
117:
118:     char *strncat(char *dest, const char *src, __ssize_t len)
119:     {
120:         char *orig;
121:
122:         orig = dest;
123:         while(*dest) {
124:             dest++;
125:         }
126:         while(*src && len) {
127:             *dest = *src;
128:             dest++;
129:             src++;
130:             len--;
131:         }
132:         *dest = NULL;
133:         return orig;
134:     }
```


lib/strings.c

Page 3/5

```
135:
136: int strlen(const char *str)
137: {
138:     int n;
139:
140:     n = 0;
141:     while(str && *str) {
142:         n++;
143:         str++;
144:     }
145:     return n;
146: }
147:
148: char * get_basename(const char *path)
149: {
150:     char *basename;
151:     char c;
152:
153:     basename = NULL;
154:
155:     while(path) {
156:         while(*path == '/') {
157:             path++;
158:         }
159:         if(*path != NULL) {
160:             basename = (char *)path;
161:         }
162:         while((c = *(path++)) && (c != '/'));
163:         if(!c) {
164:             break;
165:         }
166:     }
167:     return basename;
168: }
169:
170: char * remove_trailing_slash(char *path)
171: {
172:     char *p;
173:
174:     p = path + (strlen(path) - 1);
175:     while(p > path && *p == '/') {
176:         *p = NULL;
177:         p--;
178:     }
179:     return path;
180: }
181:
182: int is_dir(const char *path)
183: {
184:     while*(path + 1) {
185:         path++;
186:     }
187:     if(*path == '/') {
188:         return 1;
189:     }
190:     return 0;
191: }
192:
193: int atoi(const char *str)
194: {
195:     int n;
196:
197:     n = 0;
198:     while(IS_SPACE(*str)) {
199:         str++;
200:     }
201:     while(IS_NUMERIC(*str)) {
```

lib/strings.c

Page 4/5

```
202:             n = (n * 10) + (*str++ - '0');
203:         }
204:         return n;
205:     }
206:
207: void memcpy_b(void *dest, const void *src, unsigned int count)
208: {
209:     unsigned char *d;
210:     unsigned char *s;
211:
212:     d = (unsigned char *)dest;
213:     s = (unsigned char *)src;
214:     while(count--) {
215:         *d = *s;
216:         d++;
217:         s++;
218:     }
219: }
220:
221: void memcpy_w(void *dest, const void *src, unsigned int count)
222: {
223:     unsigned short int *d;
224:     unsigned short int *s;
225:
226:     d = (unsigned short int *)dest;
227:     s = (unsigned short int *)src;
228:     while(count--) {
229:         *d = *s;
230:         d++;
231:         s++;
232:     }
233: }
234:
235: void memcpy_l(void *dest, const void *src, unsigned int count)
236: {
237:     unsigned int *d;
238:     unsigned int *s;
239:
240:     d = (unsigned int *)dest;
241:     s = (unsigned int *)src;
242:     while(count--) {
243:         *d = *s;
244:         d++;
245:         s++;
246:     }
247: }
248:
249: void memset_b(void *dest, unsigned char value, unsigned int count)
250: {
251:     unsigned char *d;
252:
253:     d = (unsigned char *)dest;
254:     while(count--) {
255:         *d = value;
256:         d++;
257:     }
258: }
259:
260: void memset_w(void *dest, unsigned short int value, unsigned int count)
261: {
262:     unsigned short int *d;
263:
264:     d = (unsigned short int *)dest;
265:     while(count--) {
266:         *d = value;
267:         d++;
268:     }
}
```

lib/strings.c

Page 5/5

```
269: }
270:
271: void memset_l(void *dest, unsigned int value, unsigned int count)
272: {
273:     unsigned int *d;
274:
275:     d = (unsigned int *)dest;
276:     while(count-- > 0) {
277:         *d = value;
278:         d++;
279:     }
280: }
```

include/fiwix/asm.h

Page 1/3

```

1: /*
2:  * fiwix/include/fiwix/asm.h
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_ASM_H
9: #define _FIWIX_ASM_H
10:
11: extern void except0(void);
12: extern void except1(void);
13: extern void except2(void);
14: extern void except3(void);
15: extern void except4(void);
16: extern void except5(void);
17: extern void except6(void);
18: extern void except7(void);
19: extern void except8(void);
20: extern void except9(void);
21: extern void except10(void);
22: extern void except11(void);
23: extern void except12(void);
24: extern void except13(void);
25: extern void except14(void);
26: extern void except15(void);
27: extern void except16(void);
28: extern void except17(void);
29: extern void except18(void);
30: extern void except19(void);
31: extern void except20(void);
32: extern void except21(void);
33: extern void except22(void);
34: extern void except23(void);
35: extern void except24(void);
36: extern void except25(void);
37: extern void except26(void);
38: extern void except27(void);
39: extern void except28(void);
40: extern void except29(void);
41: extern void except30(void);
42: extern void except31(void);
43:
44: extern void irq0(void);
45: extern void irq1(void);
46: extern void irq2(void);
47: extern void irq3(void);
48: extern void irq4(void);
49: extern void irq5(void);
50: extern void irq6(void);
51: extern void irq7(void);
52: extern void irq8(void);
53: extern void irq9(void);
54: extern void irq10(void);
55: extern void irq11(void);
56: extern void irq12(void);
57: extern void irq13(void);
58: extern void irq14(void);
59: extern void irq15(void);
60: extern void unknown_irq(void);
61:
62: extern void switch_to_user_mode(void);
63: extern void sighandler_trampoline(void);
64: extern void end_sighandler_trampoline(void);
65: extern void syscall(void);
66: extern void return_from_syscall(void);
67: extern void do_switch(unsigned int *, unsigned int *, unsigned int, unsigned int

```

include/fiwix/asm.h

Page 2/3

```

, unsigned int, unsigned short int);
68:
69: int cpuid(void);
70: int getfpu(void);
71: int vendor_id(void);
72: int signature_flags(void);
73: int brand_str(void);
74: int tlbinfo(void);
75:
76: unsigned char inport_b(unsigned int);
77: short int inport_w(unsigned int);
78: void inport_sw(unsigned int, void *, unsigned int);
79: void outport_b(unsigned int, unsigned char);
80: void outport_w(unsigned int, unsigned int);
81: void outport_sw(unsigned int, void *, unsigned int);
82:
83: void load_gdt(unsigned int);
84: void load_idt(unsigned int);
85: void activate_kpage_dir(void);
86: void load_tr(unsigned int);
87: unsigned long long int get_rdtsc(void);
88: void invalidate_tlb(void);
89:
90: #define CLI() __asm__ __volatile__ ("cli"::"memory")
91: #define STI() __asm__ __volatile__ ("sti"::"memory")
92: #define NOP() __asm__ __volatile__ ("nop"::"memory")
93: #define HLT() __asm__ __volatile__ ("hlt"::"memory")
94:
95: #define GET_CR2(cr2) __asm__ __volatile__ ("movl %%cr2, %0" : "=r" (cr2));
96: #define GET_ESP(esp) __asm__ __volatile__ ("movl %%esp, %0" : "=r" (esp));
97: #define SET_ESP(esp) __asm__ __volatile__ ("movl %0, %%esp" : "=r" (esp));
98:
99: #define SAVE_FLAGS(flags) \
100:     __asm__ __volatile__( \
101:         "pushfl ; popl %0\n\t" \
102:         : "=r" (flags) \
103:         : /* no input */ \
104:         : "memory" \
105:     );
106:
107: #define RESTORE_FLAGS(x) \
108:     __asm__ __volatile__( \
109:         "pushl %0 ; popfl\n\t" \
110:         : /* no output */ \
111:         : "r" (flags) \
112:         : "memory" \
113:     );
114:
115: #define USER_SYSCALL(num, arg1, arg2, arg3) \
116:     __asm__ __volatile__( \
117:         "movl %0, %%eax\n\t" \
118:         "movl %1, %%ebx\n\t" \
119:         "movl %2, %%ecx\n\t" \
120:         "movl %3, %%edx\n\t" \
121:         "int $0x80\n\t" \
122:         : /* no output */ \
123:         : "eax"((unsigned int)num), "ebx"((unsigned int)arg1), "ecx"((un- \
signed int)arg2), "edx"((unsigned int)arg3) \
124:     );
125:
126: /*
127: static inline unsigned long long int get_rdtsc(void)
128: {
129:     unsigned int eax, edx;
130:
131:     __asm__ __volatile__("rdtsc" : "=a" (eax), "=d" (edx));
132:     return ((unsigned long long int)eax) | (((unsigned long long int)edx) <<

```

include/fiwix/asm.h

Page 3/3

```
32);  
133: }  
134: */  
135:  
136: #endif /* _FIWIX_ASM_H */
```

include/fiwix/bios.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/bios.h
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_BIOS_H
9: #define _FIWIX_BIOS_H
10:
11: #include <fiwix/multiboot1.h>
12:
13: #define NR_BIOS_MM_ENT      50      /* entries in BIOS memory map */
14:
15: struct bios_mem_map {
16:     unsigned long int from;
17:     unsigned long int to;
18:     int type;
19: };
20: struct bios_mem_map bios_mem_map[NR_BIOS_MM_ENT];
21:
22: int addr_in_bios_map(unsigned int);
23: void bios_map_add(unsigned long int, unsigned long int, int, int);
24: void bios_map_init(struct multiboot_mmap_entry *, unsigned long int);
25:
26: #endif /* _FIWIX_BIOS_H */
```

include/fiwix/buffer.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/buffer.h
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_BUFFER_H
9: #define _FIWIX_BUFFER_H
10:
11: #include <fiwix/types.h>
12: #include <fiwix/fs.h>
13:
14: /* buffer flags */
15: #define BUFFER_VALID    0x0001
16: #define BUFFER_LOCKED  0x0002
17: #define BUFFER_DIRTY   0x0004
18:
19: struct buffer {
20:     __dev_t dev;                /* device number */
21:     __blk_t block;             /* block number */
22:     int size;                   /* block size (in bytes) */
23:     int flags;
24:     char *data;                 /* block contents */
25:     struct buffer *prev_hash;
26:     struct buffer *next_hash;
27:     struct buffer *prev_free;
28:     struct buffer *next_free;
29:     struct buffer *prev_dirty;
30:     struct buffer *next_dirty;
31: };
32: extern struct buffer *buffer_table;
33: extern struct buffer **buffer_hash_table;
34:
35: /* values to be determined during system startup */
36: extern unsigned int buffer_table_size;        /* size in bytes */
37: extern unsigned int buffer_hash_table_size;   /* size in bytes */
38:
39: struct buffer * bread(__dev_t, __blk_t, int);
40: void bwrite(struct buffer *);
41: void brelse(struct buffer *);
42: void sync_buffers(__dev_t);
43: void invalidate_buffers(__dev_t);
44: int reclaim_buffers(void);
45: void buffer_init(void);
46:
47: #endif /* _FIWIX_BUFFER_H */
```


include/fiwix/cmos.h

Page 1/1

```

1: /*
2:  * fiwix/include/fiwix/cmos.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_CMOS_H
9: #define _FIWIX_CMOS_H
10:
11: #define CMOS_INDEX      0x70
12: #define CMOS_DATA      0x71
13:
14: #define CMOS_STATA_IRQF 0x0F    /* periodic interrupt frequency */
15: #define CMOS_STATA_UIP  0x80    /* time update in progress */
16:
17: #define CMOS_STATB_DSE  0x01    /* enable daylight savings */
18: #define CMOS_STATB_24H  0x02    /* 24-hour mode (0=12h, 1=24h) */
19: #define CMOS_STATB_DM   0x04    /* time/date in binary mode (0=BCD, 1=binary) */
20: #define CMOS_STATB_SQWE 0x08    /* enable square wave frequency */
21: #define CMOS_STATB_UIE  0x10    /* enable update-ended interrupt */
22: #define CMOS_STATB_AIE  0x20    /* enable alarm interrupt */
23: #define CMOS_STATB_PIE  0x40    /* enable periodic interrupt */
24: #define CMOS_STATB_SET  0x80    /* abort clock update */
25:
26: #define CMOS_STATD_VRT  0x80    /* valid RAM and time */
27:
28: /* CMOS RAM data registers */
29: #define CMOS_SEC        0x00    /* second */
30: #define CMOS_ASEC      0x01    /* alarm second */
31: #define CMOS_MIN       0x02    /* minute */
32: #define CMOS_AMIN     0x03    /* alarm minute */
33: #define CMOS_HOUR      0x04    /* hour */
34: #define CMOS_AHOUR    0x05    /* alarm hour */
35: #define CMOS_DOW       0x06    /* day of week */
36: #define CMOS_DAY       0x07    /* day */
37: #define CMOS_MONTH     0x08    /* month */
38: #define CMOS_YEAR      0x09    /* last two digits of year */
39: #define CMOS_STATA     0x0A    /* status register A */
40: #define CMOS_STATB     0x0B    /* status register B */
41: #define CMOS_STATC     0x0C    /* status register C */
42: #define CMOS_STATD     0x0D    /* status register D */
43: #define CMOS_DIAG     0x0E    /* diagnostics status */
44: #define CMOS_FDDTYPE   0x10    /* floppy disk drive type */
45: #define CMOS_HDDTYPE   0x12    /* hard disk drive type */
46: #define CMOS_CENTURY   0x32    /* century */
47:
48: /* conversions */
49: #define BCD2BIN(bcd)    (((bcd) >> 4) * 10) + ((bcd) & 0x0F)
50: #define BIN2BCD(bin)    ((bin) % 10) | (((bin) / 10) << 4)
51:
52: int cmos_update_in_progress(void);
53: unsigned char cmos_read_date(unsigned char);
54: void cmos_write_date(unsigned char, unsigned char);
55: unsigned char cmos_read(unsigned char);
56: void cmos_write(unsigned char, unsigned char);
57:
58: #endif /* _FIWIX_CMOS_H */

```

include/fiwix/config.h

Page 1/1

```

1: /*
2:  * fiwix/include/fiwix/config.h
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_CONFIG_H
9: #define _FIWIX_CONFIG_H
10:
11: /* maximum number of processes */
12: #define NR_PROCS          64
13:
14: /* maximum number of callout functions (timer) */
15: #define NR_CALLOUTS      NR_PROCS
16:
17: /* maximum number of mounted filesystems */
18: #define NR_MOUNT_POINTS  8
19:
20: /* maximum number of opened files in system */
21: #define NR_OPENS         1024
22:
23: /* maximum number of flocks in system */
24: #define NR_FLOCKS       (NR_PROCS * 5)
25:
26:
27:
28: /* percentage of memory that buffer cache will borrow from available memory */
29: #define BUFFER_PERCENTAGE 100
30:
31: /* percentage of hash buckets relative to the size of the buffer table */
32: #define BUFFER_HASH_PERCENTAGE 10
33:
34: /* buffers reclaimed in a single shot */
35: #define NR_BUF_RECLAIM   150
36:
37:
38: /* percentage of memory assigned to the inode table and hash table */
39: #define INODE_PERCENTAGE 1
40:
41: /* percentage of hash buckets relative to the size of the inode table */
42: #define INODE_HASH_PERCENTAGE 10
43:
44:
45: /* maximum value for PID */
46: #define MAX_PID_VALUE    32767
47:
48: /* number of screens in console' scroll back */
49: #define SCREENS_LOG      6
50:
51: /* maximum number of messages on spurious interrupts */
52: #define MAX_SPU_NOTICES  10
53:
54: /* maximum number of virtual memory mappings (regions) */
55: #define VMA_REGIONS     150
56:
57:
58:
59: /* #define CONFIG_VERBOSE_SEGFAULTS */
60:
61: #endif /* _FIWIX_CONFIG_H */

```

include/fiwix/console.h

Page 1/3

```

1: /*
2:  * fiwix/include/fiwix/console.h
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_CONSOLE_H
9: #define _FIWIX_CONSOLE_H
10:
11: #include <fiwix/vt.h>
12:
13: #define NR_VCONSOLES          12      /* number of virtual consoles */
14:
15: #define VCONSOLES_MAJOR      4        /* virtual consoles major number */
16: #define SYSCON_MAJOR        5        /* system console major number */
17:
18: /* Graphic Rendition Combination Modes */
19: #define SGR_DEFAULT          0        /* back to the default rendition */
20: #define SGR_BOLD             1        /* set bold */
21: #define SGR_BLINK           5        /* set slowly blinking */
22: #define SGR_REVERSE         7        /* set reverse video */
23: #define SGR_BOLD_OFF        21       /* unset bold */
24: #define SGR_BLINK_OFF       25       /* unset blinking */
25: #define SGR_REVERSE_OFF     27       /* unset reverse video */
26:
27: #define SGR_BLACK_FG         30       /* set black foreground */
28: #define SGR_RED_FG          31       /* set red foreground */
29: #define SGR_GREEN_FG        32       /* set green foreground */
30: #define SGR_BROWN_FG        33       /* set brown foreground */
31: #define SGR_BLUE_FG         34       /* set blue foreground */
32: #define SGR_MAGENTA_FG      35       /* set magenta foreground */
33: #define SGR_CYAN_FG         36       /* set cyan foreground */
34: #define SGR_WHITE_FG        37       /* set white foreground */
35:
36: #define SGR_DEFAULT_FG_U_ON  38       /* set def. fg color (underline on) */
37: #define SGR_DEFAULT_FG_U_OFF 39       /* set def. fg color (underline off) */
38:
39: #define SGR_BLACK_BG         40       /* set black background */
40: #define SGR_RED_BG          41       /* set red background */
41: #define SGR_GREEN_BG        42       /* set green background */
42: #define SGR_BROWN_BG        43       /* set brown background */
43: #define SGR_BLUE_BG         44       /* set blue background */
44: #define SGR_MAGENTA_BG      45       /* set magenta background */
45: #define SGR_CYAN_BG         46       /* set cyan background */
46: #define SGR_WHITE_BG        47       /* set white background */
47:
48: #define SGR_DEFAULT_BG      49       /* set default background color */
49:
50: #define NPARMS                16
51: #define BLANK_INTERVAL      (0 * HZ) /* 0 seconds (no screen blank) */
52:
53: #define COLOR_BLACK          0x0000
54: #define COLOR_BLUE           0x0100
55: #define COLOR_GREEN          0x0200
56: #define COLOR_CYAN           0x0300
57: #define COLOR_RED            0x0400
58: #define COLOR_MAGENTA        0x0500
59: #define COLOR_BROWN         0x0600
60: #define COLOR_WHITE          0x0700
61: #define BG_BLACK             0x0000
62: #define BG_BLUE              0x1000
63: #define BG_GREEN             0x2000
64: #define BG_CYAN              0x3000
65: #define BG_RED               0x4000
66: #define BG_MAGENTA           0x5000
67: #define BG_BROWN             0x6000

```

include/fiwix/console.h

Page 2/3

```

68: #define BG_WHITE          0x7000
69:
70: #define DEF_MODE          (COLOR_WHITE | BG_BLACK)
71: #define BLANK_MEM        (DEF_MODE | ' ')
72:
73: #define SCREEN_COLS      video.columns
74: #define SCREEN_LINES     video.lines
75: #define SCREEN_SIZE      (video.columns * video.lines)
76: #define VC_BUF_LINES     (video.lines * SCREENS_LOG)
77: #define VC_BUF_SIZE      (video.columns * VC_BUF_LINES)
78:
79: #define SCROLL_UP        1
80: #define SCROLL_DOWN     2
81:
82: #define BS                127      /* backspace */
83:
84: #define VPF_VGA           0x01     /* VGA text mode */
85: #define VPF_VESAFB       0x02     /* x86 frame buffer */
86: #define VPF_CURSOR_ON    0x04     /* draw cursor */
87:
88: #define ON                1
89: #define OFF               0
90: #define COND              2
91:
92: /* console flags */
93: #define CONSOLE_HAS_FOCUS 0x0001
94: #define CONSOLE_BLANKED  0x0002
95:
96:
97: extern short int current_cons; /* current console (/dev/tty1 ... /dev/tty12) */
98:
99: short int *vc_screen[NR_VCONSOLES + 1];
100:
101: /*
102:  * This is the scrollbar history buffer which is used only in the active
103:  * vconsole. Everytime a vconsole is switched, the screen contents of the
104:  * new vconsole is copied back to this buffer. Only the visible screen is
105:  * copied, so switching vconsoles means losing the scrollbar history.
106:  */
107: short int *vcbuf;
108:
109: struct vconsole {
110:     int x;          /* current column */
111:     int y;          /* current line */
112:     int top, lines, columns;
113:     short int check_x;
114:     unsigned char led_status;
115:     unsigned char scrlock, numlock, capslock;
116:     unsigned char esc, sbracket, semicolon, question;
117:     int flags;
118:     int parmvl, parmvl2;
119:     int nparms, parms[NPARMS];
120:     unsigned short int color_attr;
121:     unsigned char bold, underline, blink, reverse;
122:     int insert_mode;
123:     unsigned char *vidmem; /* write here only when console has focus */
124:     short int *screen;     /* the back-buffer of the screen */
125:     int saved_x, cursor_x;
126:     int saved_y, cursor_y;
127:     struct vt_mode vt_mode;
128:     unsigned char vc_mode;
129:     int switcho_tty;
130:     struct tty *tty;
131: };
132:
133: struct video_parms {
134:     int flags;

```

include/fiwix/console.h

Page 3/3

```

135:     unsigned int *address;
136:     int port;
137:     int memsize;
138:     unsigned char signature[32];
139:     int columns;
140:     int lines;
141:     int buf_y;
142:     int buf_top;
143:     int fb_version;
144:     int fb_width;
145:     int fb_height;
146:     int fb_char_width;
147:     int fb_char_height;
148:     int fb_bpp;
149:     int fb_pixelwidth;
150:     int fb_pitch;
151:     int fb_linesize;
152:     int fb_size;      /* size of screen based on resolution */
153:     int fb_vsize;    /* size of screen based on columns x lines */
154:
155:     /* formerly video driver operations */
156:     void (*put_char)(struct vconsole *, unsigned char);
157:     void (*insert_char)(struct vconsole *);
158:     void (*delete_char)(struct vconsole *);
159:     void (*update_curpos)(struct vconsole *);
160:     void (*show_cursor)(struct vconsole *, int);
161:     void (*get_curpos)(struct vconsole *);
162:     void (*write_screen)(struct vconsole *, int, int, short int);
163:     void (*blank_screen)(struct vconsole *);
164:     void (*scroll_screen)(struct vconsole *, int, int);
165:     void (*restore_screen)(struct vconsole *);
166:     void (*screen_on)(struct vconsole *);
167:     void (*buf_scroll)(struct vconsole *, int);
168:     void (*cursor_blink)(unsigned int);
169: };
170: extern struct video_parms video;
171:
172: void vconsole_reset(struct tty *);
173: void vconsole_write(struct tty *);
174: void vconsole_select(int);
175: void vconsole_select_final(int);
176: void vconsole_restore(struct vconsole *);
177: void unblank_screen(struct vconsole *);
178: void vconsole_start(struct tty *);
179: void vconsole_stop(struct tty *);
180: void vconsole_beep(void);
181: void vconsole_deltab(struct tty *);
182: void console_flush_log_buf(char *, unsigned int);
183: void console_init(void);
184:
185: #endif /* _FIWIX_CONSOLE_H */

```

include/fiwix/cpu.h

Page 1/2

```

1: /*
2:  * fiwix/include/fiwix/cpu.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_CPU_H
9: #define _FIWIX_CPU_H
10:
11: #define CPU_FPU          0x00000001    /* Floating-Point Unit on chip */
12: #define CPU_VME          0x00000002    /* Virtual 8086 Mode Enhancements */
13: #define CPU_DE           0x00000004    /* Debugging Extensions */
14: #define CPU_PSE          0x00000008    /* Page Size Extension */
15: #define CPU_TSC          0x00000010    /* Time Stamp Counter */
16: #define CPU_MSR          0x00000020    /* Model Specific Registers */
17: #define CPU_PAE          0x00000040    /* Physical Address Extension */
18: #define CPU_MCE          0x00000080    /* Machine Check Exception */
19: #define CPU_CX8          0x00000100    /* CMPXCHG8B instruction supported */
20: #define CPU_APIC         0x00000200    /* On-chip APIC hardware supported */
21: #define CPU_RES10       0x00000400    /* Reserved */
22: #define CPU_SEP          0x00000800    /* Fast System Call */
23: #define CPU_MTRR         0x00001000    /* Memory Type Range Registers */
24: #define CPU_PGE          0x00002000    /* Page Global Enable */
25: #define CPU_MCA          0x00004000    /* Machine Check Architecture */
26: #define CPU_CMOV         0x00008000    /* Conditional Move Instruction */
27: #define CPU_PAT          0x00010000    /* Page Attribute Table */
28: #define CPU_PSE36        0x00020000    /* 36-bit Page Size Extension */
29: #define CPU_PSN          0x00040000    /* Processor Serial Number */
30: #define CPU_CLFSH        0x00080000    /* CLFLUSH instruction supported */
31: #define CPU_RES20       0x00100000    /* Reserved */
32: #define CPU_DS           0x00200000    /* Debug Store */
33: #define CPU_ACPI         0x00400000    /* Thermal Monitor and others */
34: #define CPU_MMX          0x00800000    /* Intel Architecture MMX Technology */
35: #define CPU_FXSR         0x01000000    /* Fast Floating Point Save and Rest. */
36: #define CPU_SSE          0x02000000    /* Streaming SIMD Extensions */
37: #define CPU_SSE2         0x04000000    /* Streaming SIMD Extensions 2 */
38: #define CPU_SS           0x08000000    /* Self-Snoop */
39: #define CPU_HTT          0x10000000    /* Hyper-Threading Technology */
40: #define CPU_TM           0x20000000    /* Thermal Monitor */
41: #define CPU_RES30       0x40000000    /* Reserved */
42: #define CPU_PBE          0x80000000    /* Pending Break Enable */
43:
44: #define RESERVED_DESC    0x80000000    /* TLB descriptor reserved */
45:
46: struct cpu {
47:     char *vendor_id;
48:     char family;
49:     char model;
50:     char *model_name;
51:     char stepping;
52:     unsigned long int hz;
53:     char *cache;
54:     char has_cpuid;
55:     char has_fpu;
56:     int flags;
57: };
58: struct cpu cpu_table;
59:
60: struct cpu_type {
61:     int cpu;
62:     char *name[20];
63: };
64:
65: int get_cpu_flags(char *, int);
66: void cpu_init(void);
67:

```

include/fiwix/cpu.h

Page 2/2

```
68: #endif /* _FIWIX_CPU_H */
```

include/fiwix/ctype.h

Page 1/1

```

1:  /*
2:  *  fiwix/include/fiwix/ctype.h
3:  *
4:  *  Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  *  Distributed under the terms of the Fiwix License.
6:  */
7:
8:  #ifndef _FIWIX_CTYPE_H
9:  #define _FIWIX_CTYPE_H
10:
11: #define _U      0x01    /* upper case */
12: #define _L      0x02    /* lower case */
13: #define _N      0x04    /* numeral (digit) */
14: #define _S      0x08    /* spacing character */
15: #define _P      0x10    /* punctuation */
16: #define _C      0x20    /* control character */
17: #define _X      0x40    /* hexadecimal */
18: #define _B      0x80    /* blank */
19:
20: extern unsigned char _ctype[];
21:
22: #define ISALPHA(ch)      ((_ctype + 1)[ch] & (_U | _L))
23: #define ISUPPER(ch)     ((_ctype + 1)[ch] & _U)
24: #define ISLOWER(ch)     ((_ctype + 1)[ch] & _L)
25: #define ISDIGIT(ch)     ((_ctype + 1)[ch] & _N)
26: #define ISALNUM(ch)     ((_ctype + 1)[ch] & (_U | _L | _N))
27: #define ISSPACE(ch)     ((_ctype + 1)[ch] & _S)
28: #define ISPUNCT(ch)     ((_ctype + 1)[ch] & _P)
29: #define ISCNTRL(ch)     ((_ctype + 1)[ch] & _C)
30: #define ISXDIGIT(ch)    ((_ctype + 1)[ch] & (_N | _X))
31:
32: #define ISASCII(ch)     ((unsigned) ch <= 0x7F)
33: #define TOASCII(ch)     ((unsigned) ch & 0x7F)
34:
35: #define TOUPPER(ch)     ((ch) & ~32)
36: #define TOLOWER(ch)     ((ch) | 32)
37:
38: #endif /* _FIWIX_CTYPE_H */

```


include/fiwix/devices.h

Page 1/1

```

1: /*
2:  * fiwix/include/fiwix/devices.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_DEVICES_H
9: #define _FIWIX_DEVICES_H
10:
11: #include <fiwix/types.h>
12: #include <fiwix/fs.h>
13:
14: #define NR_BLKDEV          128      /* maximum number of block devices */
15: #define NR_CHRDEV          128      /* maximum number of char devices */
16:
17: #define BLK_DEV            1        /* block device */
18: #define CHR_DEV            2        /* character device */
19:
20: #define SET_MINOR(minors, bit)    ((minors[(bit) / 32]) |= (1 << ((bit) % 32)))
21: #define CLEAR_MINOR(minors, bit) ((minors[(bit) / 32]) &= ~(1 << ((bit) % 32)))
22: #define TEST_MINOR(minors, bit)  ((minors[(bit) / 32]) & (1 << ((bit) % 32)))
23:
24: struct device {
25:     char *name;
26:     unsigned char major;
27:     unsigned int minors[8];        /* bitmap of 256 bits */
28:     int blksize;
29:     void *device_data;            /* mostly used for minor sizes, in KB */
30:     struct fs_operations *fsop;
31:     struct device *next;
32: };
33:
34: extern struct device *chr_device_table[NR_CHRDEV];
35: extern struct device *blk_device_table[NR_BLKDEV];
36:
37: int register_device(int, struct device *);
38: struct device * get_device(int, __dev_t);
39: int chr_dev_open(struct inode *, struct fd *);
40: int blk_dev_open(struct inode *, struct fd *);
41: int blk_dev_close(struct inode *, struct fd *);
42: int blk_dev_read(struct inode *, struct fd *, char *, __size_t);
43: int blk_dev_write(struct inode *, struct fd *, const char *, __size_t);
44: int blk_dev_ioctl(struct inode *, int, unsigned long int);
45: int blk_dev_lseek(struct inode *, __off_t);
46:
47: void dev_init(void);
48:
49: #endif /* _FIWIX_DEVICES_H */

```

include/fiwix/dirent.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/dirent.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_DIRENT_H
9: #define _FIWIX_DIRENT_H
10:
11: #include <fiwix/types.h>
12: #include <fiwix/limits.h>
13:
14: struct dirent {
15:     __ino_t d_ino;           /* inode number */
16:     __off_t d_off;          /* offset to next dirent */
17:     unsigned short int d_reclen; /* length of this dirent */
18:     char d_name[NAME_MAX + 1]; /* file name (null-terminated) */
19: };
20:
21: #endif /* _FIWIX_DIRENT_H */
```

include/fiwix/dma.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/dma.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_DMA_H
9: #define _FIWIX_DMA_H
10:
11: #define DMA_CHANNELS          8          /* max. number of DMA channels */
12:
13: #define DMA_MASK_CHANNEL     0x04
14: #define DMA_UNMASK_CHANNEL   0x00
15:
16: #define DMA_MODE_VERIFY      0x00
17: #define DMA_MODE_WRITE       0x04      /* read device -> write memory */
18: #define DMA_MODE_READ        0x08      /* read memory -> write device */
19: #define DMA_MODE_AUTOINIT    0x10
20: #define DMA_MODE_ADDRES_DEC  0x20
21: #define DMA_MODE_DEMAND      0x00
22: #define DMA_MODE_SINGLE      0x40
23: #define DMA_MODE_BLOCK       0x80
24: #define DMA_MODE_CASCADE    0xC0
25:
26: char *dma_resources[DMA_CHANNELS];
27:
28: void start_dma(int, void *, unsigned int, int);
29: int dma_register(int, char *);
30: int dma_unregister(int);
31: void dma_init(void);
32:
33: #endif /* _FIWIX_DMA_H */
```

include/fiwix/errno.h

Page 1/3

```

1: #ifndef _FIWIX_ERRNO_H
2: #define _FIWIX_ERRNO_H
3:
4: #define EPERM          1      /* Operation not permitted - Not owner */
5: #define ENOENT         2      /* No such file or directory */
6: #define ESRCH         3      /* No such process */
7: #define EINTR         4      /* Interrupted system call */
8: #define EIO           5      /* I/O error */
9: #define ENXIO         6      /* No such device or address */
10: #define E2BIG         7      /* Arg list too long */
11: #define ENOEXEC       8      /* Exec format error */
12: #define EBADF         9      /* Bad file number */
13: #define ECHILD        10     /* No child processes */
14: #define EAGAIN        11     /* Try again - No more processes */
15: #define ENOMEM        12     /* Out of memory - No enough space */
16: #define EACCES        13     /* Permission denied */
17: #define EFAULT        14     /* Bad address */
18: #define ENOTBLK       15     /* Block device required */
19: #define EBUSY         16     /* Device or resource busy */
20: #define EEXIST        17     /* File exists */
21: #define EXDEV         18     /* Cross-device link */
22: #define ENODEV        19     /* No such device */
23: #define ENOTDIR       20     /* Not a directory */
24: #define EISDIR        21     /* Is a directory */
25: #define EINVAL        22     /* Invalid argument */
26: #define ENFILE        23     /* File table overflow */
27: #define EMFILE        24     /* Too many open files */
28: #define ENOTTY        25     /* Not a typewriter */
29: #define ETXTBSY       26     /* Text file busy */
30: #define EFBIG         27     /* File too large */
31: #define ENOSPC        28     /* No space left on device */
32: #define ESPIPE        29     /* Illegal seek */
33: #define EROFS         30     /* Read-only file system */
34: #define EMLINK        31     /* Too many links */
35: #define EPIPE         32     /* Broken pipe */
36: #define EDOM          33     /* Math argument out of domain of func */
37: #define ERANGE        34     /* Math result not representable */
38: #define EDEADLK       35     /* Resource deadlock would occur */
39: #define ENAMETOOLONG  36     /* File name too long */
40: #define ENOLCK        37     /* No record locks available */
41: #define ENOSYS        38     /* Function not implemented */
42: #define ENOTEMPTY     39     /* Directory not empty */
43: #define ELOOP         40     /* Too many symbolic links encountered */
44: #define EWouldBlock   EAGAIN /* Operation would block */
45: #define ENOMSG        42     /* No message of desired type */
46: #define EIDRM         43     /* Identifier removed */
47: #define ECHRNG        44     /* Channel number out of range */
48: #define EL2NSYNC      45     /* Level 2 not synchronized */
49: #define EL3HLT        46     /* Level 3 halted */
50: #define EL3RST        47     /* Level 3 reset */
51: #define ELNRNG        48     /* Link number out of range */
52: #define EUNATCH       49     /* Protocol driver not attached */
53: #define ENOCCSI       50     /* No CSI structure available */
54: #define EL2HLT        51     /* Level 2 halted */
55: #define EBADE         52     /* Invalid exchange */
56: #define EBADR         53     /* Invalid request descriptor */
57: #define EXFULL        54     /* Exchange full */
58: #define ENOANO        55     /* No anode */
59: #define EBADRQC       56     /* Invalid request code */
60: #define EBADSLT       57     /* Invalid slot */
61:
62: #define EDEADLOCK     EDEADLK
63:
64: #define EBFONT        59     /* Bad font file format */
65: #define ENOSTR        60     /* Device not a stream */
66: #define ENODATA       61     /* No data available */
67: #define ETIME         62     /* Timer expired */

```

include/fiwix/errno.h

Page 2/3

```

68: #define ENOSR          63      /* Out of streams resources */
69: #define ENONET         64      /* Machine is not on the network */
70: #define ENOPKG         65      /* Package not installed */
71: #define EREMOTE        66      /* Object is remote */
72: #define ENOLINK        67      /* Link has been severed */
73: #define EADV           68      /* Advertise error */
74: #define ESRMNT         69      /* Srmount error */
75: #define ECOMM          70      /* Communication error on send */
76: #define EPROTO         71      /* Protocol error */
77: #define EMULTIHOP       72      /* Multihop attempted */
78: #define EDOTDOT        73      /* RFS specific error */
79: #define EBADMSG        74      /* Not a data message */
80: #define EOVERFLOW      75      /* Value too large for defined data type */
81: #define ENOTUNIQ       76      /* Name not unique on network */
82: #define EBADF          77      /* File descriptor in bad state */
83: #define EREMCHG        78      /* Remote address changed */
84: #define ELIBACC        79      /* Can not access a needed shared library */
85: #define ELIBBAD        80      /* Accessing a corrupted shared library */
86: #define ELIBSCN        81      /* .lib section in a.out corrupted */
87: #define ELIBMAX        82      /* Attempting to link in too many shared librari
es */
88: #define ELIBEXEC       83      /* Cannot exec a shared library directly */
89: #define EILSEQ         84      /* Illegal byte sequence */
90: #define ERESTART       85      /* Interrupted system call should be restarted */
/
91: #define ESTRPIPE       86      /* Streams pipe error */
92: #define EUSERS         87      /* Too many users */
93: #define ENOTSOCK       88      /* Socket operation on non-socket */
94: #define EDESTADDRREQ   89      /* Destination address required */
95: #define EMSGSIZE       90      /* Message too long */
96: #define EPROTOTYPE     91      /* Protocol wrong type for socket */
97: #define ENOPROTOOPT    92      /* Protocol not available */
98: #define EPROTONOSUPPORT 93      /* Protocol not supported */
99: #define ESOCKTNOSUPPORT 94      /* Socket type not supported */
100: #define EOPNOTSUPP     95      /* Operation not supported on transport endpoint
*/
101: #define EPNOSUPPORT    96      /* Protocol family not supported */
102: #define EAFNOSUPPORT    97      /* Address family not supported by protocol */
103: #define EADDRINUSE     98      /* Address already in use */
104: #define EADDRNOTAVAIL  99      /* Cannot assign requested address */
105: #define ENETDOWN       100     /* Network is down */
106: #define ENETUNREACH    101     /* Network is unreachable */
107: #define ENETRESET      102     /* Network dropped connection because of reset */
/
108: #define ECONNABORTED   103     /* Software caused connection abort */
109: #define ECONNRESET     104     /* Connection reset by peer */
110: #define ENOBUFS        105     /* No buffer space available */
111: #define EISCONN        106     /* Transport endpoint is already connected */
112: #define ENOTCONN       107     /* Transport endpoint is not connected */
113: #define ESHUTDOWN      108     /* Cannot send after transport endpoint shutdown
*/
114: #define ETOOMANYREFS   109     /* Too many references: cannot splice */
115: #define ETIMEDOUT      110     /* Connection timed out */
116: #define ECONNREFUSED   111     /* Connection refused */
117: #define EHOSTDOWN      112     /* Host is down */
118: #define EHOSTUNREACH   113     /* No route to host */
119: #define EALREADY       114     /* Operation already in progress */
120: #define EINPROGRESS    115     /* Operation now in progress */
121: #define ESTALE         116     /* Stale NFS file handle */
122: #define EUCLEAN        117     /* Structure needs cleaning */
123: #define ENOTNAM        118     /* Not a XENIX named type file */
124: #define ENAVAIL        119     /* No XENIX semaphores available */
125: #define EISNAM         120     /* Is a named type file */
126: #define EREMOTEIO      121     /* Remote I/O error */
127: #define EDQUOT         122     /* Quota exceeded */
128:
129: #define ENOMEDIUM      123     /* No medium found */

```

include/fiwix/errno.h

Page 3/3

```
130: #define EMEDIUMTYPE      124      /* Wrong medium type          */
131:
132: #endif /* _FIWIX_ERRNO_H */
```

include/fiwix/fbcon.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/fbcon.h
3:  *
4:  * Copyright 2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_FBCON_H
9: #define _FIWIX_FBCON_H
10:
11: void fbcon_put_char(struct vconsole *, unsigned char);
12:
13: void fbcon_put_char(struct vconsole *, unsigned char);
14: void fbcon_insert_char(struct vconsole *);
15: void fbcon_delete_char(struct vconsole *);
16: void fbcon_update_curpos(struct vconsole *);
17: void fbcon_show_cursor(struct vconsole *, int);
18: void fbcon_get_curpos(struct vconsole *);
19: void fbcon_write_screen(struct vconsole *, int, int, short int);
20: void fbcon_blank_screen(struct vconsole *);
21: void fbcon_scroll_screen(struct vconsole *, int, int);
22: void fbcon_restore_screen(struct vconsole *);
23: void fbcon_screen_on(struct vconsole *);
24: void fbcon_screen_off(unsigned int);
25: void fbcon_buf_scroll(struct vconsole *, int);
26: void fbcon_cursor_blink(unsigned int);
27: void fbcon_init(void);
28:
29: #endif /* _FIWIX_FBCON_H */
```

include/fiwix/fb.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/fb.h
3:  *
4:  * Copyright 2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_FB_H
9: #define _FIWIX_FB_H
10:
11: #include <fiwix/fs.h>
12:
13: #define FB_MAJOR          29      /* major number */
14: #define FB_MINORS        1       /* number of supported minors */
15:
16: int fb_open(struct inode *, struct fd *);
17: int fb_close(struct inode *, struct fd *);
18: int fb_read(struct inode *, struct fd *, char *, __size_t);
19: int fb_write(struct inode *, struct fd *, const char *, __size_t);
20: int fb_ioctl(struct inode *, int, unsigned long int);
21: int fb_lseek(struct inode *, __off_t);
22:
23: void fb_init(void);
24:
25: #endif /* _FIWIX_FB_H */
```


include/fiwix/fcntl.h

Page 1/1

```

1: /*
2:  * fiwix/include/fiwix/fcntl.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_FCNTL_H
9: #define _FIWIX_FCNTL_H
10:
11: #include <fiwix/types.h>
12:
13: #define O_ACCMODE          0003
14: #define O_RDONLY          00
15: #define O_WRONLY          01
16: #define O_RDWR           02
17:
18: /* for open() only */
19: #define O_CREAT            0100 /* create file if it does not exist */
20: #define O_EXCL             0200 /* exclusive use flag */
21: #define O_NOCTTY          0400 /* do not assign controlling terminal */
22: #define O_TRUNC           01000 /* truncate flag */
23: #define O_NOFOLLOW        0400000 /* do not follow symbolic links */
24:
25: #define O_APPEND           02000
26: #define O_NONBLOCK        04000
27: #define O_NDELAY           O_NONBLOCK
28: #define O_SYNC             010000
29:
30: #define F_DUPFD            0      /* duplicate file descriptor */
31: #define F_GETFD            1      /* get file descriptor flags */
32: #define F_SETFD            2      /* set file descriptor flags */
33: #define F_GETFL            3      /* get status flags and file access modes */
34: #define F_SETFL            4      /* set file status flags */
35: #define F_GETLK            5      /* get record locking information */
36: #define F_SETLK            6      /* set record locking information */
37: #define F_SETLKW           7      /* same as F_SETLK; wait if blocked */
38:
39: /* get/set process or process group ID to receive SIGURG signals */
40: #define F_SETOWN            8      /* for sockets only */
41: #define F_GETOWN            9      /* for sockets only */
42:
43: /* for F_[GET|SET]FL */
44: #define FD_CLOEXEC          1      /* close the file descriptor upon exec() */
45:
46: /* for POSIX fcntl() */
47: #define F_RDLCK             0      /* shared or read lock */
48: #define F_WRLCK             1      /* exclusive or write lock */
49: #define F_UNLCK             2      /* unlock */
50:
51: /* for BSD flock() */
52: #define LOCK_SH              1      /* shared lock */
53: #define LOCK_EX              2      /* exclusive lock */
54: #define LOCK_NB              4      /* or'd with one of the above to prevent
55:                                     blocking */
56: #define LOCK_UN              8      /* unlock */
57:
58: /* IEEE Std 1003.1, 2004 Edition */
59: struct flock {
60:     short int l_type;          /* type of lock: F_RDLCK, F_WRLCK, F_UNLCK */
61:     short int l_whence;       /* flag for 'l_start': SEEK_SET, SEEK_CUR, ... */
62:     __off_t l_start;          /* relative offset in bytes */
63:     __off_t l_len;            /* size; if 0 then until EOF */
64:     __pid_t l_pid;            /* PID holding the lock; returned in F_GETLK */
65: };
66:
67: #endif /* _FIWIX_FCNTL_H */

```

include/fiwix/filesystems.h

Page 1/3

```

1: /*
2:  * fiwix/include/fiwix/filesystems.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_FILESYSTEMS_H
9: #define _FIWIX_FILESYSTEMS_H
10:
11: #include <fiwix/types.h>
12: #include <fiwix/limits.h>
13:
14: #define NR_FILESYSTEMS          5          /* supported filesystems */
15:
16: /* value to be determined during system startup */
17: extern unsigned int mount_table_size; /* size in bytes */
18:
19: struct filesystems {
20:     const char *name;          /* filesystem name */
21:     struct fs_operations *fsop; /* filesystem operations */
22:     struct mount *mt;          /* mount-table entry (only for nodev) */
23: };
24: struct filesystems filesystems_table[NR_FILESYSTEMS];
25:
26: struct mount {
27:     __dev_t dev;                /* device number */
28:     char devname[DEVNAME_MAX + 1]; /* device name */
29:     char dirname[NAME_MAX + 1]; /* mount point directory name */
30:     unsigned char used;          /* 1=busy, 0=free */
31:     struct superblock sb;        /* superblock */
32:     struct filesystems *fs;      /* pointer to filesystem structure */
33: };
34: extern struct mount *mount_table;
35:
36: int register_filesystem(const char *, struct fs_operations *);
37: struct filesystems * get_filesystem(const char *);
38: void fs_init(void);
39:
40: struct superblock * get_superblock(__dev_t);
41: void sync_superblocks(__dev_t);
42: int kern_mount(__dev_t, struct filesystems *);
43: int mount_root(void);
44: void mount_init(void);
45:
46:
47: /* minix prototypes */
48: int minix_file_open(struct inode *, struct fd *);
49: int minix_file_close(struct inode *, struct fd *);
50: int minix_file_write(struct inode *, struct fd *, const char *, __size_t);
51: int minix_file_lseek(struct inode *, __off_t);
52: int minix_dir_open(struct inode *, struct fd *);
53: int minix_dir_close(struct inode *, struct fd *);
54: int minix_dir_read(struct inode *, struct fd *, char *, __size_t);
55: int minix_dir_write(struct inode *, struct fd *, const char *, __size_t);
56: int minix_dir_readdir(struct inode *, struct fd *, struct dirent *, unsigned int
);
57: int minix_readlink(struct inode *, char *, __size_t);
58: int minix_followlink(struct inode *, struct inode *, struct inode **);
59: int minix_bmap(struct inode *, __off_t, int);
60: int minix_lookup(const char *, struct inode *, struct inode **);
61: int minix_rmdir(struct inode *, struct inode *);
62: int minix_link(struct inode *, struct inode *, char *);
63: int minix_unlink(struct inode *, struct inode *, char *);
64: int minix_symlink(struct inode *, char *, char *);
65: int minix_mkdir(struct inode *, char *, __mode_t);
66: int minix_mknod(struct inode *, char *, __mode_t, __dev_t);

```

include/fiwix/filesystems.h

Page 2/3

```

67: int minix_truncate(struct inode *, __off_t);
68: int minix_create(struct inode *, char *, __mode_t, struct inode **);
69: int minix_rename(struct inode *, struct inode *, struct inode *, struct inode *,
char *, char *);
70: int minix_read_inode(struct inode *);
71: int minix_write_inode(struct inode *);
72: int minix_ialloc(struct inode *, int);
73: void minix_ifree(struct inode *);
74: void minix_statfs(struct superblock *, struct statfs *);
75: int minix_read_superblock(__dev_t, struct superblock *);
76: int minix_remount_fs(struct superblock *, int);
77: int minix_write_superblock(struct superblock *);
78: void minix_release_superblock(struct superblock *);
79: int minix_init(void);
80:
81:
82: /* ext2 prototypes */
83: int ext2_file_open(struct inode *, struct fd *);
84: int ext2_file_close(struct inode *, struct fd *);
85: int ext2_file_write(struct inode *, struct fd *, const char *, __size_t);
86: int ext2_file_lseek(struct inode *, __off_t);
87: int ext2_dir_open(struct inode *, struct fd *);
88: int ext2_dir_close(struct inode *, struct fd *);
89: int ext2_dir_read(struct inode *, struct fd *, char *, __size_t);
90: int ext2_dir_write(struct inode *, struct fd *, const char *, __size_t);
91: int ext2_dir_readdir(struct inode *, struct fd *, struct dirent *, unsigned int)
;
92: int ext2_readlink(struct inode *, char *, __size_t);
93: int ext2_followlink(struct inode *, struct inode *, struct inode **);
94: int ext2_bmap(struct inode *, __off_t, int);
95: int ext2_lookup(const char *, struct inode *, struct inode **);
96: int ext2_rmdir(struct inode *, struct inode *);
97: int ext2_link(struct inode *, struct inode *, char *);
98: int ext2_unlink(struct inode *, struct inode *, char *);
99: int ext2_symlink(struct inode *, char *, char *);
100: int ext2_mkdir(struct inode *, char *, __mode_t);
101: int ext2_mknod(struct inode *, char *, __mode_t, __dev_t);
102: int ext2_truncate(struct inode *, __off_t);
103: int ext2_create(struct inode *, char *, __mode_t, struct inode **);
104: int ext2_rename(struct inode *, struct inode *, struct inode *, struct inode *,
char *, char *);
105: int ext2_read_inode(struct inode *);
106: int ext2_write_inode(struct inode *);
107: int ext2_ialloc(struct inode *, int);
108: void ext2_ifree(struct inode *);
109: void ext2_statfs(struct superblock *, struct statfs *);
110: int ext2_read_superblock(__dev_t, struct superblock *);
111: int ext2_remount_fs(struct superblock *, int);
112: int ext2_write_superblock(struct superblock *);
113: void ext2_release_superblock(struct superblock *);
114: int ext2_init(void);
115:
116:
117: /* pipefs prototypes */
118: int fifo_open(struct inode *, struct fd *);
119: int pipefs_close(struct inode *, struct fd *);
120: int pipefs_read(struct inode *, struct fd *, char *, __size_t);
121: int pipefs_write(struct inode *, struct fd *, const char *, __size_t);
122: int pipefs_ioctl(struct inode *, int, unsigned long int);
123: int pipefs_lseek(struct inode *, __off_t);
124: int pipefs_select(struct inode *, int);
125: int pipefs_ialloc(struct inode *, int);
126: void pipefs_ifree(struct inode *);
127: int pipefs_read_superblock(__dev_t, struct superblock *);
128: int pipefs_init(void);
129:
130:

```

include/fiwix/filesystems.h

Page 3/3

```

131: /* iso9660 prototypes */
132: int iso9660_file_open(struct inode *, struct fd *);
133: int iso9660_file_close(struct inode *, struct fd *);
134: int iso9660_file_lseek(struct inode *, __off_t);
135: int iso9660_dir_open(struct inode *, struct fd *);
136: int iso9660_dir_close(struct inode *, struct fd *);
137: int iso9660_dir_read(struct inode *, struct fd *, char *, __size_t);
138: int iso9660_dir_readdir(struct inode *, struct fd *, struct dirent *, unsigned i
nt);
139: int iso9660_readlink(struct inode *, char *, __size_t);
140: int iso9660_followlink(struct inode *, struct inode *, struct inode **);
141: int iso9660_bmap(struct inode *, __off_t, int);
142: int iso9660_lookup(const char *, struct inode *, struct inode **);
143: int iso9660_read_inode(struct inode *);
144: void iso9660_statfs(struct superblock *, struct statfs *);
145: int iso9660_read_superblock(__dev_t, struct superblock *);
146: void iso9660_release_superblock(struct superblock *);
147: int iso9660_init(void);
148:
149:
150: /* procfs prototypes */
151: int procfs_file_open(struct inode *, struct fd *);
152: int procfs_file_close(struct inode *, struct fd *);
153: int procfs_file_read(struct inode *, struct fd *, char *, __size_t);
154: int procfs_file_lseek(struct inode *, __off_t);
155: int procfs_dir_open(struct inode *, struct fd *);
156: int procfs_dir_close(struct inode *, struct fd *);
157: int procfs_dir_read(struct inode *, struct fd *, char *, __size_t);
158: int procfs_dir_readdir(struct inode *, struct fd *, struct dirent *, unsigned in
t);
159: int procfs_readlink(struct inode *, char *, __size_t);
160: int procfs_followlink(struct inode *, struct inode *, struct inode **);
161: int procfs_bmap(struct inode *, __off_t, int);
162: int procfs_lookup(const char *, struct inode *, struct inode **);
163: int procfs_read_inode(struct inode *);
164: void procfs_statfs(struct superblock *, struct statfs *);
165: int procfs_read_superblock(__dev_t, struct superblock *);
166: int procfs_init(void);
167:
168: #endif /* _FIWIX_FILESYSTEMS_H */

```

include/fiwix/floppy.h

Page 1/2

```

1: /*
2:  * fiwix/include/fiwix/floppy.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_FLOPPY_H
9: #define _FIWIX_FLOPPY_H
10:
11: #include <fiwix/fs.h>
12: #include <fiwix/sigcontext.h>
13:
14: #define FLOPPY_IRQ        6
15: #define FLOPPY_DMA        2        /* DMA channel */
16:
17: #define FDC_MAJOR        2        /* fdd major number */
18:
19: #define FDC_SECTSIZE      512      /* sector size (in bytes) */
20: #define FDC_TR_DEFAULT    0        /* timer reason is IRQ */
21: #define FDC_TR_MOTOR      1        /* timer reason is motor on */
22:
23: #define FDC_SRA           0x3F0    /* Status Register A */
24: #define FDC_SRB           0x3F1    /* Status Register B */
25: #define FDC_DOR           0x3F2    /* Digital Output Register */
26: #define FDC_MSR           0x3F4    /* Main Status Register */
27: #define FDC_DATA          0x3F5    /* command/data register */
28: #define FDC_DIR           0x3F7    /* Digital Input Register */
29: #define FDC_CCR           0x3F7    /* Configuration Control Register */
30:
31: #define FDC_ENABLE        0x04    /* bit #2 FDC enabled (normal op) */
32: #define FDC_DMA_ENABLE    0x08    /* bit #3 DMA enabled */
33: #define FDC_DRIVE0        0x10    /* motor on for the first drive, the rest will
34:  * be calculated by left-shifting this value
35:  * with 'current_fdd'.
36:  */
37:
38: #define FDC_DIO           0x40    /* bit #6 DIO I/O direction */
39: #define FDC_RQM           0x80    /* bit #7 RQM is ready for I/O */
40:
41: #define MAX_FDC_RESULTS   7
42: #define MAX_FDC_ERR       5
43:
44: #define FDC_RESET         0xFF    /* reset indicator */
45: #define FDC_READ          0xE6
46: #define FDC_WRITE         0xC5
47: #define FDC_VERSION       0x10
48: #define FDC_FORMAT_TRK    0x4D
49: #define FDC_RECALIBRATE   0x07
50: #define FDC_SENSEI        0x08
51: #define FDC_SPECIFY       0x03
52: #define FDC_SEEK          0x0F
53: #define FDC_LOCK          0x14
54: #define FDC_PARTID        0x18
55:
56: #define ST0               0x00    /* Status Register 0 */
57: #define ST1               0x01    /* Status Register 1 */
58: #define ST2               0x02    /* Status Register 2 */
59:
60: #define ST0_IC            0xC0    /* bits #7 and #6 interrupt code */
61: #define ST0_SE            0x20    /* bit #5 successful implied seek */
62: #define ST0_RECALIBRATE   ST0_SE  /* bit #5 successful FDC_RECALIBRATE */
63: #define ST0_SEEK          ST0_SE  /* bit #5 successful FDC_SEEK */
64: #define ST0_UC            0x10    /* bit #4 unit needs check (fault) */
65: #define ST0_NR            0x8     /* bit #3 drive not ready */
66:
67: #define ST1_NW            0x02    /* bit #1 not writable */

```

include/fiwix/floppy.h

Page 2/2

```
68:
69: #define ST_PCN          0x01    /* present cylinder */
70: #define ST_CYL         0x03    /* cylinder returned */
71: #define ST_HEAD        0x04    /* head returned */
72: #define ST_SECTOR      0x05    /* sector returned */
73:
74: /* floppy disk drive type */
75: struct fdtd {
76:     short int size;           /* number of sectors */
77:     short int sizekb;        /* size in KB */
78:     char tracks;             /* number of tracks */
79:     char spt;                /* number of sectors per track */
80:     char heads;              /* number of heads */
81:     char gpl1;               /* GAP in READ/WRITE operations */
82:     char gpl2;               /* GAP in FORMAT TRACK operations */
83:     char rate;               /* data rate value */
84:     char spec;               /* SRT+HUT (StepRate + HeadUnload) Time */
85:     char hlt;                /* HLT (Head Load Time) */
86:     char *name;              /* unit name */
87: };
88:
89: void irq_floppy(int, struct sigcontext *);
90: void fdc_timer(unsigned int);
91:
92: int fdc_open(struct inode *, struct fd *);
93: int fdc_close(struct inode *, struct fd *);
94: int fdc_read(__dev_t, __blk_t, char *, int);
95: int fdc_write(__dev_t, __blk_t, char *, int);
96: int fdc_ioctl(struct inode *, int, unsigned long int);
97: int fdc_lseek(struct inode *, __off_t);
98:
99: void floppy_init(void);
100:
101: #endif /* _FIWIX_FLOPPY_H */
```

include/fiwix/font.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/font.h
3:  *
4:  * Copyright 2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_FONT_H
9: #define _FIWIX_FONT_H
10:
11: struct fbcon_font_desc {
12:     char *name;
13:     int width;
14:     int height;
15:     void *data;
16:     void *cursorshape;
17: };
18:
19: extern struct fbcon_font_desc font_vga_8x8;
20: extern struct fbcon_font_desc font_vga_8x10;
21: extern struct fbcon_font_desc font_vga_8x12;
22: extern struct fbcon_font_desc font_vga_8x14;
23: extern struct fbcon_font_desc font_vga_8x16;
24:
25: struct fbcon_font_desc *fbcon_find_font(int);
26:
27: #endif /* _FIWIX_FONT_H */
```

include/fiwix/fs_ext2.h

Page 1/4

```

1: /*
2:  * fiwix/include/fiwix/fs_ext2.h
3:  *
4:  * This file from: Linux 2.0.40
5:  * Copyright (C) 1992, 1993, 1994, 1995
6:  * Remy Card (card@masi.ibp.fr)
7:  * Laboratoire MASI - Institut Blaise Pascal
8:  * Universite Pierre et Marie Curie (Paris VI)
9:  * Copyright (C) 1991, 1992 Linus Torvalds
10: */
11:
12: #ifndef _FIWIX_FS_EXT2_H
13: #define _FIWIX_FS_EXT2_H
14:
15: #include <fiwix/types.h>
16:
17: #define EXT2_ROOT_INO          2          /* Root inode */
18: #define EXT2_SUPER_MAGIC      0xEF53
19:
20: /*
21:  * Macro-instructions used to manage several block sizes
22:  */
23: #define EXT2_MIN_BLOCK_SIZE    1024
24: #define EXT2_MAX_BLOCK_SIZE    4096
25: #define EXT2_MIN_BLOCK_LOG_SIZE 10
26: # define EXT2_BLOCK_SIZE(s)    ((s)->s_blocksize)
27: # define EXT2_BLOCK_SIZE_BITS(s) ((s)->s_blocksize_bits)
28:
29: /*
30:  * Structure of a blocks group descriptor
31:  */
32: struct ext2_group_desc
33: {
34:     __u32    bg_block_bitmap;          /* Blocks bitmap block */
35:     __u32    bg_inode_bitmap;         /* Inodes bitmap block */
36:     __u32    bg_inode_table;          /* Inodes table block */
37:     __u16    bg_free_blocks_count;    /* Free blocks count */
38:     __u16    bg_free_inodes_count;    /* Free inodes count */
39:     __u16    bg_used_dirs_count;      /* Directories count */
40:     __u16    bg_pad;
41:     __u32    bg_reserved[3];
42: };
43:
44: /*
45:  * Macro-instructions used to manage group descriptors
46:  */
47: #define EXT2_BLOCKS_PER_GROUP(s)    ((s)->u.ext2.sb.s_blocks_per_group)
48: #define EXT2_INODES_PER_GROUP(s)    ((s)->u.ext2.sb.s_inodes_per_group)
49: # define EXT2_DESC_PER_BLOCK_BITS(s) ((s)->u.ext2_sb.s_desc_per_block_bits)
50: #define EXT2_DESC_PER_BLOCK(s)      ((s)->u.ext2.desc_per_block)
51:
52: /*
53:  * Constants relative to the data blocks
54:  */
55: #define EXT2_NDIR_BLOCKS          12
56: #define EXT2_IND_BLOCK            EXT2_NDIR_BLOCKS
57: #define EXT2_DIND_BLOCK           (EXT2_IND_BLOCK + 1)
58: #define EXT2_TIND_BLOCK           (EXT2_DIND_BLOCK + 1)
59: #define EXT2_N_BLOCKS             (EXT2_TIND_BLOCK + 1)
60:
61: /*
62:  * Structure of an inode on the disk
63:  */
64: struct ext2_inode {
65:     __u16    i_mode;                  /* File mode */
66:     __u16    i_uid;                  /* Low 16 bits of Owner Uid */
67:     __u32    i_size;                 /* Size in bytes */

```


include/fiwix/fs_ext2.h

Page 2/4

```

68:     __u32    i_atime;           /* Access time */
69:     __u32    i_ctime;           /* Creation time */
70:     __u32    i_mtime;           /* Modification time */
71:     __u32    i_dtime;           /* Deletion Time */
72:     __u16    i_gid;             /* Low 16 bits of Group Id */
73:     __u16    i_links_count;     /* Links count */
74:     __u32    i_blocks;          /* Blocks count */
75:     __u32    i_flags;           /* File flags */
76:     union {
77:         struct {
78:             __u32    l_i_reserved1;
79:         } linux1;
80:         struct {
81:             __u32    h_i_translator;
82:         } hurd1;
83:         struct {
84:             __u32    m_i_reserved1;
85:         } masix1;
86:     } osd1;                       /* OS dependent 1 */
87:     __u32    i_block[EXT2_N_BLOCKS]; /* Pointers to blocks */
88:     __u32    i_generation;       /* File version (for NFS) */
89:     __u32    i_file_acl;         /* File ACL */
90:     __u32    i_dir_acl;          /* Directory ACL */
91:     __u32    i_faddr;            /* Fragment address */
92:     union {
93:         struct {
94:             __u8     l_i_frag;     /* Fragment number */
95:             __u8     l_i_fsize;    /* Fragment size */
96:             __u16    i_pad1;
97:             __u16    l_i_uid_high; /* these 2 fields */
98:             __u16    l_i_gid_high; /* were reserved2[0] */
99:             __u32    l_i_reserved2;
100:        } linux2;
101:        struct {
102:            __u8     h_i_frag;     /* Fragment number */
103:            __u8     h_i_fsize;    /* Fragment size */
104:            __u16    h_i_mode_high;
105:            __u16    h_i_uid_high;
106:            __u16    h_i_gid_high;
107:            __u32    h_i_author;
108:        } hurd2;
109:        struct {
110:            __u8     m_i_frag;     /* Fragment number */
111:            __u8     m_i_fsize;    /* Fragment size */
112:            __u16    m_pad1;
113:            __u32    m_i_reserved2[2];
114:        } masix2;
115:    } osd2;                       /* OS dependent 2 */
116: };
117:
118: /*
119:  * File system states
120:  */
121: #define EXT2_VALID_FS            0x0001 /* Unmounted cleanly */
122: #define EXT2_ERROR_FS           0x0002 /* Errors detected */
123:
124: /*
125:  * Structure of the super block
126:  */
127: struct ext2_super_block {
128:     __u32    s_inodes_count;      /* Inodes count */
129:     __u32    s_blocks_count;      /* Blocks count */
130:     __u32    s_r_blocks_count;    /* Reserved blocks count */
131:     __u32    s_free_blocks_count; /* Free blocks count */
132:     __u32    s_free_inodes_count; /* Free inodes count */
133:     __u32    s_first_data_block;  /* First Data Block */
134:     __u32    s_log_block_size;    /* Block size */

```

include/fiwix/fs_ext2.h

Page 3/4

```

135:     __s32    s_log_frag_size;          /* Fragment size */
136:     __u32    s_blocks_per_group;      /* # Blocks per group */
137:     __u32    s_frags_per_group;       /* # Fragments per group */
138:     __u32    s_inodes_per_group;     /* # Inodes per group */
139:     __u32    s_mtime;                /* Mount time */
140:     __u32    s_wtime;                /* Write time */
141:     __u16    s_mnt_count;             /* Mount count */
142:     __s16    s_max_mnt_count;        /* Maximal mount count */
143:     __u16    s_magic;                /* Magic signature */
144:     __u16    s_state;                /* File system state */
145:     __u16    s_errors;               /* Behaviour when detecting errors */
146:     __u16    s_minor_rev_level;      /* minor revision level */
147:     __u32    s_lastcheck;            /* time of last check */
148:     __u32    s_checkinterval;        /* max. time between checks */
149:     __u32    s_creator_os;           /* OS */
150:     __u32    s_rev_level;            /* Revision level */
151:     __u16    s_def_resuid;           /* Default uid for reserved blocks */
152:     __u16    s_def_resgid;           /* Default gid for reserved blocks */
153:     /*
154:      * These fields are for EXT2_DYNAMIC_REV superblocks only.
155:      *
156:      * Note: the difference between the compatible feature set and
157:      * the incompatible feature set is that if there is a bit set
158:      * in the incompatible feature set that the kernel doesn't
159:      * know about, it should refuse to mount the filesystem.
160:      *
161:      * e2fsck's requirements are more strict; if it doesn't know
162:      * about a feature in either the compatible or incompatible
163:      * feature set, it must abort and not try to meddle with
164:      * things it doesn't understand...
165:      */
166:     __u32    s_first_ino;             /* First non-reserved inode */
167:     __u16    s_inode_size;           /* size of inode structure */
168:     __u16    s_block_group_nr;      /* block group # of this superblock */
169:     __u32    s_feature_compat;       /* compatible feature set */
170:     __u32    s_feature_incompat;     /* incompatible feature set */
171:     __u32    s_feature_ro_compat;    /* readonly-compatible feature set */
172:     __u8     s_uuid[16];             /* 128-bit uuid for volume */
173:     char     s_volume_name[16];      /* volume name */
174:     char     s_last_mounted[64];     /* directory where last mounted */
175:     __u32    s_algorithm_usage_bitmap; /* For compression */
176:     /*
177:      * Performance hints. Directory preallocation should only
178:      * happen if the EXT2_COMPAT_PREALLOC flag is on.
179:      */
180:     __u8     s_prealloc_blocks;      /* Nr of blocks to try to preallocate*/
181:     __u8     s_prealloc_dir_blocks;  /* Nr to preallocate for dirs */
182:     __u16    s_padding1;
183:     /*
184:      * Journaling support valid if EXT3_FEATURE_COMPAT_HAS_JOURNAL set.
185:      */
186:     __u8     s_journal_uuid[16];     /* uuid of journal superblock */
187:     __u32    s_journal_inum;         /* inode number of journal file */
188:     __u32    s_journal_dev;         /* device number of journal file */
189:     __u32    s_last_orphan;         /* start of list of inodes to delete */
190:     __u32    s_hash_seed[4];        /* HTREE hash seed */
191:     __u8     s_def_hash_version;     /* Default hash version to use */
192:     __u8     s_reserved_char_pad;
193:     __u16    s_reserved_word_pad;
194:     __u32    s_default_mount_opts;
195:     __u32    s_first_meta_bg;       /* First metablock block group */
196:     __u32    s_reserved[190];       /* Padding to the end of the block */
197: };
198:
199: /*
200:  * Structure of a directory entry
201:  */

```

include/fiwix/fs_ext2.h

Page 4/4

```

202: #define EXT2_NAME_LEN 255
203:
204: struct ext2_dir_entry {
205:     __u32    inode;           /* Inode number */
206:     __u16    rec_len;        /* Directory entry length */
207:     __u16    name_len;       /* Name length */
208:     char     name[EXT2_NAME_LEN]; /* File name */
209: };
210:
211: /*
212:  * The new version of the directory entry. Since EXT2 structures are
213:  * stored in intel byte order, and the name_len field could never be
214:  * bigger than 255 chars, it's safe to reclaim the extra byte for the
215:  * file_type field.
216:  */
217: struct ext2_dir_entry_2 {
218:     __u32    inode;           /* Inode number */
219:     __u16    rec_len;        /* Directory entry length */
220:     __u8     name_len;       /* Name length */
221:     __u8     file_type;
222:     char     name[EXT2_NAME_LEN]; /* File name */
223: };
224:
225: /*
226:  * Ext2 directory file types. Only the low 3 bits are used. The
227:  * other bits are reserved for now.
228:  */
229: #define EXT2_FT_UNKNOWN      0
230: #define EXT2_FT_REG_FILE    1
231: #define EXT2_FT_DIR         2
232: #define EXT2_FT_CHRDEV      3
233: #define EXT2_FT_BLKDEV      4
234: #define EXT2_FT_FIFO        5
235: #define EXT2_FT_SOCK        6
236: #define EXT2_FT_SYMLINK     7
237:
238: /* superblock in memory */
239: struct ext2_sb_info {
240:     unsigned int desc_per_block;
241:     unsigned int block_groups;
242:     struct ext2_super_block sb;
243: };
244:
245: /* inode in memory */
246: struct ext2_i_info {
247:     __u32    i_data[EXT2_N_BLOCKS]; /* Pointers to blocks */
248:     __u32    i_dtime;
249: };
250:
251: #endif /* _FIWIX_FS_EXT2_H */

```

include/fiwix/fs.h

Page 1/4

```

1: /*
2:  * fiwix/include/fiwix/fs.h
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_FS_H
9: #define _FIWIX_FS_H
10:
11: #include <fiwix/config.h>
12: #include <fiwix/types.h>
13:
14: #define CHECK_UFD(ufd)                                     \
15: {                                                         \
16:     if((ufd) > OPEN_MAX || current->fd[(ufd)] == 0) {   \
17:         return -EBADF;                                   \
18:     }                                                     \
19: }                                                         \
20:
21: struct fd {
22:     struct inode *inode;           /* file inode */
23:     unsigned short int flags;     /* flags */
24:     unsigned short int count;     /* number of opened instances */
25:     __off_t offset;              /* r/w pointer position */
26: };
27:
28: #include <fiwix/statfs.h>
29: #include <fiwix/limits.h>
30: #include <fiwix/process.h>
31: #include <fiwix/dirent.h>
32: #include <fiwix/fs_minix.h>
33: #include <fiwix/fs_ext2.h>
34: #include <fiwix/fs_pipe.h>
35: #include <fiwix/fs_iso9660.h>
36: #include <fiwix/fs_proc.h>
37:
38: #define BPS 512 /* bytes per sector */
39: #define BLKSIZE_1K 1024 /* 1KB block size */
40: #define BLKSIZE_2K 2048 /* 2KB block size */
41: #define SUPERBLOCK 1 /* block 1 is for superblock */
42:
43: #define MAJOR(dev) (((__dev_t) (dev)) >> 8)
44: #define MINOR(dev) (((__dev_t) (dev)) & 0xFF)
45: #define MKDEV(major, minor) (((major) << 8) | (minor))
46:
47: /* filesystem independent mount-flags */
48: #define MS_RDONLY 1 /* mount read-only */
49: #define MS_REMOUNT 32 /* alter flags of a mounted FS */
50:
51: /* old magic mount flag and mask */
52: #define MS_MGC_VAL 0xC0ED0000
53: #define MS_MGC_MSK 0xFFFF0000
54:
55: #define IS_RDONLY_FS(inode) (((inode)->sb) && ((inode)->sb->flags & MS_RDONLY))
56:
57: #define FOLLOW_LINKS 1
58: #define MAX_SYMLINKS 8 /* this prevents infinite loops in symlinks */
59:
60: #define SEEK_SET 0
61: #define SEEK_CUR 1
62: #define SEEK_END 2
63:
64: #define FOR_READING 0
65: #define FOR_WRITING 1
66:
67: #define VERIFY_READ 1

```

include/fiwix/fs.h

Page 2/4

```

68: #define VERIFY_WRITE      2
69:
70: #define SEL_R              1
71: #define SEL_W              2
72: #define SEL_E              4
73:
74: struct inode {
75:     __mode_t      i_mode;        /* file mode */
76:     __uid_t       i_uid;        /* owner uid */
77:     __size_t      i_size;       /* size in bytes */
78:     __u32         i_atime;      /* access time */
79:     __u32         i_ctime;      /* creation time */
80:     __u32         i_mtime;      /* modification time */
81:     __gid_t       i_gid;        /* group id */
82:     __nlink_t     i_nlink;      /* links count */
83:     __blk_t       i_blocks;     /* blocks count */
84:     __u32         i_flags;      /* file flags */
85:     unsigned char locked;
86:     unsigned char dirty;        /* 1 = delayed write */
87:     struct inode *mount_point;
88:     __dev_t       dev;
89:     __ino_t       inode;
90:     __s16         count;
91:     __dev_t       rdev;
92:     struct fs_operations *fsop;
93:     struct superblock *sb;
94:     struct inode *prev_hash;
95:     struct inode *next_hash;
96:     struct inode *prev_free;
97:     struct inode *next_free;
98:     union {
99:         struct minix_i_info minix;
100:        struct ext2_i_info ext2;
101:        struct pipefs_inode pipefs;
102:        struct iso9660_inode iso9660;
103:        struct procfs_inode procfs;
104:    } u;
105: };
106: extern struct inode *inode_table;
107: extern struct inode **inode_hash_table;
108:
109: /* values to be determined during system startup */
110: extern unsigned int inode_table_size;        /* size in bytes */
111: extern unsigned int inode_hash_table_size;  /* size in bytes */
112: extern unsigned int fd_table_size;          /* size in bytes */
113:
114: extern struct fd *fd_table;
115:
116: struct superblock {
117:     __dev_t dev;
118:     unsigned char locked;
119:     unsigned char wanted;
120:     struct inode *root;        /* root inode of mounted fs */
121:     struct inode *dir;        /* inode on which the fs was mounted */
122:     unsigned int flags;
123:     unsigned char dirty;      /* 1 = delayed write */
124:     struct fs_operations *fsop;
125:     __u32 s_blocksize;
126:     union {
127:         struct minix_sb_info minix;
128:         struct ext2_sb_info ext2;
129:         struct iso9660_sb_info iso9660;
130:     } u;
131: };
132:
133:
134: #define FSOP_REQUIRES_DEV      1        /* requires a block device */

```

include/fiwix/fs.h

Page 3/4

```

135: #define FSOP_KERN_MOUNT          2          /* mounted by kernel */
136:
137: struct fs_operations {
138:     int flags;
139:     int fsdev;                    /* internal filesystem (nodev) */
140:
141: /* file operations */
142:     int (*open)(struct inode *, struct fd *);
143:     int (*close)(struct inode *, struct fd *);
144:     int (*read)(struct inode *, struct fd *, char *, __size_t);
145:     int (*write)(struct inode *, struct fd *, const char *, __size_t);
146:     int (*ioctl)(struct inode *, int, unsigned long int);
147:     int (*lseek)(struct inode *, __off_t);
148:     int (*readdir)(struct inode *, struct fd *, struct dirent *, unsigned in
t);
149:     int (*mmap)(struct inode *, struct vma *);
150:     int (*select)(struct inode *, int);
151:
152: /* inode operations */
153:     int (*readlink)(struct inode *, char *, __size_t);
154:     int (*followlink)(struct inode *, struct inode *, struct inode **);
155:     int (*bmap)(struct inode *, __off_t, int);
156:     int (*lookup)(const char *, struct inode *, struct inode **);
157:     int (*rmdir)(struct inode *, struct inode *);
158:     int (*link)(struct inode *, struct inode *, char *);
159:     int (*unlink)(struct inode *, struct inode *, char *);
160:     int (*symlink)(struct inode *, char *, char *);
161:     int (*mkdir)(struct inode *, char *, __mode_t);
162:     int (*mknod)(struct inode *, char *, __mode_t, __dev_t);
163:     int (*truncate)(struct inode *, __off_t);
164:     int (*create)(struct inode *, char *, __mode_t, struct inode **);
165:     int (*rename)(struct inode *, struct inode *, struct inode *, struct ino
de *, char *, char *);
166:
167: /* block device I/O operations */
168:     int (*read_block)(__dev_t, __blk_t, char *, int);
169:     int (*write_block)(__dev_t, __blk_t, char *, int);
170:
171: /* superblock operations */
172:     int (*read_inode)(struct inode *);
173:     int (*write_inode)(struct inode *);
174:     int (*ialloc)(struct inode *, int);
175:     void (*ifree)(struct inode *);
176:     void (*statfs)(struct superblock *, struct statfs *);
177:     int (*read_superblock)(__dev_t, struct superblock *);
178:     int (*remount_fs)(struct superblock *, int);
179:     int (*write_superblock)(struct superblock *);
180:     void (*release_superblock)(struct superblock *);
181: };
182:
183: extern struct fs_operations def_chr_fsop;
184: extern struct fs_operations def_blk_fsop;
185:
186: /* fs_minix.h prototypes */
187: extern struct fs_operations minix_fsop;
188: extern struct fs_operations minix_file_fsop;
189: extern struct fs_operations minix_dir_fsop;
190: extern struct fs_operations minix_symlink_fsop;
191: extern int minix_count_free_inodes(struct superblock *);
192: extern int minix_count_free_blocks(struct superblock *);
193: extern int minix_find_first_zero(struct superblock *, __blk_t, int, int);
194: extern int minix_change_bit(int, struct superblock *, int, int);
195: extern int minix_balloc(struct superblock *);
196: extern void minix_bfree(struct superblock *, int);
197:
198: /* fs_ext2.h prototypes */
199: extern struct fs_operations ext2_fsop;

```

include/fiwix/fs.h

Page 4/4

```

200: extern struct fs_operations ext2_file_fsop;
201: extern struct fs_operations ext2_dir_fsop;
202: extern struct fs_operations ext2_symlink_fsop;
203: extern int ext2_balloc(struct superblock *);
204: extern void ext2_bfree(struct superblock *, int);
205:
206: /* fs_proc.h prototypes */
207: extern struct fs_operations procfs_fsop;
208: extern struct fs_operations procfs_file_fsop;
209: extern struct fs_operations procfs_dir_fsop;
210: extern struct fs_operations procfs_symlink_fsop;
211: struct procfs_dir_entry * get_procfs_by_inode(struct inode *);
212:
213: /* fs_iso9660.h prototypes */
214: extern int isonum_711(char *);
215: extern int isonum_723(char *);
216: extern int isonum_731(char *);
217: extern int isonum_733(char *);
218: extern unsigned long int isodate(char *);
219: extern int iso9660_cleanfilename(char *, int);
220: extern struct fs_operations iso9660_fsop;
221: extern struct fs_operations iso9660_file_fsop;
222: extern struct fs_operations iso9660_dir_fsop;
223: extern struct fs_operations iso9660_symlink_fsop;
224: void check_rrip_inode(struct iso9660_directory_record *, struct inode *);
225: int get_rrip_filename(struct iso9660_directory_record *, struct inode *, char *)
;
226: int get_rrip_symlink(struct inode *, char *);
227:
228:
229: /* generic VFS function prototypes */
230: void inode_lock(struct inode *);
231: void inode_unlock(struct inode *);
232: struct inode * ialloc(struct superblock *, int);
233: struct inode * iget(struct superblock *, __ino_t);
234: int bmap(struct inode *, __off_t, int);
235: int check_fs_busy(__dev_t, struct inode *);
236: void iput(struct inode *);
237: void sync_inodes(__dev_t);
238: void invalidate_inodes(__dev_t);
239: void inode_init(void);
240:
241: int parse_namei(char *, struct inode *, struct inode **, struct inode **, int);
242: int namei(char *, struct inode **, struct inode **, int);
243:
244: void superblock_lock(struct superblock *);
245: void superblock_unlock(struct superblock *);
246: struct mount * get_free_mount_point(__dev_t);
247: void release_mount_point(struct mount *);
248: struct mount * get_mount_point(struct inode *);
249:
250: int get_new_fd(struct inode *);
251: void release_fd(unsigned int);
252: void fd_init(void);
253:
254: void free_name(const char *);
255: int malloc_name(const char *, char **);
256: int check_user_permission(struct inode *);
257: int check_group(struct inode *);
258: int check_user_area(int, const void *, unsigned int);
259: int check_permission(int, struct inode *);
260:
261: int do_select(int, fd_set *, fd_set *, fd_set *, fd_set *, fd_set *, fd_set *);
262:
263: #endif /* _FIWIX_FS_H */

```

include/fiwix/fs_iso9660.h

Page 1/4

```

1: /*
2:  * fiwix/include/fiwix/fs_iso9660.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_FS_ISO9660_H
9: #define _FIWIX_FS_ISO9660_H
10:
11: #include <fiwix/types.h>
12: #include <fiwix/limits.h>
13:
14: #define ISO9660_SUPERBLOCK      16      /* ISO9660 superblock is in block 16 */
15: #define ISO9660_STANDARD_ID    "CD001" /* standard identification */
16: #define ISO9660_SUPER_MAGIC    0x9660
17:
18: #define ISO9660_VD_BOOT        0
19: #define ISO9660_VD_PRIMARY     1
20: #define ISO9660_VD_SUPPLEMENTARY 2
21: #define ISO9660_VD_PARTITION  3
22: #define ISO9660_VD_END        255
23:
24: #define ISODCL(from, to)      ((to - from) + 1)      /* descriptor length */
25:
26: #define ISO9660_MAX_VD       10      /* maximum number of VD per CDROM */
27:
28: /* inodes will have their directory block and their offset packed as follows:
29:  * 7FF7FF
30:  * \-/\-/\
31:  * ^ ^
32:  * | +----- offset value          (11bit entries)
33:  * +----- directory block where to find it (11bit entries)
34:  */
35: #define ISO9660_INODE_BITS    11      /* FIXME: it could be greater (16bit) */
36: #define ISO9660_INODE_MASK    0x7FF
37:
38: #define ISO9660_FILE_NOTEXIST  0x01   /* file shouldn't exists for the user */
39: #define ISO9660_FILE_ISDIR    0x02   /* is a directory */
40: #define ISO9660_FILE_ISASSOC  0x04   /* associated file */
41: #define ISO9660_FILE_HASRECFMT 0x08   /* has a record format */
42: #define ISO9660_FILE_HASOWNER 0x10   /* has owner and group defined */
43: #define ISO9660_FILE_RESERVED5 0x20   /* reserved */
44: #define ISO9660_FILE_RESERVED6 0x40   /* reserved */
45: #define ISO9660_FILE_ISMULTIEXT 0x80  /* has more directory records */
46:
47: #define SP_MAGIC1              0xBE
48: #define SP_MAGIC2              0xEF
49: #define GET_SIG(s1, s2)       ((s1 << 8) | s2)
50:
51: #define SL_CURRENT             0x02
52: #define SL_PARENT              0x04
53: #define SL_ROOT                0x08
54:
55: #define TF_CREATION            0x01
56: #define TF_MODIFY              0x02
57: #define TF_ACCESS              0x04
58: #define TF_ATTRIBUTES         0x08
59: #define TF_BACKUP              0x10
60: #define TF_EXPIRATION         0x20
61: #define TF_EFFECTIVE          0x40
62: #define TF_LONG_FORM          0x80
63:
64: #define NM_CONTINUE            0
65: #define NM_CURRENT             1
66: #define NM_PARENT              2
67:

```


include/fiwix/fs_iso9660.h

Page 2/4

```

68: /* formerly Primary Volume Descriptor */
69: struct iso9660_super_block {
70:     char type [ISODCL( 1, 1)]; /* 7.1.1 */
71:     char id [ISODCL( 2, 6)];
72:     char version [ISODCL( 7, 7)]; /* 7.1.1 */
73:     char unused1 [ISODCL( 8, 8)];
74:     char system_id [ISODCL( 9, 40)]; /* a-chars */
75:     char volume_id [ISODCL( 41, 72)]; /* d-chars */
76:     char unused2 [ISODCL( 73, 80)];
77:     char volume_space_size [ISODCL( 81, 88)]; /* 7.3.3 */
78:     char unused3 [ISODCL( 89, 120)];
79:     char volume_set_size [ISODCL(121, 124)]; /* 7.2.3 */
80:     char volume_sequence_number [ISODCL(125, 128)]; /* 7.2.3 */
81:     char logical_block_size [ISODCL(129, 132)]; /* 7.2.3 */
82:     char path_table_size [ISODCL(133, 140)]; /* 7.3.3 */
83:     char type_l_path_table [ISODCL(141, 144)]; /* 7.3.1 */
84:     char opt_type_l_path_table [ISODCL(145, 148)]; /* 7.3.1 */
85:     char type_m_path_table [ISODCL(149, 152)]; /* 7.3.2 */
86:     char opt_type_m_path_table [ISODCL(153, 156)]; /* 7.3.2 */
87:     char root_directory_record [ISODCL(157, 190)]; /* 9.1 */
88:     char volume_set_id [ISODCL(191, 318)]; /* d-chars */
89:     char publisher_id [ISODCL(319, 446)]; /* a-chars */
90:     char preparer_id [ISODCL(447, 574)]; /* a-chars */
91:     char application_id [ISODCL(575, 702)]; /* a-chars */
92:     char copyright_file_id [ISODCL(703, 739)]; /* 7.5 d-chars */
/
93:     char abstract_file_id [ISODCL(740, 776)]; /* 7.5 d-chars */
/
94:     char bibliographic_file_id [ISODCL(777, 813)]; /* 7.5 d-chars */
/
95:     char creation_date [ISODCL(814, 830)]; /* 8.4.26.1 */
96:     char modification_date [ISODCL(831, 847)]; /* 8.4.26.1 */
97:     char expiration_date [ISODCL(848, 864)]; /* 8.4.26.1 */
98:     char effective_date [ISODCL(865, 881)]; /* 8.4.26.1 */
99:     char file_structure_version [ISODCL(882, 882)];
100:    char unused4 [ISODCL(883, 883)];
101:    char application_data [ISODCL(884, 1395)];
102:    char unused5 [ISODCL(1396, 2048)];
103: };
104:
105: struct iso9660_directory_record
106: {
107:     char length [ISODCL( 1, 1)]; /* 7.1.1 */
108:     char ext_attr_length [ISODCL( 2, 2)]; /* 7.1.1 */
109:     char extent [ISODCL( 3, 10)]; /* 7.3.3 */
110:     char size [ISODCL(11, 18)]; /* 7.3.3 */
111:     char date [ISODCL(19, 25)]; /* 7 by 7.1.1 */
112:     char flags [ISODCL(26, 26)];
113:     char file_unit_size [ISODCL(27, 27)]; /* 7.1.1 */
114:     char interleave [ISODCL(28, 28)]; /* 7.1.1 */
115:     char volume_sequence_number [ISODCL(29, 32)]; /* 7.2.3 */
116:     char name_len [ISODCL(33, 33)]; /* 7.1.1 */
117:     char name[0];
118: };
119:
120: struct iso9660_pathtable_record
121: {
122:     char length [ISODCL( 1, 1)]; /* 7.1.1 */
123:     char ext_attr_length [ISODCL( 2, 2)]; /* 7.1.1 */
124:     char extent [ISODCL( 3, 6)]; /* 7.3 */
125:     char parent [ISODCL( 7, 8)]; /* 7.2 */
126:     char name[0];
127: };
128:
129: struct susp_sp {
130:     unsigned char magic[2];
131:     char len_skip;

```

```

132: };
133:
134: struct susp_ce {
135:     char block[8];
136:     char offset[8];
137:     char size[8];
138: };
139:
140: struct susp_er {
141:     char len_id;
142:     char len_des;
143:     char len_src;
144:     char ext_ver;
145:     char data[0];
146: };
147:
148: struct rrip_px {
149:     char mode[8];
150:     char nlink[8];
151:     char uid[8];
152:     char gid[8];
153:     char sn[8];
154: };
155:
156: struct rrip_pn {
157:     char dev_h[8];
158:     char dev_l[8];
159: };
160:
161: struct rrip_sl_component {
162:     unsigned char flags;
163:     unsigned char len;
164:     char name[0];
165: };
166:
167: struct rrip_sl {
168:     unsigned char flags;
169:     struct rrip_sl_component area;
170: };
171:
172: struct rrip_nm {
173:     unsigned char flags;
174:     char name[0];
175: };
176:
177: struct rrip_tf_timestamp {
178:     char time[7];           /* assumes LONG_FORM bit always set to zero */
179: };
180:
181: struct rrip_tf {
182:     char flags;
183:     struct rrip_tf_timestamp times[0];
184: };
185:
186: struct susp_rrip {
187:     char signature[2];
188:     unsigned char len;
189:     unsigned char version;
190:     union {
191:         struct susp_sp sp;
192:         struct susp_ce ce;
193:         struct susp_er er;
194:         struct rrip_px px;
195:         struct rrip_pn pn;
196:         struct rrip_sl sl;
197:         struct rrip_nm nm;
198:         struct rrip_tf tf;

```

include/fiwix/fs_iso9660.h

Page 4/4

```
199:         } u;
200: };
201:
202: struct iso9660_inode {
203:     __blk_t i_extent;
204:     struct inode *i_parent;           /* inode of its parent directory */
205: };
206:
207: struct iso9660_sb_info {
208:     __u32 s_root_inode;
209:     char *pathtable_raw;
210:     struct iso9660_pathtable_record **pathtable;
211:     int paths;
212:     unsigned char rrip;
213:     struct iso9660_super_block *sb;
214: };
215:
216: #endif /* _FIWIX_FS_ISO9660_H */
```

include/fiwix/fs_minix.h

Page 1/2

```

1: /*
2:  * fiwix/include/fiwix/fs_minix.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_FS_MINIX_H
9: #define _FIWIX_FS_MINIX_H
10:
11: #include <fiwix/types.h>
12: #include <fiwix/limits.h>
13:
14: #define MINIX_ROOT_INO          1          /* root inode */
15:
16: #define MINIX_SUPER_MAGIC      0x137F    /* Minix v1, 14 char names */
17: #define MINIX_SUPER_MAGIC2    0x138F    /* Minix v1, 30 char names */
18: #define MINIX2_SUPER_MAGIC    0x2468    /* Minix v2, 14 char names */
19: #define MINIX2_SUPER_MAGIC2   0x2478    /* Minix v2, 30 char names */
20:
21: #define MINIX_VALID_FS        1          /* clean filesystem */
22: #define MINIX_ERROR_FS        2          /* needs fsck */
23:
24: #define CLEAR_BIT              0
25: #define SET_BIT                 1
26:
27: #define V1_MAX_BITMAP_BLOCKS   8          /* 64MB filesystem size */
28: #define V2_MAX_BITMAP_BLOCKS  128         /* 1GB filesystem size */
29:
30: /*
31:  * Minix (v1 and v2) file system physical layout:
32:  *
33:  *      +-----+
34:  *      |                size in blocks of BLKSIZE_1K (1024 bytes)                |
35:  * +-----+-----+
36:  * | block 0          | 1 |
37:  * +-----+-----+
38:  * | superblock      | 1 |
39:  * +-----+-----+
40:  * | inode map       | number of inodes / (BLKSIZE_1K * 8) |
41:  * +-----+-----+
42:  * | zone map        | number of zones / (BLKSIZE_1K * 8) |
43:  * +-----+-----+
44:  * | inode table    | ((32 or 64) * number of inodes) / BLKSIZE_1K |
45:  * +-----+-----+
46:  * | data zones     | ... |
47:  * +-----+-----+
48:  *
49:  * The implementation of this filesystem in Fiwix might have slow disk writes
50:  * because I don't keep in memory the superblock, nor the blocks of the inode
51:  * map nor the blocks of the zone map. Keeping them in memory would be a waste
52:  * of 137KB per each mounted v2 filesystem (1GB of size).
53:  *
54:  * - superblock      -> 1KB
55:  * - inode map       -> 8KB (1KB (8192 bits) x 8 = 65536 inodes)
56:  * - zone map        -> 128KB (1KB (8192 bits) x 128 = 1048576 1k-blocks)
57:  *
58:  */
59:
60: struct minix_super_block {
61:     __u16 s_ninodes;          /* number of inodes */
62:     __u16 s_nzones;          /* number of data zones */
63:     __u16 s_imap_blocks;     /* blocks used by inode bitmap */
64:     __u16 s_zmap_blocks;     /* blocks used by zone bitmap */
65:     __u16 s_firstdatazone;   /* number of first data zone */
66:     __u16 s_log_zone_size;   /* 1024 << s_log_zone_size */
67:     __u32 s_max_size;        /* maximum file size (in bytes) */

```

include/fiwix/fs_minix.h

Page 2/2

```

68:         __u16 s_magic;           /* magic number */
69:         __u16 s_state;           /* filesystem state */
70:         __u32 s_zones;          /* number of data zones (for v2 only) */
71: };
72:
73: struct minix_inode {
74:     __u16 i_mode;
75:     __u16 i_uid;
76:     __u32 i_size;
77:     __u32 i_time;
78:     __u8 i_gid;
79:     __u8 i_nlinks;
80:     __u16 i_zone[9];
81: };
82:
83: struct minix2_inode {
84:     __u16 i_mode;
85:     __u16 i_nlink;
86:     __u16 i_uid;
87:     __u16 i_gid;
88:     __u32 i_size;
89:     __u32 i_atime;
90:     __u32 i_mtime;
91:     __u32 i_ctime;
92:     __u32 i_zone[10];
93: };
94:
95: struct minix_dir_entry {
96:     __u16 inode;
97:     char name[0];
98: };
99:
100: /* super block in memory */
101: struct minix_sb_info {
102:     unsigned char namelen;
103:     unsigned char dirsize;
104:     unsigned short int version;
105:     unsigned int nzones;
106:     struct minix_super_block sb;
107: };
108:
109: /* inode in memory */
110: struct minix_i_info {
111:     union {
112:         __u16 i1_zone[9];
113:         __u32 i2_zone[10];
114:     } u;
115: };
116:
117: int v1_minix_read_inode(struct inode *);
118: int v1_minix_write_inode(struct inode *);
119: int v1_minix_ialloc(struct inode *, int);
120: void v1_minix_ifree(struct inode *);
121: int v1_minix_bmap(struct inode *, __off_t, int);
122: int v1_minix_truncate(struct inode *, __off_t);
123:
124: int v2_minix_read_inode(struct inode *);
125: int v2_minix_write_inode(struct inode *);
126: int v2_minix_ialloc(struct inode *, int);
127: void v2_minix_ifree(struct inode *);
128: int v2_minix_bmap(struct inode *, __off_t, int);
129: int v2_minix_truncate(struct inode *, __off_t);
130:
131: #endif /* _FIWIX_FS_MINIX_H */

```

include/fiwix/fs_pipe.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/fs_pipe.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_FS_PIPE_H
9: #define _FIWIX_FS_PIPE_H
10:
11: #define PIPE_DEV          0xFFF0          /* special device number for nodev fs */
12:
13: extern struct fs_operations pipefs_fsop;
14:
15: struct pipefs_inode {
16:     char *i_data;          /* buffer */
17:     unsigned int i_readoff; /* offset for reads */
18:     unsigned int i_writeoff; /* offset for writes */
19:     unsigned int i_readers; /* number of readers */
20:     unsigned int i_writers; /* number of writers */
21: };
22:
23: #endif /* _FIWIX_FS_PIPE_H */
```

include/fiwix/fs_proc.h

Page 1/2

```

1: /*
2:  * fiwix/include/fiwix/fs_proc.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_FS_PROC_H
9: #define _FIWIX_FS_PROC_H
10:
11: #include <fiwix/types.h>
12:
13: #define PROC_DEV          0xFFF1 /* special device number for nodev fs */
14: #define PROC_ROOT_INO    1      /* root inode */
15: #define PROC_SUPER_MAGIC 0x9FA0 /* same as in Linux */
16:
17: #define PROC_PID_INO     0x40000000 /* base for PID inodes */
18: #define PROC_PID_LEV    1        /* array level for PID */
19:
20: #define PROC_ARRAY_ENTRIES 20
21:
22: enum pid_dir_inodes {
23:     PROC_PID_FD = PROC_PID_INO + 1001,
24:     PROC_PID_CMDLINE,
25:     PROC_PID_CWD,
26:     PROC_PID_ENVIRON,
27:     PROC_PID_EXE,
28:     PROC_PID_MAPS,
29:     PROC_PID_MOUNTINFO,
30:     PROC_PID_ROOT,
31:     PROC_PID_STAT,
32:     PROC_PID_STATM,
33:     PROC_PID_STATUS
34: };
35:
36: struct procfs_inode {
37:     unsigned int i_lev; /* array level (directory depth) */
38: };
39:
40: struct procfs_dir_entry {
41:     __ino_t inode;
42:     __mode_t mode;
43:     __nlink_t nlink;
44:     int lev; /* array level (directory depth) */
45:     unsigned short int name_len;
46:     char *name;
47:     int (*data_fn)(char *, __pid_t);
48: };
49:
50: extern struct procfs_dir_entry procfs_array[][PROC_ARRAY_ENTRIES + 1];
51:
52: int data_proc_cmdline(char *, __pid_t);
53: int data_proc_cpuinfo(char *, __pid_t);
54: int data_proc_devices(char *, __pid_t);
55: int data_proc_dma(char *, __pid_t);
56: int data_proc_filesystems(char *, __pid_t);
57: int data_proc_interrupts(char *, __pid_t);
58: int data_proc_loadavg(char *, __pid_t);
59: int data_proc_locks(char *, __pid_t);
60: int data_proc_meminfo(char *, __pid_t);
61: int data_proc_mounts(char *, __pid_t);
62: int data_proc_partitions(char *, __pid_t);
63: int data_proc_rtc(char *, __pid_t);
64: int data_proc_self(char *, __pid_t);
65: int data_proc_stat(char *, __pid_t);
66: int data_proc_uptime(char *, __pid_t);
67: int data_proc_fullversion(char *, __pid_t);

```

include/fiwix/fs_proc.h

Page 2/2

```
68: int data_proc_domainname(char *, __pid_t);
69: int data_proc_filemax(char *, __pid_t);
70: int data_proc_filenr(char *, __pid_t);
71: int data_proc_hostname(char *, __pid_t);
72: int data_proc_inodemax(char *, __pid_t);
73: int data_proc_inodenr(char *, __pid_t);
74: int data_proc_osrelease(char *, __pid_t);
75: int data_proc_ostype(char *, __pid_t);
76: int data_proc_version(char *, __pid_t);
77:
78: /* PID related functions */
79: int data_proc_pid_fd(char *, __pid_t);
80: int data_proc_pid_cmdline(char *, __pid_t);
81: int data_proc_pid_cwd(char *, __pid_t);
82: int data_proc_pid_environ(char *, __pid_t);
83: int data_proc_pid_exe(char *, __pid_t);
84: int data_proc_pid_maps(char *, __pid_t);
85: int data_proc_pid_mountinfo(char *, __pid_t);
86: int data_proc_pid_root(char *, __pid_t);
87: int data_proc_pid_stat(char *, __pid_t);
88: int data_proc_pid_statm(char *, __pid_t);
89: int data_proc_pid_status(char *, __pid_t);
90:
91: #endif /* _FIWIX_FS_PROC_H */
```


include/fiwix/i386elf.h

Page 1/5

```

1: /*
2:  * fiwix/include/fiwix/i386elf.h
3:  */
4:
5: #ifndef _FIWIX_ELF_H
6: #define _FIWIX_ELF_H
7:
8: typedef unsigned long   Elf32_Addr;
9: typedef unsigned short Elf32_Half;
10: typedef unsigned long  Elf32_Off;
11: typedef long           Elf32_Sword;
12: typedef unsigned long  Elf32_Word;
13:
14: #define ELFMAG0          0x7f          /* EI_MAG */
15: #define ELFMAG1          'E'
16: #define ELFMAG2          'L'
17: #define ELFMAG3          'F'
18: #define ELFMAG           "\177ELF"
19: #define SELFMAG          4
20:
21: #define EI_NIDENT        16
22:
23: typedef struct elf32_hdr{
24:     unsigned char e_ident[EI_NIDENT]; /* ELF "magic number" */
25:     Elf32_Half    e_type;             /* File type */
26:     Elf32_Half    e_machine;         /* Target machine */
27:     Elf32_Word    e_version;         /* File version */
28:     Elf32_Addr    e_entry;           /* Entry point virtual address */
29:     Elf32_Off     e_phoff;           /* Program header table file offset */
30:     Elf32_Off     e_shoff;           /* Section header table file offset */
31:     Elf32_Word    e_flags;           /* File flags */
32:     Elf32_Half    e_ehsize;          /* Sizeof Ehdr (ELF header) */
33:     Elf32_Half    e_phentsize;       /* Sizeof Phdr (Program header) */
34:     Elf32_Half    e_phnum;           /* Number Phdrs (Program header) */
35:     Elf32_Half    e_shentsize;       /* Sizeof Shdr (Section header) */
36:     Elf32_Half    e_shnum;           /* Number Shdrs (Section header) */
37:     Elf32_Half    e_shstrndx;        /* Shdr string index */
38: } Elf32_Ehdr;
39:
40: #define EI_MAG0          0           /* e_ident[] indexes */
41: #define EI_MAG1          1
42: #define EI_MAG2          2
43: #define EI_MAG3          3
44: #define EI_CLASS         4
45: #define EI_DATA          5
46: #define EI_VERSION       6
47: #define EI_PAD           7
48:
49: #define ELFCLASSNONE     0           /* EI_CLASS */
50: #define ELFCLASS32       1
51: #define ELFCLASS64       2
52: #define ELFCLASSNUM      3
53:
54: #define ELFDATANONE     0           /* e_ident[EI_DATA] */
55: #define ELFDATA2LSB     1
56: #define ELFDATA2MSB     2
57: #define ELFDATANUM      3
58:
59: /* ELF file types */
60: #define ET_NONE          0
61: #define ET_REL           1
62: #define ET_EXEC          2
63: #define ET_DYN           3
64: #define ET_CORE          4
65: #define ET_LOPROC        5
66: #define ET_HIPROC        6
67:

```

include/fiwix/i386elf.h

Page 2/5

```

68: #define EM_386      3
69:
70: #define EV_NONE      0          /* e_version, EI_VERSION */
71: #define EV_CURRENT   1
72: #define EV_NUM       2
73:
74: typedef struct elf32_phdr{
75:     Elf32_Word      p_type;      /* Entry type */
76:     Elf32_Off       p_offset;    /* File offset */
77:     Elf32_Addr      p_vaddr;    /* Virtual address */
78:     Elf32_Addr      p_paddr;    /* Physical address */
79:     Elf32_Word      p_filesz;   /* File size */
80:     Elf32_Word      p_memsz;    /* Memory size */
81:     Elf32_Word      p_flags;    /* Entry flags */
82:     Elf32_Word      p_align;    /* Memory & file alignment */
83: } Elf32_Phdr;
84:
85: /* segment types stored in the image headers */
86: #define PT_NULL      0
87: #define PT_LOAD      1
88: #define PT_DYNAMIC   2
89: #define PT_INTERP    3
90: #define PT_NOTE      4
91: #define PT_SHLIB     5
92: #define PT_PHDR      6
93: #define PT_NUM       7
94: #define PT_LOPROC    0x70000000
95: #define PT_HIPROC    0x7fffffff
96:
97: /* permission types on sections in the program header, p_flags. */
98: #define PF_R         0x4
99: #define PF_W         0x2
100: #define PF_X         0x1
101:
102: #define PF_MASKPROC 0xf0000000
103:
104: typedef struct {
105:     Elf32_Word      sh_name;     /* Section name, index in string tbl */
106:     Elf32_Word      sh_type;     /* Type of section */
107:     Elf32_Word      sh_flags;    /* Miscellaneous section attributes */
108:     Elf32_Addr      sh_addr;     /* Section virtual addr at execution */
109:     Elf32_Off       sh_offset;   /* Section file offset */
110:     Elf32_Word      sh_size;     /* Size of section in bytes */
111:     Elf32_Word      sh_link;     /* Index of another section */
112:     Elf32_Word      sh_info;     /* Additional section information */
113:     Elf32_Word      sh_addralign; /* Section alignment */
114:     Elf32_Word      sh_entsize;  /* Entry size if section holds table */
115: } Elf32_Shdr;
116:
117: /* sh_type */
118: #define SHT_NULL      0
119: #define SHT_PROGBITS  1
120: #define SHT_SYMTAB    2
121: #define SHT_STRTAB    3
122: #define SHT_RELA      4
123: #define SHT_HASH      5
124: #define SHT_DYNAMIC   6
125: #define SHT_NOTE      7
126: #define SHT_NOBITS    8
127: #define SHT_REL       9
128: #define SHT_SHLIB     10
129: #define SHT_DYNSYM    11
130: #define SHT_NUM       12
131:
132: #define SHT_LOPROC    0x70000000
133: #define SHT_HIPROC    0x7fffffff
134: #define SHT_LOUSER    0x80000000

```

include/fiwix/i386elf.h

Page 3/5

```

135: #define SHT_HIUSER      0xffffffff
136:
137: /* sh_flags */
138: #define SHF_WRITE       0x1
139: #define SHF_ALLOC       0x2
140: #define SHF_EXECINSTR   0x4
141:
142: /* special section indexes */
143: #define SHN_UNDEF       0
144: #define SHN_LORESERVE   0xff00
145: #define SHN_ABS         0xffff1
146: #define SHN_COMMON      0xffff2
147: #define SHN_HIRESERVE   0xfffff
148: #define SHN_LOPROC      0xff00
149: #define SHN_HIPROC      0xff1f
150:
151: typedef struct elf32_sym{
152:     Elf32_Word    st_name;           /* Symbol name, index in string tbl */
153:     Elf32_Addr    st_value;         /* Value of the symbol */
154:     Elf32_Word    st_size;         /* Associated symbol size */
155:     unsigned char st_info;         /* Type and binding attributes */
156:     unsigned char st_other;        /* No defined meaning, 0 */
157:     Elf32_Half    st_shndx;        /* Associated section index */
158: } Elf32_Sym;
159:
160: #define ELF32_ST_BIND(info)        ((info) >> 4)
161: #define ELF32_ST_TYPE(info)        (((unsigned int) info) & 0xf)
162:
163: /* This info is needed when parsing the symbol table */
164: #define STB_LOCAL          0
165: #define STB_GLOBAL        1
166: #define STB_WEAK          2
167: #define STB_NUM           3
168:
169: #define STT_NOTYPE        0
170: #define STT_OBJECT        1
171: #define STT_FUNC          2
172: #define STT_SECTION       3
173: #define STT_FILE          4
174: #define STT_NUM           5
175:
176: typedef struct elf32_rel {
177:     Elf32_Addr    r_offset;         /* Location at which to apply the action */
178:     Elf32_Word    r_info;          /* Index and type of relocation */
179: } Elf32_Rel;
180:
181: typedef struct elf32_rela{
182:     Elf32_Addr    r_offset;         /* Location at which to apply the action */
183:     Elf32_Word    r_info;          /* Index and type of relocation */
184:     Elf32_Sword   r_addend;        /* Constant addend used to compute value */
185: } Elf32_Rela;
186:
187: /* The following are used with relocations */
188: #define ELF32_R_SYM(info)          ((info) >> 8)
189: #define ELF32_R_TYPE(info)         ((info) & 0xff)
190:
191: /* This is the info that is needed to parse the dynamic section of the file */
192: #define DT_NULL                0
193: #define DT_NEEDED               1
194: #define DT_PLTRELSZ            2
195: #define DT_PLTGOT              3
196: #define DT_HASH                 4
197: #define DT_STRTAB               5
198: #define DT_SYMTAB               6
199: #define DT_RELA                 7
200: #define DT_RELASZ              8
201: #define DT_RELAENT              9

```

include/fiwix/i386elf.h

Page 4/5

```

202: #define DT_STRSZ          10
203: #define DT_SYMENT        11
204: #define DT_INIT          12
205: #define DT_FINI          13
206: #define DT_SONAME        14
207: #define DT_RPATH         15
208: #define DT_SYMBOLIC      16
209: #define DT_REL           17
210: #define DT_RELSZ        18
211: #define DT_RELENT       19
212: #define DT_PLTREL       20
213: #define DT_DEBUG        21
214: #define DT_TEXTREL      22
215: #define DT_JMPREL       23
216: #define DT_LOPROC       0x70000000
217: #define DT_HIPROC       0x7fffffff
218:
219: /* Symbolic values for the entries in the auxiliary table
220:    put on the initial stack */
221: #define AT_NULL          0    /* end of vector */
222: #define AT_IGNORE       1    /* entry should be ignored */
223: #define AT_EXECFD       2    /* file descriptor of program */
224: #define AT_PHDR         3    /* program headers for program */
225: #define AT_PHEMT        4    /* size of program header entry */
226: #define AT_PHNUM        5    /* number of program headers */
227: #define AT_PAGESZ       6    /* system page size */
228: #define AT_BASE         7    /* base address of interpreter */
229: #define AT_FLAGS        8    /* flags */
230: #define AT_ENTRY        9    /* entry point of program */
231: #define AT_NOTELF      10    /* program is not ELF */
232: #define AT_UID         11    /* real uid */
233: #define AT_EUID        12    /* effective uid */
234: #define AT_GID         13    /* real gid */
235: #define AT_EGID        14    /* effective gid */
236:
237:
238: typedef struct dynamic{
239:     Elf32_Sword d_tag;          /* entry tabg value */
240:     union{
241:         Elf32_Sword d_val;
242:         Elf32_Addr d_ptr;
243:     } d_un;
244: } Elf32_Dyn;
245:
246: #define R_386_NONE        0
247: #define R_386_32         1
248: #define R_386_PC32       2
249: #define R_386_GOT32      3
250: #define R_386_PLT32      4
251: #define R_386_COPY       5
252: #define R_386_GLOB_DAT   6
253: #define R_386_JMP_SLOT   7
254: #define R_386_RELATIVE   8
255: #define R_386_GOTOFF     9
256: #define R_386_GOTPC     10
257: #define R_386_NUM        11
258:
259: /* Notes used in ET_CORE */
260: #define NT_PRSTATUS      1
261: #define NT_PRFPREG      2
262: #define NT_PRPSINFO     3
263: #define NT_TASKSTRUCT   4
264:
265: /* Note header in a PT_NOTE section */
266: typedef struct elf32_note {
267:     Elf32_Word    n_namesz;    /* Name size */
268:     Elf32_Word    n_descsz;    /* Content size */

```

include/fiwix/i386elf.h

Page 5/5

```
269:   Elf32_Word    n_type;           /* Content type */
270: } Elf32_Nhdr;
271:
272: #define ELF_START_MMAP 0x80000000
273:
274: extern Elf32_Dyn __DYNAMIC [];
275: #define elfhdr      elf32_hdr
276: #define elf_phdr    elf32_phdr
277: #define elf_note    elf32_note
278:
279: #endif /* _FIWIX_ELF_H */
```

include/fiwix/ide_cd.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/ide_cd.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_IDE_CD_H
9: #define _FIWIX_IDE_CD_H
10:
11: #include <fiwix/fs.h>
12:
13: #define IDE_CD_SECTSIZE          BLKSIZE_2K          /* sector size (in bytes) */
14:
15: void ide_cd_timer(unsigned int);
16:
17: int ide_cd_open(struct inode *, struct fd *);
18: int ide_cd_close(struct inode *, struct fd *);
19: int ide_cd_read(__dev_t, __blk_t, char *, int);
20: int ide_cd_ioctl(struct inode *, int, unsigned long int);
21:
22: int ide_cd_init(struct ide *, int);
23:
24: #endif /* _FIWIX_IDE_CD_H */
```

include/fiwix/ide.h

Page 1/5

```

1: /*
2:  * fiwix/include/fiwix/ide.h
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_IDE_H
9: #define _FIWIX_IDE_H
10:
11: #include <fiwix/fs.h>
12: #include <fiwix/part.h>
13: #include <fiwix/sigcontext.h>
14: #include <fiwix/sleep.h>
15:
16: #define IDE0_IRQ          14      /* primary controller interrupt */
17: #define IDE1_IRQ          15      /* secondary controller interrupt */
18:
19: #define IDE0_MAJOR        3       /* 1st controller major number */
20: #define IDE1_MAJOR        22      /* 2nd controller major number */
21: #define IDE_MINORS        4       /* max. minors/partitions per unit */
22: #define IDE_MASTER_MSF    0       /* IDE master minor shift factor */
23: #define IDE_SLAVE_MSF    6       /* IDE slave minor shift factor */
24:
25: #define IDE_PRIMARY        0
26: #define IDE_SECONDARY      1
27: #define IDE_MASTER        0
28: #define IDE_SLAVE        1
29: #define IDE_ATA            0
30: #define IDE_ATAPI          1
31:
32: #define NR_IDE_CTRL        2       /* IDE controllers */
33: #define NR_IDE_DRV        2       /* max. drives per IDE controller */
34:
35: /* controller addresses */
36: #define IDE0_BASE          0x1F0   /* primary controller base addr */
37: #define IDE0_CTRL          0x3F4   /* primary controller control port */
38: #define IDE1_BASE          0x170   /* secondary controller base addr */
39: #define IDE1_CTRL          0x374   /* secondary controller control port */
40:
41: #define IDE_BASE_LEN       7       /* controller address length */
42:
43: #define IDE_RDY_RETR_LONG  50000  /* long delay for fast CPUs */
44: #define IDE_RDY_RETR_SHORT 500    /* short delay for slow CPUs */
45: #define MAX_IDE_ERR        10      /* number of retries */
46: #define MAX_CD_ERR         5       /* number of retries in CDROMs */
47:
48: #define SET_IDE_RDY_RETR(retries) \
49:     if((cpu_table.hz / 1000000) <= 100) { \
50:         retries = IDE_RDY_RETR_SHORT; \
51:     } else { \
52:         retries = IDE_RDY_RETR_LONG; \
53:     }
54:
55: #define WAIT_FOR_IDE        (1 * HZ) /* timeout for hard disk */
56: #define WAIT_FOR_CD         (3 * HZ) /* timeout for cdrom */
57:
58: /* controller registers */
59: #define IDE_DATA            0x0     /* Data Port Register (R/W) */
60: #define IDE_ERROR          0x1     /* Error Register (R) */
61: #define IDE_FEATURES       0x1     /* Features Register (W) */
62: #define IDE_SECCNT         0x2     /* Sector Count Register (R/W) */
63: #define IDE_SECTNUM        0x3     /* Sector Number Register (R/W) */
64: #define IDE_LCYL           0x4     /* Cylinder Low Register (R/W) */
65: #define IDE_HCYL           0x5     /* Cylinder High Register (R/W) */
66: #define IDE_DRVHD          0x6     /* Drive/Head Register (R/W) */
67: #define IDE_STATUS         0x7     /* Status Register (R) */

```

include/fiwix/ide.h

Page 2/5

```

68: #define IDE_COMMAND          0x7      /* Command Register (W) */
69:
70: #define IDE_ALT_STATUS        0x2      /* Alternate Register (R) */
71: #define IDE_DEV_CTRL          0x2      /* Device Control Register (W) */
72:
73: /* error register bits */
74: #define IDE_ERR_AMNF          0x01     /* Address Mark Not Found */
75: #define IDE_ERR_TKONF         0x02     /* Track 0 Not Found */
76: #define IDE_ERR_ABRT          0x04     /* Aborted Command */
77: #define IDE_ERR_MCR           0x08     /* Media Change Registered */
78: #define IDE_ERR_IDNF          0x10     /* Sector ID Field Not Found */
79: #define IDE_ERR_MC            0x20     /* Media Changed */
80: #define IDE_ERR_UNC           0x40     /* Uncorrectable Data Error */
81: #define IDE_ERR_BBK           0x80     /* Bad Block */
82:
83: /* status register bits */
84: #define IDE_STAT_ERR          0x01     /* an error occurred */
85: #define IDE_STAT_SENS         0x02     /* sense data available */
86: #define IDE_STAT_CORR         0x04     /* a correctable error occurred */
87: #define IDE_STAT_DRQ          0x08     /* device is ready to transfer */
88: #define IDE_STAT_DSC          0x10     /* device requests service o intr. */
89: #define IDE_STAT_DWF          0x20     /* drive write fault */
90: #define IDE_STAT_RDY          0x40     /* drive is ready */
91: #define IDE_STAT_BSY          0x80     /* drive is busy */
92:
93: #define IDE_CHS_MODE          0xA0     /* select CHS mode */
94: #define IDE_LBA_MODE          0xE0     /* select LBA mode */
95:
96: /* alternate & device control register bits */
97: #define IDE_DEVCTR_DRQ        0x08     /* Data Request */
98: #define IDE_DEVCTR_NIEN       0x02     /* Disable Interrupt */
99: #define IDE_DEVCTR_SRST       0x04     /* Software Reset */
100:
101: /* ATA commands */
102: #define ATA_READ_PIO           0x20     /* read sector(s) with retries */
103: #define ATA_READ_MULTIPLE_PIO 0xC4     /* read multiple sectors */
104: #define ATA_WRITE_PIO          0x30     /* write sector(s) with retries */
105: #define ATA_WRITE_MULTIPLE_PIO 0xC5     /* write multiple sectors */
106: #define ATA_SET_MULTIPLE_MODE  0xC6     /* set multiple mode */
107: #define ATA_PACKET             0xA0     /* packet mode */
108: #define ATA_IDENTIFY_PACKET    0xA1     /* identify ATAPI device */
109: #define ATA_IDENTIFY           0xEC     /* identify ATA device */
110:
111: /* ATAPI commands */
112: #define ATAPI_TEST_UNIT        0x00     /* test unit ready */
113: #define ATAPI_REQUEST_SENSE    0x03     /* request sense */
114: #define ATAPI_START_STOP       0x1B     /* start/stop */
115: #define ATAPI_MEDIUM_REMOVAL   0x1E     /* medium removal */
116: #define ATAPI_READ10           0x28     /* read 10 */
117:
118: #define CD_UNLOCK_MEDIUM        0x00     /* allow medium removal */
119: #define CD_LOCK_MEDIUM         0x01     /* prevent medium removal */
120: #define CD_EJECT                0x02     /* eject the CD if possible */
121: #define CD_LOAD                  0x03     /* load the CD (closes tray) */
122:
123: /* ATAPI CD additional sense code */
124: #define ASC_NOT_READY           0x04     /* not ready */
125: #define ASC_NO_MEDIUM          0x3A     /* no medium */
126:
127: /* capabilities */
128: #define IDE_SUPPORTS_CFA        0x848A   /* CFA support */
129: #define IDE_HAS_DMA             0x100    /* DMA support */
130: #define IDE_HAS_LBA             0x200    /* LBA support */
131: #define IDE_MIN_LBA             16514064 /* sectors limit for using CHS */
132:
133: /* general configuration bits */
134: #define IDE_HAS_UDMA            0x04     /* device supports UDMA */

```


include/fiwix/ide.h

Page 3/5

```

135: #define IDE_REMOVABLE          0x80    /* removable media device */
136:
137: /* ATAPI types */
138: #define ATAPI_IS_SEQ_ACCESS    0x01    /* sequential-access device */
139: #define ATAPI_IS_PRINTER      0x02
140: #define ATAPI_IS_PROCESSOR    0x03
141: #define ATAPI_IS_WRITE_ONCE   0x04
142: #define ATAPI_IS_CDROM        0x05
143: #define ATAPI_IS_SCANNER      0x06
144:
145: /* IDE drive flags */
146: #define DEVICE_IS_ATAPI        0x01
147: #define DEVICE_IS_CFA          0x02
148: #define DEVICE_IS_DISK        0x04
149: #define DEVICE_IS_CDROM        0x08
150: #define DEVICE_REQUIRES_LBA    0x10
151: #define DEVICE_HAS_RW_MULTIPLE 0x20
152:
153: /* ATA/ATAPI-4 based */
154: struct ide_drv_ident {
155:     unsigned short int gen_config; /* general configuration bits */
156:     unsigned short int logic_cyls; /* logical cylinders */
157:     unsigned short int reserved2;
158:     unsigned short int logic_heads; /* logical heads */
159:     unsigned short int retired4;
160:     unsigned short int retired5;
161:     unsigned short int logic_spt; /* logical sectors/track */
162:     unsigned short int retired7;
163:     unsigned short int retired8;
164:     unsigned short int retired9;
165:     char serial_number[20]; /* serial number */
166:     unsigned short int vendor_spec20;
167:     unsigned short int buffer_cache;
168:     unsigned short int vendor_spec22; /* reserved */
169:     char firmware_rev[8]; /* firmware version */
170:     char model_number[40]; /* model number */
171:     unsigned short int rw_multiple;
172:     unsigned short int reserved48;
173:     unsigned short int capabilities; /* capabilities */
174:     unsigned short int reserved50;
175:     unsigned short int pio_mode; /* PIO data transfer mode */
176:     unsigned short int dma_mode;
177:     unsigned short int fields_validity; /* fields validity */
178:     unsigned short int cur_log_cyls; /* current logical cylinders */
179:     unsigned short int cur_log_heads; /* current logical heads */
180:     unsigned short int cur_log_spt; /* current logical sectors/track */
*/
181:     unsigned short int cur_capacity; /* current capacity in sectors */
/
182:     unsigned short int cur_capacity2; /* 32bit number */
183:     unsigned short int mult_sect_set; /* multiple sector settings */
184:     unsigned short int tot_sectors; /* sectors (LBA only) */
185:     unsigned short int tot_sectors2; /* 32bit number */
186:     unsigned short int singleword_dma;
187:     unsigned short int multiword_dma; /* multiword DMA settings */
188:     unsigned short int adv_pio_modes; /* advanced PIO modes */
189:     unsigned short int min_multiword; /* min. Multiword DMA transfer */
/
190:     unsigned short int rec_multiword; /* recommended Multiword DMS tra
nsfer */
191:     unsigned short int min_pio_wo_fc; /* min. PIO w/o flow control */
192:     unsigned short int min_pio_w_fc; /* min. PIO with flow control */
193:     unsigned short int reserved69;
194:     unsigned short int reserved70;
195:     unsigned short int reserved71;
196:     unsigned short int reserved72;
197:     unsigned short int reserved73;

```

include/fiwix/ide.h

Page 4/5

```

198:     unsigned short int reserved74;
199:     unsigned short int queue_depth;           /* queue depth */
200:     unsigned short int reserved76;
201:     unsigned short int reserved77;
202:     unsigned short int reserved78;
203:     unsigned short int reserved79;
204:     unsigned short int majorver;             /* major version number */
205:     unsigned short int minorver;           /* minor version number */
206:     unsigned short int cmdset1;            /* command set supported */
207:     unsigned short int cmdset2;            /* command set supported */
208:     unsigned short int cmdsf_ext;          /* command set/feature sup.ext.
*/
209:     unsigned short int cmdsf_enable1;      /* command s/f enabled */
210:     unsigned short int cmdsf_enable2;      /* command s/f enabled */
211:     unsigned short int cmdsf_default;      /* command s/f default */
212:     unsigned short int ultradma;          /* ultra DMA mode */
213:     unsigned short int reserved89;
214:     unsigned short int reserved90;
215:     unsigned short int curapm;            /* current APM values */
216:     unsigned short int reserved92_126[35];
217:     unsigned short int r_status_notif;     /* removable media status notif.
*/
218:     unsigned short int security_status;    /* security status */
219:     unsigned short int vendor_spec129_159[31];
220:     unsigned short int reserved160_255[96];
221: };
222:
223: struct ide_drv {
224:     int drive;                               /* master or slave */
225:     char *dev_name;
226:     unsigned char major;                     /* major number */
227:     unsigned int flags;
228:     int minor_shift;                         /* shift factor to get the real minor */
229:     int lba_cyls;
230:     int lba_heads;
231:     short int lba_factor;
232:     unsigned int nr_sects;                   /* total sectors (LBA) */
233:     struct fs_operations *fsop;
234:     struct ide_drv_ident ident;
235:     struct partition part_table[NR_PARTITIONS];
236: };
237:
238: struct ide {
239:     int channel;                             /* primary or secondary */
240:     int base;                                /* base address */
241:     int ctrl;                                /* control port address */
242:     short int irq;
243:     int current;                             /* avoids re-select the drive */
244:     struct resource resource;
245:     struct ide_drv drive[NR_IDE_DRVS];
246: };
247:
248: extern struct ide ide_table[NR_IDE_CTRLs];
249:
250: extern int ide0_need_reset;
251: extern int ide0_wait_interrupt;
252: extern int ide0_timeout;
253: extern int idel_need_reset;
254: extern int idel_wait_interrupt;
255: extern int idel_timeout;
256:
257: void irq_ide0(int, struct sigcontext *);
258: void ide0_timer(unsigned int);
259: void irq_idel(int, struct sigcontext *);
260: void idel_timer(unsigned int);
261:
262: void ide_error(struct ide *, int);

```

include/fiwix/ide.h

Page 5/5

```
263: void ide_delay(void);
264: void ide_wait400ns(struct ide *);
265: int ide_drvsel(struct ide *, int, int, unsigned char);
266: int ide_softreset(struct ide *);
267:
268: struct ide * get_ide_controller(__dev_t);
269: int get_ide_drive(__dev_t);
270:
271: int ide_open(struct inode *, struct fd *);
272: int ide_close(struct inode *, struct fd *);
273: int ide_read(__dev_t, __blk_t, char *, int);
274: int ide_write(__dev_t, __blk_t, char *, int);
275: int ide_ioctl(struct inode *, int, unsigned long int);
276:
277: void ide_init(void);
278:
279: #endif /* _FIWIX_IDE_H */
```

include/fiwix/ide_hd.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/ide_hd.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_IDE_HD_H
9: #define _FIWIX_IDE_HD_H
10:
11: #include <fiwix/types.h>
12:
13: #define IDE_HD_SECTSIZE          512      /* sector size (in bytes) */
14:
15: int ide_hd_open(struct inode *, struct fd *);
16: int ide_hd_close(struct inode *, struct fd *);
17: int ide_hd_read(__dev_t, __blk_t, char *, int);
18: int ide_hd_write(__dev_t, __blk_t, char *, int);
19: int ide_hd_ioctl(struct inode *, int, unsigned long int);
20:
21: int ide_hd_init(struct ide *, int);
22:
23: #endif /* _FIWIX_IDE_HD_H */
```

include/fiwix/ioctl.h

Page 1/2

```

1: /*
2:  * fiwix/include/fiwix/ioctl.h
3:  */
4:
5: #ifndef _FIWIX_IOCTL_H
6: #define _FIWIX_IOCTL_H
7:
8: #define HDIO_GETGEO      0x0301      /* get device geometry */
9:
10: #define BLKRRPART       0x125F       /* re-read partition table */
11: #define BLKGETSIZE      0x1260       /* return device size */
12: #define BLKFLSBUF       0x1261       /* flush buffer cache */
13:
14: /* 0x54 is just a magic number to make these relatively unique ('T') */
15: #define TCGETS          0x5401
16: #define TCSETS          0x5402
17: #define TCSETSW         0x5403
18: #define TCSETSF         0x5404
19: #define TCGETA          0x5405
20: #define TCSETA          0x5406
21: #define TCSETAW         0x5407
22: #define TCSETAF         0x5408
23: #define TCSBRK          0x5409
24: #define TCXONC          0x540A
25: #define TCFLSH          0x540B
26: #define TIOCEXCL        0x540C
27: #define TIOCNXCL        0x540D
28: #define TIOCSCTTY       0x540E
29: #define TIOCGPGRP       0x540F
30: #define TIOCSPPGRP      0x5410
31: #define TIOCOUTQ        0x5411
32: #define TIOCSTI         0x5412
33: #define TIOCGWINSZ      0x5413
34: #define TIOCSWINSZ      0x5414
35: #define TIOCMGET        0x5415
36: #define TIOCMBIS        0x5416
37: #define TIOCMBIC        0x5417
38: #define TIOCMSET        0x5418
39: #define TIOCGSOFTCAR    0x5419
40: #define TIOCSSOFTCAR    0x541A
41: #define FIONREAD        0x541B
42: #define TIOCINQ          FIONREAD
43: #define TIOCLINUX       0x541C
44: #define TIOCCONS        0x541D
45: #define TIOCGSERIAL     0x541E
46: #define TIOCSSERIAL     0x541F
47: #define TIOCPKT         0x5420
48: #define FIONBIO         0x5421
49: #define TIOCNOTTY       0x5422
50: #define TIOCSETD        0x5423
51: #define TIOCGETD        0x5424
52: #define TCSBRKP         0x5425      /* Needed for POSIX tcsendbreak() */
53: #define TIOCTTYGSTRUCT  0x5426      /* For debugging only */
54: #define TIOCSBRK        0x5427      /* BSD compatibility */
55: #define TIOCCBRK        0x5428      /* BSD compatibility */
56: #define TIOCGSID        0x5429      /* Return the session ID of FD */
57: #define TIOCGPTN        _IOR('T',0x30, unsigned int) /* Get Pty Number (of pty-m
ux device) */
58: #define TIOCSPTLCK      _IOW('T',0x31, int) /* Lock/unlock Pty */
59:
60: #define FIONCLEX        0x5450      /* these numbers need to be adjusted. */
61: #define FIOCLEX         0x5451
62: #define FIOASYNC        0x5452
63: #define TIOCSERCONFIG   0x5453
64: #define TIOCSERGWILD    0x5454
65: #define TIOCSERSWILD    0x5455
66: #define TIOCGCLKTRMIOS  0x5456

```

include/fiwix/ioctl.h

Page 2/2

```
67: #define TIOCSLCKTRMIOS 0x5457
68: #define TIOCSERGSTRUCT 0x5458 /* For debugging only */
69: #define TIOCSERGETLSR 0x5459 /* Get line status register */
70: #define TIOCSERGETMULTI 0x545A /* Get multiport config */
71: #define TIOCSERSETMULTI 0x545B /* Set multiport config */
72:
73: #define TIOCMIWAIT 0x545C /* wait for a change on serial input line(s) */
74: #define TIOCGICOUNT 0x545D /* read serial port inline interrupt counts */
75: #define TIOCGHAYESESP 0x545E /* Get Hayes ESP configuration */
76: #define TIOCSHAYESESP 0x545F /* Set Hayes ESP configuration */
77:
78: /* Used for packet mode */
79: #define TIOCPKT_DATA 0
80: #define TIOCPKT_FLUSHREAD 1
81: #define TIOCPKT_FLUSHWRITE 2
82: #define TIOCPKT_STOP 4
83: #define TIOCPKT_START 8
84: #define TIOCPKT_NOSTOP 16
85: #define TIOCPKT_DOSTOP 32
86:
87: #define TIOCSER_TEMT 0x01 /* Transmitter physically empty */
88:
89: #endif /* _FIWIX_IOCTL_H */
```

include/fiwix/kd.h

Page 1/3

```

1: /*
2:  * fiwix/include/fiwix/kd.h
3:  */
4:
5: #ifndef _LINUX_KD_H
6: #define _LINUX_KD_H
7:
8: /* Prefix 0x4B is 'K', to avoid collision with termios and vt */
9:
10: #define GIO_FONT      0x4B60 /* gets font in expanded form */
11: #define PIO_FONT      0x4B61 /* use font in expanded form */
12:
13: #define GIO_FONTX     0x4B6B /* get font using struct consolefontdesc */
14: #define PIO_FONTX     0x4B6C /* set font using struct consolefontdesc */
15: struct consolefontdesc {
16:     unsigned short int charcount; /* characters in font (256 or 512) */
17:     unsigned short int charheight; /* scan lines per character (1-32) */
18:     char *chardata; /* font data in expanded form */
19: };
20:
21: #define PIO_FONTRESET 0x4B6D /* reset to default font */
22:
23: #define GIO_CMAP      0x4B70 /* gets colour palette on VGA+ */
24: #define PIO_CMAP      0x4B71 /* sets colour palette on VGA+ */
25:
26: #define KIOCSOUND     0x4B2F /* start sound generation (0 for off) */
27: #define KDMKTONE      0x4B30 /* generate tone */
28:
29: #define KDGETLED      0x4B31 /* return current led state */
30: #define KDSETLED      0x4B32 /* set led state [lights, not flags] */
31: #define LED_SCR       0x01 /* scroll lock led */
32: #define LED_NUM       0x02 /* num lock led */
33: #define LED_CAP       0x04 /* caps lock led */
34:
35: #define KDGBKTYPE     0x4B33 /* get keyboard type */
36: #define KB_84         0x01
37: #define KB_101        0x02 /* this is what we always answer */
38: #define KB_OTHER      0x03
39:
40: #define KDADDIO       0x4B34 /* add i/o port as valid */
41: #define KDDELIO       0x4B35 /* del i/o port as valid */
42: #define KDENABIO      0x4B36 /* enable i/o to video board */
43: #define KDDISABIO     0x4B37 /* disable i/o to video board */
44:
45: #define KDSETMODE     0x4B3A /* set text/graphics mode */
46: #define KD_TEXT       0x00
47: #define KD_GRAPHICS   0x01
48: #define KD_TEXT0      0x02 /* obsolete */
49: #define KD_TEXT1      0x03 /* obsolete */
50: #define KDGETMODE     0x4B3B /* get current mode */
51:
52: #define KDMAPDISP     0x4B3C /* map display into address space */
53: #define KDUNMAPDISP   0x4B3D /* unmap display from address space */
54:
55: typedef char scrnmap_t;
56: #define E_TABSZ        256
57: #define GIO_SCRNMAP    0x4B40 /* get screen mapping from kernel */
58: #define PIO_SCRNMAP    0x4B41 /* put screen mapping table in kernel */
59: #define GIO_UNISCRNMAP 0x4B69 /* get full Unicode screen mapping */
60: #define PIO_UNISCRNMAP 0x4B6A /* set full Unicode screen mapping */
61:
62: #define GIO_UNIMAP     0x4B66 /* get unicode-to-font mapping from kernel */
63: struct unipair {
64:     unsigned short int unicode;
65:     unsigned short int fontpos;
66: };
67: struct unimapdesc {

```

include/fiwix/kd.h

Page 2/3

```

68:         unsigned short int entry_ct;
69:         struct unipair *entries;
70: };
71: #define PIO_UNIMAP      0x4B67 /* put unicode-to-font mapping in kernel */
72: #define PIO_UNIMAPCLR   0x4B68 /* clear table, possibly advise hash algorithm */
/
73: struct unimapinit {
74:     unsigned short int advised_hashsize; /* 0 if no opinion */
75:     unsigned short int advised_hashstep; /* 0 if no opinion */
76:     unsigned short int advised_hashlevel; /* 0 if no opinion */
77: };
78:
79: #define UNI_DIRECT_BASE 0xF000 /* start of Direct Font Region */
80: #define UNI_DIRECT_MASK 0x01FF /* Direct Font Region bitmask */
81:
82: #define K_RAW           0x00
83: #define K_XLATE        0x01
84: #define K_MEDIUMRAW    0x02
85: #define K_UNICODE      0x03
86: #define KDGBKMODE     0x4B44 /* gets current keyboard mode */
87: #define KDSKMODE      0x4B45 /* sets current keyboard mode */
88:
89: #define K_METABIT      0x03
90: #define K_ESCPREFIX    0x04
91: #define KDGBKMETA     0x4B62 /* gets meta key handling mode */
92: #define KDSKBMETA     0x4B63 /* sets meta key handling mode */
93:
94: #define K_SCROLLLOCK   0x01
95: #define K_NUMLOCK      0x02
96: #define K_CAPSLOCK     0x04
97: #define KDGBKLED      0x4B64 /* get led flags (not lights) */
98: #define KDSKLED       0x4B65 /* set led flags (not lights) */
99:
100: struct kbentry {
101:     unsigned char kb_table;
102:     unsigned char kb_index;
103:     unsigned short int kb_value;
104: };
105: #define K_NORMTAB      0x00
106: #define K_SHIFTTAB     0x01
107: #define K_ALTTAB      0x02
108: #define K_ALTSHIFTTAB 0x03
109:
110: #define KDGBKENT      0x4B46 /* gets one entry in translation table */
111: #define KDSKENT      0x4B47 /* sets one entry in translation table */
112:
113: struct kbsentry {
114:     unsigned char kb_func;
115:     unsigned char kb_string[512];
116: };
117: #define KDGBKSENT     0x4B48 /* gets one function key string entry */
118: #define KDSKSENT     0x4B49 /* sets one function key string entry */
119:
120: struct kbdiacr {
121:     unsigned char diacr, base, result;
122: };
123: struct kbdiacrs {
124:     unsigned int kb_cnt; /* number of entries in following array */
125:     struct kbdiacr kbdiacr[256]; /* MAX_DIACR from keyboard.h */
126: };
127: #define KDGBKDIACR    0x4B4A /* read kernel accent table */
128: #define KDSKDIACR    0x4B4B /* write kernel accent table */
129:
130: struct kbkeycode {
131:     unsigned int scancode, keycode;
132: };
133: #define KDGETKEYCODE  0x4B4C /* read kernel keycode table entry */

```


include/fiwix/kd.h

Page 3/3

```

134: #define KDSETKEYCODE    0x4B4D  /* write kernel keycode table entry */
135:
136: #define KDSIGACCEPT     0x4B4E  /* accept kbd generated signals */
137:
138: struct kbd_repeat {
139:     int delay;           /* in msec; <= 0: don't change */
140:     int rate;           /* in msec; <= 0: don't change */
141: };
142:
143: #define KDKBDREP        0x4B52  /* set keyboard delay/repeat rate;
144:                                * actually used values are returned */
145:
146: #define KDFONTOP        0x4B72  /* font operations */
147:
148: struct console_font_op {
149:     unsigned int op;     /* operation code KD_FONT_OP_* */
150:     unsigned int flags;  /* KD_FONT_FLAG_* */
151:     unsigned int width, height; /* font size */
152:     unsigned int charcount;
153:     unsigned char *data; /* font data with height fixed to 32 */
154: };
155:
156: #define KD_FONT_OP_SET      0      /* Set font */
157: #define KD_FONT_OP_GET      1      /* Get font */
158: #define KD_FONT_OP_SET_DEFAULT 2    /* Set font to default, data points to n
ame / NULL */
159: #define KD_FONT_OP_COPY      3      /* Copy from another console */
160:
161: #define KD_FONT_FLAG_DONT_RECALC 1    /* Don't recalculate hw charcell
size [compat] */
162:
163: /* note: 0x4B00-0x4B4E all have had a value at some time;
164:    don't reuse for the time being */
165: /* note: 0x4B60-0x4B6D, 0x4B70-0x4B72 used above */
166:
167: #endif /* _LINUX_KD_H */

```

include/fiwix/kernel.h

Page 1/2

```

1: /*
2:  * fiwix/include/fiwix/kernel.h
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_KERNEL_H
9: #define _FIWIX_KERNEL_H
10:
11: #include <fiwix/limits.h>
12: #include <fiwix/i386elf.h>
13:
14: #define PANIC(format, args...)                                \
15: {                                                            \
16:     printk("\nPANIC: in %s()", __FUNCTION__);                \
17:     printk("\n");                                           \
18:     printk(format, ## args);                                \
19:     stop_kernel();                                         \
20: }
21:
22: #define CURRENT_TIME      (kstat.system_time)
23: #define CURRENT_TICKS    (kstat.ticks)
24: #define INIT_PROGRAM      "/sbin/init"
25:
26: extern char *init_argv[];
27: extern char *init_envp[];
28: extern char *init_args;
29:
30: extern Elf32_Shdr *symtab, *strtab;
31: extern unsigned int _last_data_addr;
32:
33: extern int _memsize;
34: extern int _extmemsize;
35: extern int _rootdev;
36: extern int _noramdisk;
37: extern int _ramdisksize;
38: extern char _rootfstype[10];
39: extern char _rootdevname[DEVNAME_MAX + 1];
40: extern char _initrd[DEVNAME_MAX + 1];
41: extern int _syscondev;
42:
43: extern int _cputype;
44: extern int _cpusignature;
45: extern int _cpuflags;
46: extern int _brandid;
47: extern char _vendorid[12];
48: extern char _brandstr[48];
49: extern unsigned int _tlbinfo_eax;
50: extern unsigned int _tlbinfo_ebx;
51: extern unsigned int _tlbinfo_ecx;
52: extern unsigned int _tlbinfo_edx;
53: extern char _etext[], _edata[], _end[];
54:
55: extern char cmdline[NAME_MAX + 1];
56:
57: struct kernel_stat {
58:     unsigned int cpu_user;           /* ticks in user-mode */
59:     unsigned int cpu_nice;          /* ticks in user-mode (with priority) */
60:     unsigned int cpu_system;        /* ticks in kernel-mode */
61:     unsigned int irq;               /* irq counter */
62:     unsigned int sirqs;              /* spurious irq counter */
63:     unsigned int ctxt;              /* context switches */
64:     unsigned int ticks;              /* ticks (1/HZths of sec) since boot */
65:     unsigned int system_time;       /* current system time (since the Epoch) */
66:     unsigned int boot_time;         /* boot time (since the Epoch) */

```

include/fiwix/kernel.h

Page 2/2

```
67:      int tz_minuteswest;          /* minutes west of GMT */
68:      int tz_dsttime;             /* type of DST correction */
69:      unsigned int uptime;         /* seconds since boot */
70:      unsigned int processes;     /* number of forks since boot */
71:      int physical_pages;         /* physical memory (in pages) */
72:      int kernel_reserved;        /* kernel memory reserved (in KB) */
73:      int physical_reserved;      /* physical memory reserved (in KB) */
74:      int total_mem_pages;        /* total memory (in pages) */
75:      int free_pages;             /* pages on free list */
76:      int free_inodes;           /* inodes on free list */
77:      int buffers;               /* memory used by buffers (in KB) */
78:      int cached;                /* memory used to cache file pages */
79:      int shared;                /* pages with count > 1 */
80:      int dirty;                 /* dirty buffers (in KB) */
81:      unsigned long int random_seed; /* next random seed */
82: };
83: extern struct kernel_stat kstat;
84:
85: unsigned int get_last_boot_addr(unsigned int);
86: void multiboot(unsigned long, unsigned long);
87: void start_kernel(unsigned long, unsigned long, unsigned int);
88: void stop_kernel(void);
89: void init_init(void);
90: void cpu_idle(void);
91:
92: #endif /* _FIWIX_KERNEL_H */
```

include/fiwix/keyboard.h

Page 1/3

```

1: /*
2:  * fiwix/include/fiwix/keyboard.h
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_KEYBOARD_H
9: #define _FIWIX_KEYBOARD_H
10:
11: #define KEYBOARD_IRQ    1
12:
13: #define NR_MODIFIERS    16      /* max. number of modifiers per keymap */
14: #define NR_SCODES      128     /* max. number of scancodes */
15: #define NR_DIACR       10
16:
17: #define SCRLBIT         0x01    /* scroll lock led */
18: #define NUMSBIT         0x02    /* num lock led */
19: #define CAPSBIT         0x04    /* caps lock led */
20:
21: #define C(ch)           ((ch) & 0x1F)
22: #define A(ch)           ((ch) | META_KEYS)
23: #define L(ch)           ((ch) | LETTER_KEYS)
24:
25: #define SLASH_NPAD     53
26:
27: #define EOENTER         96
28: #define RCTRL           97
29: #define EOSLASH         98
30: #define ALTGR           100
31: #define EOHOME          102
32: #define EOUP            103
33: #define EOPGUP          104
34: #define EOLEFT          105
35: #define EORIGHT         106
36: #define EOEND           107
37: #define EODOWN          108
38: #define EOPGDN          109
39: #define EOINS           110
40: #define EODEL           111
41:
42: #define MOD_BASE        0
43: #define MOD_SHIFT       1
44: #define MOD_ALTGR       2
45: #define MOD_CTRL        3
46: #define MOD_ALT         4
47: #define MOD_SHIFTL      5
48: #define MOD_SHIFTR      6
49: #define MOD_CTRLRL      7
50: #define MOD_CTRLRLR     8
51:
52: #define FN_KEYS         0x100
53: #define SPEC_KEYS       0x200
54: #define PAD_KEYS        0x300
55: #define DEAD_KEYS       0x400
56: #define CONS_KEYS       0x500
57: #define SHIFT_KEYS      0x700
58: #define META_KEYS       0x800
59: #define LETTER_KEYS     0xB00
60:
61: #define CR               (0x01 + SPEC_KEYS)
62: #define SCRL2            (0x02 + SPEC_KEYS)      /* SH_REGS (show registers) */
63: #define SCRL3            (0x03 + SPEC_KEYS)      /* SH_MEM (show memory) */
64: #define SCRL4            (0x04 + SPEC_KEYS)      /* SH_STAT (show status) */
65: #define SYSRQ           (0x06 + SPEC_KEYS)
66: #define CAPS             (0x07 + SPEC_KEYS)
67: #define NUMS             (0x08 + SPEC_KEYS)

```

include/fiwix/keyboard.h

Page 2/3

```

68: #define SCRL          (0x09 + SPEC_KEYS)
69:
70: #define INS            (0x00 + PAD_KEYS)
71: #define END            (0x01 + PAD_KEYS)
72: #define DOWN          (0x02 + PAD_KEYS)
73: #define PGDN          (0x03 + PAD_KEYS)
74: #define LEFT          (0x04 + PAD_KEYS)
75: #define MID           (0x05 + PAD_KEYS)
76: #define RIGHT         (0x06 + PAD_KEYS)
77: #define HOME          (0x07 + PAD_KEYS)
78: #define UP            (0x08 + PAD_KEYS)
79: #define PGUP          (0x09 + PAD_KEYS)
80: #define PLUS          (0x0A + PAD_KEYS)
81: #define MINUS         (0x0B + PAD_KEYS)
82: #define ASTSK         (0x0C + PAD_KEYS)
83: #define SLASH         (0x0D + PAD_KEYS)
84: #define ENTER         (0x0E + PAD_KEYS)
85: #define DEL           (0x10 + PAD_KEYS)
86:
87: #define GRAVE         (0x00 + DEAD_KEYS)
88: #define ACUTE         (0x01 + DEAD_KEYS)
89: #define CIRCUM        (0x02 + DEAD_KEYS)
90: #define DIERE         (0x04 + DEAD_KEYS)
91:
92: #define SHIFT         (0x00 + SHIFT_KEYS)
93: #define CTRL          (0x02 + SHIFT_KEYS)
94: #define ALT           (0x03 + SHIFT_KEYS)
95: #define LSHIFT        (0x04 + SHIFT_KEYS)
96: #define RSHIFT        (0x05 + SHIFT_KEYS)
97: #define LCTRL         (0x06 + SHIFT_KEYS)
98:
99: #define F1            (0x00 + FN_KEYS)
100: #define F2            (0x01 + FN_KEYS)
101: #define F3            (0x02 + FN_KEYS)
102: #define F4            (0x03 + FN_KEYS)
103: #define F5            (0x04 + FN_KEYS)
104: #define F6            (0x05 + FN_KEYS)
105: #define F7            (0x06 + FN_KEYS)
106: #define F8            (0x07 + FN_KEYS)
107: #define F9            (0x08 + FN_KEYS)
108: #define F10           (0x09 + FN_KEYS)
109: #define F11           (0x0A + FN_KEYS)
110: #define F12           (0x0B + FN_KEYS)
111:
112: #define SF1           (0x0A + FN_KEYS)
113: #define SF2           (0x0B + FN_KEYS)
114: #define SF3           (0x0C + FN_KEYS)
115: #define SF4           (0x0D + FN_KEYS)
116: #define SF5           (0x0E + FN_KEYS)
117: #define SF6           (0x0F + FN_KEYS)
118: #define SF7           (0x10 + FN_KEYS)
119: #define SF8           (0x11 + FN_KEYS)
120: #define SF9           (0x12 + FN_KEYS)
121: #define SF10          (0x13 + FN_KEYS)
122: #define SF11          (0x0A + SHIFT)
123: #define SF12          (0x0B + SHIFT)
124:
125: #define AF1           (0x00 + CONS_KEYS)
126: #define AF2           (0x01 + CONS_KEYS)
127: #define AF3           (0x02 + CONS_KEYS)
128: #define AF4           (0x03 + CONS_KEYS)
129: #define AF5           (0x04 + CONS_KEYS)
130: #define AF6           (0x05 + CONS_KEYS)
131: #define AF7           (0x06 + CONS_KEYS)
132: #define AF8           (0x07 + CONS_KEYS)
133: #define AF9           (0x08 + CONS_KEYS)
134: #define AF10          (0x09 + CONS_KEYS)

```

include/fiwix/keyboard.h

Page 3/3

```
135: #define AF11          (0x0A + CONS_KEYS)
136: #define AF12          (0x0B + CONS_KEYS)
137:
138: #ifdef __KERNEL__
139:
140: #include <fiwix/types.h>
141: #include <fiwix/sigcontext.h>
142:
143: struct diacritic {
144:     unsigned char letter;
145:     unsigned char code;
146: };
147:
148: extern __key_t keymap[NR_MODIFIERS * NR_SCODES];
149:
150: void set_leds(unsigned char);
151: void irq_keyboard(int num, struct sigcontext *);
152: void irq_keyboard_bh(void);
153: void keyboard_init(void);
154:
155: #endif /* __KERNEL__ */
156:
157: #endif /* _FIWIX_KEYBOARD_H */
```

include/fiwix/kparms.h

Page 1/1

```

1: /*
2:  * fiwix/include/fiwix/kparms.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_KPARMS_H
9: #define _FIWIX_KPARMS_H
10:
11: #define CMDL_ARG_LEN    100    /* max length of cmdline argument */
12: #define CMDL_NUM_VALUES 30    /* max values of cmdline parameter */
13:
14: struct kparms {
15:     char *name;
16:     char *value[CMDL_NUM_VALUES];
17:     unsigned int sysval[CMDL_NUM_VALUES];
18: };
19:
20: static struct kparms parm_table[] = {
21:     { "root=",
22:       { "/dev/ram0", "/dev/fd0", "/dev/fd1",
23:         "/dev/hda", "/dev/hda1", "/dev/hda2", "/dev/hda3", "/dev/hda4",
24:         "/dev/hdb", "/dev/hdb1", "/dev/hdb2", "/dev/hdb3", "/dev/hdb4",
25:         "/dev/hdc", "/dev/hdc1", "/dev/hdc2", "/dev/hdc3", "/dev/hdc4",
26:         "/dev/hdd", "/dev/hdd1", "/dev/hdd2", "/dev/hdd3", "/dev/hdd4",
27:       },
28:       { 0x100, 0x200, 0x201,
29:         0x300, 0x301, 0x302, 0x303, 0x304,
30:         0x340, 0x341, 0x342, 0x343, 0x344,
31:         0x1600, 0x1601, 0x1602, 0x1603, 0x1604,
32:         0x1640, 0x1641, 0x1642, 0x1643, 0x1644,
33:       }
34:     },
35:     { "noramdisk",
36:       { NULL },
37:       { NULL },
38:     },
39:     { "ramdisksize=",
40:       { NULL },
41:       { NULL },
42:     },
43:     { "initrd=",
44:       { NULL },
45:       { NULL },
46:     },
47:     { "rootfstype=",
48:       { "minix", "ext2", "iso9660" },
49:       { 0, 0 }
50:     },
51:     { "console=",
52:       { "/dev/tty1", "/dev/tty2", "/dev/tty3", "/dev/tty4", "/dev/tty5",
53:         "/dev/tty6", "/dev/tty7", "/dev/tty8", "/dev/tty9", "/dev/tty10",
54:         "/dev/tty11", "/dev/tty12", "/dev/ttyS0", "/dev/ttyS1",
55:         "/dev/ttyS2", "/dev/ttyS3"
56:       },
57:       { 0x401, 0x402, 0x403, 0x404, 0x405,
58:         0x406, 0x407, 0x408, 0x409, 0x410,
59:         0x411, 0x412, 0x440, 0x441, 0x442,
60:         0x443
61:       }
62:     },
63:     { NULL }
64: };
65: };
66:
67: #endif /* _FIWIX_KPARMS_H */

```

include/fiwix/limits.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/limits.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_LIMITS_H
9: #define _FIWIX_LIMITS_H
10:
11: #define DEVNAME_MAX      50      /* device name length in mount table */
12: #define ARG_MAX          32      /* length (in pages) of argv+env in 'execve' */
13: #define CHILD_MAX        64      /* simultaneous processes per real user ID */
14: #define LINK_MAX         255     /* maximum number of links to a file */
15: #define MAX_CANON        255     /* bytes in a terminal canonical input queue */
16: #define MAX_INPUT        255     /* bytes for which space will be available in a
17:                                     terminal input queue */
18: #define NGROUPS_MAX      32      /* simultaneous supplementary group IDs */
19: #define OPEN_MAX         256     /* files one process can have opened at once */
20: #define FD_SETSIZE       OPEN_MAX /* descriptors that a process may examine with
21:                                     'pselect' or 'select' */
22: #define NAME_MAX         255     /* bytes in a filename */
23: #define PATH_MAX         1024    /* bytes in a pathname */
24: #define PIPE_BUF         4096    /* bytes than can be written atomically to a
25:                                     pipe */
26:
27: #endif /* _FIWIX_LIMITS_H */
```


include/fiwix/locks.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/locks.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_LOCKS_H
9: #define _FIWIX_LOCKS_H
10:
11: #include <fiwix/config.h>
12: #include <fiwix/fs.h>
13: #include <fiwix/fcntl.h>
14:
15: struct flock_file {
16:     struct inode *inode;    /* file */
17:     unsigned char type;    /* type of lock */
18:     struct proc *proc;     /* owner */
19: };
20:
21: struct flock_file flock_file_table[NR_FLOCKS];
22:
23: int posix_lock(int, int, struct flock *);
24:
25: void flock_release_inode(struct inode *);
26: int flock_inode(struct inode *, int);
27: void flock_init(void);
28:
29: #endif /* _FIWIX_LOCKS_H */
```

include/fiwix/lp.h

Page 1/1

```

1: /*
2:  * fiwix/include/fiwix/lp.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_LP_H
9: #define _FIWIX_LP_H
10:
11: #include <fiwix/fs.h>
12:
13: #define LP_MAJOR          6          /* major number for /dev/lp[n] */
14: #define LP_MINORS        1
15:
16: /*#define LP0_ADDR        0x3BC */
17: #define LP0_ADDR          0x378
18: /*#define LP2_ADDR        0x278 */
19:
20: #define LP_STAT_ERR      0x08      /* printer error */
21: #define LP_STAT_SEL      0x10      /* select in */
22: #define LP_STAT_PE       0x20      /* paper empty or no paper */
23: #define LP_STAT_ACK      0x40      /* ack */
24: #define LP_STAT_BUS      0x80      /* printer busy */
25:
26: #define LP_CTRL_STR      0x01      /* strobe */
27: #define LP_CTRL_AUT      0x02      /* auto line feed */
28: #define LP_CTRL_INI      0x04      /* initialize printer (reset) */
29: #define LP_CTRL_SEL      0x08      /* select printer */
30: #define LP_CTRL_IRQ      0x10      /* enable IRQ */
31: #define LP_CTRL_BID      0x20      /* bidireccional (on PS/2 ports) */
32:
33: #define LP_RDY_RETR      100       /* retries before timeout */
34:
35: struct lp {
36:     int data;             /* data port address */
37:     int stat;            /* status port address */
38:     int ctrl;           /* control port address */
39:     char flags;         /* flags */
40: };
41:
42: int lp_open(struct inode *, struct fd *);
43: int lp_close(struct inode *, struct fd *);
44: int lp_write(struct inode *, struct fd *, const char *, __size_t);
45:
46: void lp_init(void);
47:
48: #endif /* _FIWIX_LP_H */

```

include/fiwix/memdev.h

Page 1/1

```

1: /*
2:  * fiwix/include/fiwix/memdev.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_MEMDEV_H
9: #define _FIWIX_MEMDEV_H
10:
11: #include <fiwix/fs.h>
12:
13: #define MEMDEV_MAJOR 1      /* major number */
14: #define MEMDEV_MINORS 5    /* number of supported minors */
15:
16: #define MEMDEV_MEM 1       /* minor for /dev/mem */
17: #define MEMDEV_KMEM 2     /* minor for /dev/kmem */
18: #define MEMDEV_NULL 3     /* minor for /dev/null */
19: #define MEMDEV_ZERO 5     /* minor for /dev/zero */
20: #define MEMDEV_RANDOM 8   /* minor for /dev/random */
21: #define MEMDEV_URANDOM 9  /* minor for /dev/urandom */
22:
23: int mem_open(struct inode *, struct fd *);
24: int mem_close(struct inode *, struct fd *);
25: int mem_read(struct inode *, struct fd *, char *, __size_t);
26: int mem_write(struct inode *, struct fd *, const char *, __size_t);
27: int mem_lseek(struct inode *, __off_t);
28:
29: int kmem_open(struct inode *, struct fd *);
30: int kmem_close(struct inode *, struct fd *);
31: int kmem_read(struct inode *, struct fd *, char *, __size_t);
32: int kmem_write(struct inode *, struct fd *, const char *, __size_t);
33: int kmem_lseek(struct inode *, __off_t);
34:
35: int null_open(struct inode *, struct fd *);
36: int null_close(struct inode *, struct fd *);
37: int null_read(struct inode *, struct fd *, char *, __size_t);
38: int null_write(struct inode *, struct fd *, const char *, __size_t);
39: int null_lseek(struct inode *, __off_t);
40:
41: int zero_open(struct inode *, struct fd *);
42: int zero_close(struct inode *, struct fd *);
43: int zero_read(struct inode *, struct fd *, char *, __size_t);
44: int zero_write(struct inode *, struct fd *, const char *, __size_t);
45: int zero_lseek(struct inode *, __off_t);
46:
47: int urandom_open(struct inode *, struct fd *);
48: int urandom_close(struct inode *, struct fd *);
49: int urandom_read(struct inode *, struct fd *, char *, __size_t);
50: int urandom_write(struct inode *, struct fd *, const char *, __size_t);
51: int urandom_lseek(struct inode *, __off_t);
52:
53: int memdev_open(struct inode *, struct fd *);
54: int mem_mmap(struct inode *, struct vma *);
55: void memdev_init(void);
56:
57: #endif /* _FIWIX_MEMDEV_H */

```

include/fiwix/mman.h

Page 1/1

```

1: /*
2:  * fiwix/include/fiwix/mman.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_MMAN_H
9: #define _FIWIX_MMAN_H
10:
11: #include <fiwix/fs.h>
12:
13: #define PROT_READ          0x1          /* page can be read */
14: #define PROT_WRITE        0x2          /* page can be written */
15: #define PROT_EXEC         0x4          /* page can be executed */
16: #define PROT_NONE         0x0          /* page cannot be accessed */
17:
18: #define MAP_SHARED        0x01         /* share changes */
19: #define MAP_PRIVATE       0x02         /* changes are private */
20: #define MAP_TYPE          0x0f         /* mask for type of mapping */
21: #define MAP_FIXED         0x10         /* interpret address exactly */
22: #define MAP_ANONYMOUS    0x20         /* don't use the file descriptor */
23:
24: #define MAP_GROWSDOWN     0x0100       /* stack-like segment */
25: #define MAP_DENYWRITE     0x0800       /* -ETXTBSY */
26: #define MAP_EXECUTABLE    0x1000       /* mark it as a executable */
27: #define MAP_LOCKED        0x2000       /* pages are locked */
28:
29: #define ZERO_PAGE         0x80000000    /* this page must be zero-filled */
30:
31: #define MS_ASYNC          1            /* sync memory asynchronously */
32: #define MS_INVALIDATE    2            /* invalidate the caches */
33: #define MS_SYNC           4            /* synchronous memory sync */
34:
35: #define MCL_CURRENT       1            /* lock all current mappings */
36: #define MCL_FUTURE        2            /* lock all future mappings */
37:
38: /* compatibility flags */
39: #define MAP_ANON          MAP_ANONYMOUS
40: #define MAP_FILE          0
41:
42: struct mmap {
43:     unsigned int start;
44:     unsigned int length;
45:     unsigned int prot;
46:     unsigned int flags;
47:     int fd;
48:     unsigned int offset;
49: };
50:
51: void show_vma_regions(struct proc *);
52: void release_binary(void);
53: struct vma * find_vma_region(unsigned int);
54: int expand_heap(unsigned int);
55: int do_mmap(struct inode *, unsigned int, unsigned int, unsigned int, unsigned int,
nt, unsigned int, char, char);
56: int do_munmap(unsigned int, __size_t);
57: int do_mprotect(struct vma *, unsigned int, __size_t, int);
58:
59: #endif /* _FIWIX_MMAN_H */

```

include/fiwix/mm.h

Page 1/2

```

1: /*
2:  * fiwix/include/fiwix/mm.h
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_MEMORY_H
9: #define _FIWIX_MEMORY_H
10:
11: #include <fiwix/types.h>
12: #include <fiwix/segments.h>
13: #include <fiwix/process.h>
14:
15: /*
16:  * Convert only from physical to virtual the addresses below KERNEL_BASE_ADDR
17:  * (formerly 0xC0000000).
18:  */
19: #define P2V(addr)          (addr < KERNEL_BASE_ADDR ? addr + KERNEL_BASE_AD
DR : addr)
20:
21: #define V2P(addr)          (addr - KERNEL_BASE_ADDR)
22:
23: #define PAGE_SIZE          4096
24: #define PAGE_SHIFT         0x0C
25: #define PAGE_MASK          ~(PAGE_SIZE - 1)          /* 0xFFFFF000 */
26: #define PAGE_ALIGN(addr)   ((addr) + (PAGE_SIZE - 1)) & PAGE_MASK
27: #define PT_ENTRIES         (PAGE_SIZE / sizeof(unsigned int))
28: #define PD_ENTRIES         (PAGE_SIZE / sizeof(unsigned int))
29:
30: #define PAGE_PRESENT       0x001    /* Present */
31: #define PAGE_RW            0x002    /* Read/Write */
32: #define PAGE_USER          0x004    /* User */
33:
34: #define PAGE_LOCKED        0x001
35: #define PAGE_RESERVED      0x100    /* kernel, BIOS address, ... */
36: #define PAGE_COW           0x200    /* marked for Copy-On-Write */
37:
38: #define PFAULT_V           0x01     /* protection violation */
39: #define PFAULT_W           0x02     /* during write */
40: #define PFAULT_U           0x04     /* in user mode */
41:
42: #define GET_PGDIR(address) ((unsigned int)((address) >> 22) & 0x3FF)
43: #define GET_PGTBL(address) ((unsigned int)((address) >> 12) & 0x3FF)
44:
45: #define P_TEXT 1           /* text section */
46: #define P_DATA 2           /* data section */
47: #define P_BSS 3           /* BSS section */
48: #define P_HEAP 4          /* heap section (sys_brk()) */
49: #define P_STACK 5         /* stack section */
50: #define P_MMAP 6          /* mmap() section */
51:
52: struct page {
53:     int page;              /* page number */
54:     int count;             /* usage counter */
55:     int flags;
56:     __ino_t inode;         /* inode of the file */
57:     __off_t offset;        /* file offset */
58:     __dev_t dev;          /* device where file resides */
59:     char *data;           /* page contents */
60:     struct page *prev_hash;
61:     struct page *next_hash;
62:     struct page *prev_free;
63:     struct page *next_free;
64: };
65:
66: extern struct page *page_table;

```

include/fiwix/mm.h

Page 2/2

```
67: extern struct page **page_hash_table;
68:
69: /* values to be determined during system startup */
70: extern unsigned int page_table_size;          /* size in bytes */
71: extern unsigned int page_hash_table_size;    /* size in bytes */
72:
73: extern unsigned int *kpage_dir;
74: extern unsigned int *kpage_table;
75:
76: /* alloc.c */
77: unsigned int kmalloc(void);
78: void kfree(unsigned int);
79:
80: /* page.c */
81: void page_lock(struct page *);
82: void page_unlock(struct page *);
83: struct page * get_free_page(void);
84: struct page * search_page_hash(struct inode *, __off_t);
85: void release_page(int);
86: int is_valid_page(int);
87: void update_page_cache(struct inode *, __off_t, const char *, int);
88: int write_page(struct page *, struct inode *, __off_t, unsigned int);
89: int bread_page(struct page *, struct inode *, __off_t, char, char);
90: int file_read(struct inode *, struct fd *, char *, __size_t);
91: void page_init(int);
92:
93: /* memory.c */
94: void bss_init(void);
95: unsigned int setup_minmem(void);
96: unsigned int get_mapped_addr(struct proc *, unsigned int);
97: int clone_pages(struct proc *);
98: int free_page_tables(struct proc *);
99: unsigned int map_page(struct proc *, unsigned int, unsigned int, unsigned int);
100: int unmap_page(unsigned int);
101: void mem_init(void);
102: void mem_stats(void);
103:
104: /* swapper.c */
105: int kswapd(void);
106:
107: #endif /* _FIWIX_MEMORY_H */
```

include/fiwix/multiboot1.h

Page 1/6

```

1:  /* multiboot.h - Multiboot header file. */
2:  /* Copyright (C) 1999,2003,2007,2008,2009,2010  Free Software Foundation, Inc.
3:  *
4:  *  Permission is hereby granted, free of charge, to any person obtaining a copy
5:  *  of this software and associated documentation files (the "Software"), to
6:  *  deal in the Software without restriction, including without limitation the
7:  *  rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
8:  *  sell copies of the Software, and to permit persons to whom the Software is
9:  *  furnished to do so, subject to the following conditions:
10: *
11: *  The above copyright notice and this permission notice shall be included in
12: *  all copies or substantial portions of the Software.
13: *
14: *  THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
15: *  IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
16: *  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.  IN NO EVENT SHALL ANY
17: *  DEVELOPER OR DISTRIBUTOR BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY
18: *  WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR
19: *  IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE
20: */
21:
22: #ifndef MULTIBOOT_HEADER
23: #define MULTIBOOT_HEADER 1
24:
25: /* How many bytes from the start of the file we search for the header. */
26: #define MULTIBOOT_SEARCH      8192
27: #define MULTIBOOT_HEADER_ALIGN 4
28:
29: /* The magic field should contain this. */
30: #define MULTIBOOT_HEADER_MAGIC      0x1BADB002
31:
32: /* This should be in %eax. */
33: #define MULTIBOOT_BOOTLOADER_MAGIC  0x2BADB002
34:
35: /* Alignment of multiboot modules. */
36: #define MULTIBOOT_MOD_ALIGN        0x00001000
37:
38: /* Alignment of the multiboot info structure. */
39: #define MULTIBOOT_INFO_ALIGN       0x00000004
40:
41: /* Flags set in the â\200\231flagsâ\200\231 member of the multiboot header. */
42:
43: /* Align all boot modules on i386 page (4KB) boundaries. */
44: #define MULTIBOOT_PAGE_ALIGN      0x00000001
45:
46: /* Must pass memory information to OS. */
47: #define MULTIBOOT_MEMORY_INFO     0x00000002
48:
49: /* Must pass video information to OS. */
50: #define MULTIBOOT_VIDEO_MODE      0x00000004
51:
52: /* This flag indicates the use of the address fields in the header. */
53: #define MULTIBOOT_AOUT_KLUDGE     0x00010000
54:
55: /* Flags to be set in the â\200\231flagsâ\200\231 member of the multiboot info s
structure. */
56:
57: /* is there basic lower/upper memory information? */
58: #define MULTIBOOT_INFO_MEMORY     0x00000001
59: /* is there a boot device set? */
60: #define MULTIBOOT_INFO_BOOTDEV   0x00000002
61: /* is the command-line defined? */
62: #define MULTIBOOT_INFO_CMDLINE   0x00000004
63: /* are there modules to do something with? */
64: #define MULTIBOOT_INFO_MODS      0x00000008

```

include/fiwix/multiboot1.h

Page 2/6

```

65:
66: /* These next two are mutually exclusive */
67:
68: /* is there a symbol table loaded? */
69: #define MULTIBOOT_INFO_AOUT_SYMS                0x00000010
70: /* is there an ELF section header table? */
71: #define MULTIBOOT_INFO_ELF_SHDR                0X00000020
72:
73: /* is there a full memory map? */
74: #define MULTIBOOT_INFO_MEM_MAP                0x00000040
75:
76: /* Is there drive info? */
77: #define MULTIBOOT_INFO_DRIVE_INFO            0x00000080
78:
79: /* Is there a config table? */
80: #define MULTIBOOT_INFO_CONFIG_TABLE          0x00000100
81:
82: /* Is there a boot loader name? */
83: #define MULTIBOOT_INFO_BOOT_LOADER_NAME      0x00000200
84:
85: /* Is there a APM table? */
86: #define MULTIBOOT_INFO_APM_TABLE            0x00000400
87:
88: /* Is there video information? */
89: #define MULTIBOOT_INFO_VBE_INFO              0x00000800
90: #define MULTIBOOT_INFO_FRAMEBUFFER_INFO     0x00001000
91:
92: #ifndef ASM_FILE
93:
94: typedef unsigned char        multiboot_uint8_t;
95: typedef unsigned short      multiboot_uint16_t;
96: typedef unsigned int        multiboot_uint32_t;
97: typedef unsigned long long  multiboot_uint64_t;
98:
99: struct multiboot_header
100: {
101:     /* Must be MULTIBOOT_MAGIC - see above. */
102:     multiboot_uint32_t magic;
103:
104:     /* Feature flags. */
105:     multiboot_uint32_t flags;
106:
107:     /* The above fields plus this one must equal 0 mod 2^32. */
108:     multiboot_uint32_t checksum;
109:
110:     /* These are only valid if MULTIBOOT_AOUT_KLUDGE is set. */
111:     multiboot_uint32_t header_addr;
112:     multiboot_uint32_t load_addr;
113:     multiboot_uint32_t load_end_addr;
114:     multiboot_uint32_t bss_end_addr;
115:     multiboot_uint32_t entry_addr;
116:
117:     /* These are only valid if MULTIBOOT_VIDEO_MODE is set. */
118:     multiboot_uint32_t mode_type;
119:     multiboot_uint32_t width;
120:     multiboot_uint32_t height;
121:     multiboot_uint32_t depth;
122: };
123:
124: /* The symbol table for a.out. */
125: struct multiboot_aout_symbol_table
126: {
127:     multiboot_uint32_t tabsize;
128:     multiboot_uint32_t strsize;
129:     multiboot_uint32_t addr;
130:     multiboot_uint32_t reserved;
131: };

```


include/fiwix/multiboot1.h

Page 3/6

```

132: typedef struct multiboot_aout_symbol_table multiboot_aout_symbol_table_t;
133:
134: /* The section header table for ELF. */
135: struct multiboot_elf_section_header_table
136: {
137:     multiboot_uint32_t num;
138:     multiboot_uint32_t size;
139:     multiboot_uint32_t addr;
140:     multiboot_uint32_t shndx;
141: };
142: typedef struct multiboot_elf_section_header_table multiboot_elf_section_header_t
able_t;
143:
144: struct multiboot_info
145: {
146:     /* Multiboot info version number */
147:     multiboot_uint32_t flags;
148:
149:     /* Available memory from BIOS */
150:     multiboot_uint32_t mem_lower;
151:     multiboot_uint32_t mem_upper;
152:
153:     /* "root" partition */
154:     multiboot_uint32_t boot_device;
155:
156:     /* Kernel command line */
157:     multiboot_uint32_t cmdline;
158:
159:     /* Boot-Module list */
160:     multiboot_uint32_t mods_count;
161:     multiboot_uint32_t mods_addr;
162:
163:     union
164:     {
165:         multiboot_aout_symbol_table_t aout_sym;
166:         multiboot_elf_section_header_table_t elf_sec;
167:     } u;
168:
169:     /* Memory Mapping buffer */
170:     multiboot_uint32_t mmap_length;
171:     multiboot_uint32_t mmap_addr;
172:
173:     /* Drive Info buffer */
174:     multiboot_uint32_t drives_length;
175:     multiboot_uint32_t drives_addr;
176:
177:     /* ROM configuration table */
178:     multiboot_uint32_t config_table;
179:
180:     /* Boot Loader Name */
181:     multiboot_uint32_t boot_loader_name;
182:
183:     /* APM table */
184:     multiboot_uint32_t apm_table;
185:
186:     /* Video */
187:     multiboot_uint32_t vbe_control_info;
188:     multiboot_uint32_t vbe_mode_info;
189:     multiboot_uint16_t vbe_mode;
190:     multiboot_uint16_t vbe_interface_seg;
191:     multiboot_uint16_t vbe_interface_off;
192:     multiboot_uint16_t vbe_interface_len;
193:
194:     multiboot_uint64_t framebuffer_addr;
195:     multiboot_uint32_t framebuffer_pitch;
196:     multiboot_uint32_t framebuffer_width;
197:     multiboot_uint32_t framebuffer_height;

```

include/fiwix/multiboot1.h

Page 4/6

```

198:  multiboot_uint8_t framebuffer_bpp;
199: #define MULTIBOOT_FRAMEBUFFER_TYPE_INDEXED 0
200: #define MULTIBOOT_FRAMEBUFFER_TYPE_RGB 1
201: #define MULTIBOOT_FRAMEBUFFER_TYPE_EGA_TEXT 2
202:  multiboot_uint8_t framebuffer_type;
203:  union
204:  {
205:      struct
206:      {
207:          multiboot_uint32_t framebuffer_palette_addr;
208:          multiboot_uint16_t framebuffer_palette_num_colors;
209:      };
210:      struct
211:      {
212:          multiboot_uint8_t framebuffer_red_field_position;
213:          multiboot_uint8_t framebuffer_red_mask_size;
214:          multiboot_uint8_t framebuffer_green_field_position;
215:          multiboot_uint8_t framebuffer_green_mask_size;
216:          multiboot_uint8_t framebuffer_blue_field_position;
217:          multiboot_uint8_t framebuffer_blue_mask_size;
218:      };
219:  };
220: };
221: typedef struct multiboot_info multiboot_info_t;
222:
223: struct multiboot_color
224: {
225:  multiboot_uint8_t red;
226:  multiboot_uint8_t green;
227:  multiboot_uint8_t blue;
228: };
229:
230: struct multiboot_mmap_entry
231: {
232:  multiboot_uint32_t size;
233:  multiboot_uint64_t addr;
234:  multiboot_uint64_t len;
235: #define MULTIBOOT_MEMORY_AVAILABLE 1
236: #define MULTIBOOT_MEMORY_RESERVED 2
237: #define MULTIBOOT_MEMORY_ACPI_RECLAIMABLE 3
238: #define MULTIBOOT_MEMORY_NVS 4
239: #define MULTIBOOT_MEMORY_BADRAM 5
240:  multiboot_uint32_t type;
241: } __attribute__((packed));
242: typedef struct multiboot_mmap_entry multiboot_memory_map_t;
243:
244: struct multiboot_mod_list
245: {
246:  /* the memory used goes from bytes â\200\231mod_startâ\200\231 to â\200\231mod
_end-1â\200\231 inclusive */
247:  multiboot_uint32_t mod_start;
248:  multiboot_uint32_t mod_end;
249:
250:  /* Module command line */
251:  multiboot_uint32_t cmdline;
252:
253:  /* padding to take it to 16 bytes (must be zero) */
254:  multiboot_uint32_t pad;
255: };
256: typedef struct multiboot_mod_list multiboot_module_t;
257:
258: /* APM BIOS info. */
259: struct multiboot_apm_info
260: {
261:  multiboot_uint16_t version;
262:  multiboot_uint16_t cseg;
263:  multiboot_uint32_t offset;

```

include/fiwix/multiboot1.h

Page 5/6

```

264:  multiboot_uint16_t cseg_16;
265:  multiboot_uint16_t dseg;
266:  multiboot_uint16_t flags;
267:  multiboot_uint16_t cseg_len;
268:  multiboot_uint16_t cseg_16_len;
269:  multiboot_uint16_t dseg_len;
270: };
271:
272: /* VBE controller information. */
273: struct vbe_controller
274: {
275:     unsigned char signature[4];
276:     unsigned short version;
277:     unsigned long oem_string;
278:     unsigned long capabilities;
279:     unsigned long video_mode;
280:     unsigned short total_memory;
281:     unsigned short oem_software_rev;
282:     unsigned long oem_vendor_name;
283:     unsigned long oem_product_name;
284:     unsigned long oem_product_rev;
285:     unsigned char reserved[222];
286:     unsigned char oem_data[256];
287: } __attribute__ ((packed));
288:
289: /* VBE mode information. */
290: struct vbe_mode
291: {
292:     unsigned short mode_attributes;
293:     unsigned char win_a_attributes;
294:     unsigned char win_b_attributes;
295:     unsigned short win_granularity;
296:     unsigned short win_size;
297:     unsigned short win_a_segment;
298:     unsigned short win_b_segment;
299:     unsigned long win_func;
300:     unsigned short bytes_per_scanline;
301:
302:     /* >=1.2 */
303:     unsigned short x_resolution;
304:     unsigned short y_resolution;
305:     unsigned char x_char_size;
306:     unsigned char y_char_size;
307:     unsigned char number_of_planes;
308:     unsigned char bits_per_pixel;
309:     unsigned char number_of_banks;
310:     unsigned char memory_model;
311:     unsigned char bank_size;
312:     unsigned char number_of_image_pages;
313:     unsigned char reserved0;
314:
315:     /* direct color */
316:     unsigned char red_mask_size;
317:     unsigned char red_field_position;
318:     unsigned char green_mask_size;
319:     unsigned char green_field_position;
320:     unsigned char blue_mask_size;
321:     unsigned char blue_field_position;
322:     unsigned char reserved_mask_size;
323:     unsigned char reserved_field_position;
324:     unsigned char direct_color_mode_info;
325:
326:     /* >=2.0 */
327:     unsigned long phys_base;
328:     unsigned long reserved1;
329:     unsigned short reserved2;
330:

```

include/fiwix/multiboot1.h

Page 6/6

```
331:  /* >=3.0 */
332:  unsigned short linear_bytes_per_scanline;
333:  unsigned char  banked_number_of_image_pages;
334:  unsigned char  linear_number_of_image_pages;
335:  unsigned char  linear_red_mask_size;
336:  unsigned char  linear_red_field_position;
337:  unsigned char  linear_green_mask_size;
338:  unsigned char  linear_green_field_position;
339:  unsigned char  linear_blue_mask_size;
340:  unsigned char  linear_blue_field_position;
341:  unsigned char  linear_reserved_mask_size;
342:  unsigned char  linear_reserved_field_position;
343:  unsigned long max_pixel_clock;
344:
345:  unsigned char reserved3[190];
346: } __attribute__ ((packed));
347:
348:
349: #endif /* ! ASM_FILE */
350:
351: #endif /* ! MULTIBOOT_HEADER */
```

include/fiwix/part.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/part.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_PART_H
9: #define _FIWIX_PART_H
10:
11: #define PARTITION_BLOCK          0
12: #define NR_PARTITIONS           4      /* partitions in the MBR */
13: #define MBR_CODE_SIZE           446
14: #define ACTIVE_PART              0x80
15:
16: struct hd_geometry {
17:     unsigned char heads;
18:     unsigned char sectors;
19:     unsigned short int cylinders;
20:     unsigned long int start;
21: };
22:
23: struct partition {
24:     unsigned char status;
25:     unsigned char head;
26:     unsigned char sector;
27:     unsigned char cyl;
28:     unsigned char type;
29:     unsigned char endhead;
30:     unsigned char endsector;
31:     unsigned char endcyl;
32:     unsigned int startsect;
33:     unsigned int nr_sects;
34: };
35:
36: int read_msdos_partition(__dev_t, struct partition *);
37:
38: #endif /* _FIWIX_PART_H */
```

include/fiwix/pic.h

Page 1/1

```

1: /*
2:  * fiwix/include/fiwix/pic.h
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_PIC_H
9: #define _FIWIX_PIC_H
10:
11: #include <fiwix/sigcontext.h>
12:
13: #define NR_IRQS          16          /* hardware interrupts */
14: #define PIC_MASTER       0x20       /* I/O base address for master PIC */
15: #define PIC_SLAVE       0xA0       /* I/O base address for slave PIC */
16:
17: #define DATA            0x01       /* offset to data port */
18: #define EOI              0x20       /* End-Of-Interrupt command code */
19:
20: /* Inicialization Command Words */
21: #define ICW1_RESET       0x11       /* ICW1_INIT + ICW1_ICW4 */
22: #define CASCADE_IRQ      0x02
23: #define ICW4_8086EOI     0x01
24:
25: #define PIC_READ_IRR     0x0A       /* OCW3 irq ready */
26: #define PIC_READ_ISR     0x0B       /* OCW3 irq service */
27:
28: /* Operational Command Words */
29: #define OCW1             0xFF       /* mask (disable) all IRQs */
30:
31: struct interrupt {
32:     unsigned int ticks;
33:     char *name;
34:     void (*handler)(int, struct sigcontext *);
35:     struct interrupt *next;
36: };
37: extern struct interrupt *irq_table[NR_IRQS];
38:
39:
40: #define BH_ACTIVE        0x01
41:
42: struct bh {
43:     int flags;
44:     void (*fn)(void);
45:     struct bh *next;
46: };
47:
48: void add_bh(struct bh *);
49: void enable_irq(int);
50: void disable_irq(int);
51: int register_irq(int, struct interrupt *);
52: int unregister_irq(int, struct interrupt *);
53: void irq_handler(int, struct sigcontext);
54: void unknown_irq_handler(void);
55: void do_bh(void);
56: void pic_init(void);
57:
58: #endif /* _FIWIX_PIC_H */

```

include/fiwix/pit.h

Page 1/1

```

1: /*
2:  * fiwix/include/fiwix/pit.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_PIT_H
9: #define _FIWIX_PIT_H
10:
11: /* Intel 8253/82c54 Programmable Interval Timer */
12:
13: #define OSCIL          1193182 /* oscillator frequency */
14:
15: #define MODEREG        0x43 /* mode/command register (w) */
16: #define CHANNEL0      0x40 /* channel 0 data port (rw) */
17: #define CHANNEL1      0x41 /* channel 1 data port (rw) */
18: #define CHANNEL2      0x42 /* channel 2 data port (rw) */
19:
20: #define BINARY_CTR    0x00 /* 16bit binary mode counter */
21: #define TERM_COUNT    0x00 /* mode 0 (Terminal Count) */
22: #define RATE_GEN      0x04 /* mode 2 (Rate Generator) */
23: #define SQUARE_WAVE  0x06 /* mode 3 (Square Wave) */
24: #define LSB_MSB       0x30 /* LSB then MSB */
25: #define SEL_CHAN0     0x00 /* select channel 0 */
26: #define SEL_CHAN2     0x80 /* select channel 2 */
27:
28: /*
29:  * PS/2 System Control Port B
30:  * -----
31:  * bit 7 -> IRQ=1, 0=reset
32:  * bit 6 -> reserved
33:  * bit 5 -> reserved
34:  * bit 4 -> reserved
35:  * bit 3 -> channel check enable
36:  * bit 2 -> parity check enable
37:  * bit 1 -> speaker data enable
38:  * bit 0 -> timer 2 gate to speaker enable
39:  */
40: #define PS2_SYCTRL_B  0x61 /* PS/2 system control port B (write) */
41:
42: #define ENABLE_TMR2G  0x01 /* timer 2 gate to speaker enable */
43: #define ENABLE_SDATA  0x02 /* speaker data enable */
44:
45: #define BEEP_FREQ     900 /* 900Hz */
46:
47: void pit_beep_on(void);
48: void pit_beep_off(unsigned int);
49: void pit_init(unsigned short int);
50:
51: #endif /* _FIWIX_PIT_H */

```

include/fiwix/process.h

Page 1/3

```

1: /*
2:  * fiwix/include/fiwix/process.h
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_PROCESS_H
9: #define _FIWIX_PROCESS_H
10:
11: struct vma {
12:     unsigned int start;
13:     unsigned int end;
14:     char prot; /* PROT_READ, PROT_WRITE, ... */
15:     unsigned int flags; /* MAP_SHARED, MAP_PRIVATE, ... */
16:     unsigned int offset;
17:     char s_type; /* section type (P_TEXT, P_DATA, ...) */
18:     struct inode *inode; /* file inode */
19:     char o_mode; /* open mode (O_RDONLY, O_RDWR, ...) */
20: };
21:
22: #include <fiwix/config.h>
23: #include <fiwix/types.h>
24: #include <fiwix/signal.h>
25: #include <fiwix/limits.h>
26: #include <fiwix/sigcontext.h>
27: #include <fiwix/time.h>
28: #include <fiwix/resource.h>
29: #include <fiwix/tty.h>
30:
31: #define IDLE 0 /* PID of idle */
32: #define INIT 1 /* PID of /sbin/init */
33: #define SAFE_SLOTS 2 /* process slots reserved for root */
34: #define SLOT(p) ((p) - (&proc_table[0]))
35:
36: /* bits in flags */
37: #define PF_KPROC 0x00000001 /* kernel internal process */
38: #define PF_PEXEC 0x00000002 /* has performed a sys_execve() */
39: #define PF_USERREAL 0x00000004 /* use real UID in permission checks */
40:
41: #define MMAP_START 0x40000000 /* mmap()'s start at 1GB */
42: #define IS_SUPERUSER (current->euid == 0)
43:
44: #define IO_BITMAP_SIZE 32 /* 32 * 32bit = 1024 = 0x3FF */
45: /* 2048 * 32bit = 65536 = 0xFFFF */
46:
47: #define PG_LEADER(p) ((p)->pid == (p)->pgid)
48: #define SESS_LEADER(p) ((p)->pid == (p)->pgid && (p)->pid == (p)->sid)
49:
50: #define FOR_EACH_PROCESS(p) p = proc_table_head->next ; while(p)
51: #define FOR_EACH_PROCESS_RUNNING(p) p = proc_run_head ; while(p)
52:
53: /* value to be determined during system startup */
54: extern unsigned int proc_table_size; /* size in bytes */
55:
56: extern char any_key_to_reboot;
57: extern int nr_processes;
58: extern __pid_t lastpid;
59: extern struct proc *proc_table_head;
60:
61: struct binargs {
62:     unsigned int page[ARG_MAX];
63:     int argc;
64:     int argv_len;
65:     int envc;
66:     int envp_len;
67:     int offset;

```


include/fiwix/process.h

Page 2/3

```

68: };
69:
70: /* Intel 386 Task Switch State */
71: struct i386tss {
72:     unsigned int prev_tss;
73:     unsigned int esp0;
74:     unsigned int ss0;
75:     unsigned int esp1;
76:     unsigned int ssl;
77:     unsigned int esp2;
78:     unsigned int ss2;
79:     unsigned int cr3;
80:     unsigned int eip;
81:     unsigned int eflags;
82:     unsigned int eax;
83:     unsigned int ecx;
84:     unsigned int edx;
85:     unsigned int ebx;
86:     unsigned int esp;
87:     unsigned int ebp;
88:     unsigned int esi;
89:     unsigned int edi;
90:     unsigned int es;
91:     unsigned int cs;
92:     unsigned int ss;
93:     unsigned int ds;
94:     unsigned int fs;
95:     unsigned int gs;
96:     unsigned int ldt;
97:     unsigned short int debug_trap;
98:     unsigned short int io_bitmap_addr;
99: };
100:
101: struct proc {
102:     struct i386tss tss;
103:     unsigned int io_bitmap[IO_BITMAP_SIZE + 1];
104:     __pid_t pid; /* process ID */
105:     __pid_t ppid; /* parent process ID */
106:     __pid_t pgid; /* process group ID */
107:     __pid_t sid; /* session ID */
108:     int flags;
109:     int groups[NGROUPS_MAX];
110:     int children; /* number of children */
111:     struct tty *ctty; /* controlling terminal */
112:     int state; /* process state */
113:     int priority;
114:     int cpu_count; /* time of process running */
115:     __time_t start_time;
116:     int exit_code;
117:     void *sleep_address;
118:     unsigned short int uid; /* real user ID */
119:     unsigned short int gid; /* real group ID */
120:     unsigned short int euid; /* effective user ID */
121:     unsigned short int egid; /* effective group ID */
122:     unsigned short int suid; /* saved user ID */
123:     unsigned short int sgid; /* saved group ID */
124:     unsigned short int fd[OPEN_MAX];
125:     unsigned char fd_flags[OPEN_MAX];
126:     struct inode *root;
127:     struct inode *pwd; /* process working directory */
128:     unsigned int entry_address;
129:     char argv0[NAME_MAX + 1];
130:     int argc;
131:     char **argv;
132:     int envc;
133:     char **envp;
134:     char pidstr[5]; /* pid number converted to string */

```

include/fiwix/process.h

Page 3/3

```

135:     struct vma vma[VMA_REGIONS];      /* virtual memory-map addresses */
136:     unsigned int brk_lower;           /* lower limit of the heap section */
137:     unsigned int brk;                 /* current limit of the heap */
138:     __sigset_t sigpending;
139:     __sigset_t sigblocked;
140:     __sigset_t sigexecuting;
141:     struct sigaction sigaction[NSIG];
142:     struct sigcontext sc[NSIG];      /* each signal has its own context */
143:     unsigned int sp;                 /* current process' stack frame */
144:     struct rusage usage;             /* process resource usage */
145:     struct rusage cusage;           /* children resource usage */
146:     unsigned long int it_real_interval, it_real_value;
147:     unsigned long int it_virt_interval, it_virt_value;
148:     unsigned long int it_prof_interval, it_prof_value;
149:     unsigned long int timeout;
150:     struct rlimit rlim[RLIM_NLIMITS];
151:     unsigned long int rss;
152:     __mode_t umask;
153:     unsigned char loopcnt;           /* nested symlinks counter */
154:     struct proc *prev;
155:     struct proc *next;
156:     struct proc *prev_sleep;
157:     struct proc *next_sleep;
158:     struct proc *prev_run;
159:     struct proc *next_run;
160: };
161:
162: extern struct proc *current;
163: extern struct proc *proc_table;
164:
165: int send_sig(struct proc *, __sigset_t);
166: int kill_pid(__pid_t, __sigset_t);
167: int kill_pgrp(__pid_t, __sigset_t);
168: void add_crusage(struct proc *, struct rusage *);
169: void get_rusage(struct proc *, struct rusage *);
170: void add_rusage(struct proc *);
171: struct proc * get_next_zombie(struct proc *);
172: __pid_t remove_zombie(struct proc *);
173: int is_orphaned_pgrp(__pid_t);
174: struct proc * get_proc_free(void);
175: void release_proc(struct proc *);
176: int get_unused_pid(void);
177: struct proc * get_proc_by_pid(__pid_t);
178:
179: int get_new_user_fd(int);
180: void release_user_fd(int);
181:
182: struct proc * kernel_process(const char *, int (*fn)(void));
183: void proc_slot_init(struct proc *);
184: void proc_init(void);
185:
186: int elf_load(struct inode *, struct binargs *, struct sigcontext *, char *);
187: int script_load(char *, char *, char *);
188:
189: #endif /* _FIWIX_PROCESS_H */

```

include/fiwix/ramdisk.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/ramdisk.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_RAMDISK_H
9: #define _FIWIX_RAMDISK_H
10:
11: #include <fiwix/fs.h>
12:
13: #define RAMDISK_MAJOR 1      /* ramdisk device major number */
14: #define RAMDISK_MINORS 1    /* number of minors */
15: #define RAMDISK_SIZE 4096   /* default ramdisk size in KBs */
16: #define RAMDISK_MAXSIZE 131072 /* maximum ramdisk size in KBs */
17:
18: struct ramdisk {
19:     char *addr;              /* ramdisk memory address */
20: };
21:
22: extern struct ramdisk ramdisk_table[RAMDISK_MINORS];
23:
24: int ramdisk_open(struct inode *, struct fd *);
25: int ramdisk_close(struct inode *, struct fd *);
26: int ramdisk_read(__dev_t, __blk_t, char *, int);
27: int ramdisk_write(__dev_t, __blk_t, char *, int);
28: int ramdisk_ioctl(struct inode *, int, unsigned long int);
29: int ramdisk_lseek(struct inode *, __off_t);
30:
31: void ramdisk_init(void);
32:
33: #endif /* _FIWIX_RAMDISK_H */
```

include/fiwix/reboot.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/reboot.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_REBOOT_H
9: #define _FIWIX_REBOOT_H
10:
11: #define BMAGIC_HARD      0x89ABCDEF
12: #define BMAGIC_SOFT     0
13: #define BMAGIC_REBOOT   0x01234567
14: #define BMAGIC_HALT     0xCDEF0123
15: #define BMAGIC_POWEROFF 0x4321FEDC
16:
17: #define BMAGIC_1        0xFEE1DEAD
18: #define BMAGIC_2        672274793
19:
20: extern char ctrl_alt_del;
21: void reboot(void);
22:
23: #endif /* _FIWIX_REBOOT_H */
```

include/fiwix/resource.h

Page 1/2

```

1:  /*
2:  * fiwix/include/fiwix/resource.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_RESOURCE_H
9: #define _FIWIX_RESOURCE_H
10:
11: #include <fiwix/time.h>
12:
13: #define RLIMIT_INFINITY 0x7FFFFFFF /* value to indicate "no limit" */
14: #define RLIM_INFINITY RLIMIT_INFINITY /* traditional name */
15:
16: #define RUSAGE_SELF 0 /* the calling process */
17: #define RUSAGE_CHILDREN (-1) /* all of its termin. child processes */
18:
19: #define RLIMIT_CPU 0 /* per-process CPU limit (secs) */
20: #define RLIMIT_FSIZE 1 /* largest file that can be created
21: (bytes) */
22: #define RLIMIT_DATA 2 /* maximum size of data segment
23: (bytes) */
24: #define RLIMIT_STACK 3 /* maximum size of stack segment
25: (bytes) */
26: #define RLIMIT_CORE 4 /* largest core file that can be created
27: (bytes) */
28: #define RLIMIT_RSS 5 /* largest resident set size (bytes) */
29: #define RLIMIT_NPROC 6 /* number of processes */
30: #define RLIMIT_NOFILE 7 /* number of open files */
31: #define RLIMIT_MEMLOCK 8 /* locked-in-memory address space */
32: #define RLIMIT_AS 9 /* address space limit */
33:
34: #define RLIM_NLIMITS 10
35:
36: struct rusage {
37:     struct timeval ru_utime; /* total amount of user time used */
38:     struct timeval ru_stime; /* total amount of system time used */
39:     long ru_maxrss; /* maximum resident set size (KB) */
40:     long ru_ixrss; /* amount of sharing of text segment
41: memory with other processes
42: (KB-secs) */
43:     long ru_idrss; /* amount of data segment memory used
44: (KB-secs) */
45:     long ru_isrss; /* amount of stack memory used
46: (KB-secs) */
47:     long ru_minflt; /* number of soft page faults (i.e.
48: those serviced by reclaiming a page
49: from the list of pages awaiting
50: relocation) */
51:     long ru_majflt; /* number of hard page faults (i.e.
52: those that required I/O) */
53:     long ru_nswap; /* number of times a process was swapped
54: out of physical memory */
55:     long ru_inblock; /* number of input operations via the
56: file system. Note this and
57: 'ru_outblock' do not include
58: operations with the cache */
59:     long ru_oublock; /* number of output operations via the
60: file system */
61:     long ru_msgsnd; /* number of IPC messages sent */
62:     long ru_msgrcv; /* number of IPC messages received */
63:     long ru_nsignals; /* number of signals delivered */
64:     long ru_nvcsw; /* number of voluntary context switches,
65: i.e. because the process gave up the
66: process before it had to (usually to
67: wait for some resource to be

```

include/fiwix/resource.h

Page 2/2

```
68:                                     availabe */
69:         long ru_nivcsw;                /* number of involuntary context
70:                                     switches. i.e. a higher priority
71:                                     process became runnable or the
72:                                     current process used up its time
73:                                     slice */
74: };
75:
76: struct rlimit {
77:     int rlim_cur;                       /* the current (soft) limit */
78:     int rlim_max;                       /* the maximum (hard) limit */
79: };
80:
81: #endif /* _FIWIX_RESOURCE_H */
```

include/fiwix/sched.h

Page 1/1

```

1: /*
2:  * fiwix/include/fiwix/sched.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_SCHED_H
9: #define _FIWIX_SCHED_H
10:
11: #include <fiwix/process.h>
12:
13: #define PRIO_PROCESS    0
14: #define PRIO_PGRP      1
15: #define PRIO_USER      2
16:
17: #define PROC_RUNNING    1
18: #define PROC_SLEEPING  2
19: #define PROC_ZOMBIE    3
20: #define PROC_STOPPED   4
21: #define PROC_IDLE      5
22:
23: #define PROC_INTERRUPTIBLE  1
24: #define PROC_UNINTERRUPTIBLE 2
25:
26: #define DEF_PRIORITY      (20 * HZ / 100) /* 200ms of time slice */
27:
28: extern int need_resched;
29:
30: #define SI_LOAD_SHIFT    16
31:
32: /*
33:  * This was brought from Linux 2.0.30 (sched.h).
34:  * Copyright Linus Torvalds et al.
35:  */
36: extern unsigned int avenrun[3];          /* Load averages */
37: #define FSHIFT          11              /* nr of bits of precision */
38: #define FIXED_1         (1<<FSHIFT)    /* 1.0 as fixed-point */
39: #define LOAD_FREQ       (5*HZ)         /* 5 sec intervals */
40: #define EXP_1           1884           /* 1/exp(5sec/1min) as fixed-point */
41: #define EXP_5           2014           /* 1/exp(5sec/5min) */
42: #define EXP_15          2037           /* 1/exp(5sec/15min) */
43:
44: #define CALC_LOAD(load,exp,n) \
45:     load *= exp; \
46:     load += n*(FIXED_1-exp); \
47:     load >>= FSHIFT;
48: /* ----- */
49:
50:
51: void do_sched(void);
52: void set_tss(struct proc *);
53: void sched_init(void);
54:
55: #endif /* _FIWIX_SCHED_H */

```

include/fiwix/segments.h

Page 1/2

```

1: /*
2:  * fiwix/include/fiwix/segments.h
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_SEGMENTS_H
9: #define _FIWIX_SEGMENTS_H
10:
11: #define KERNEL_BASE_ADDR      0xC0000000
12: #define KERNEL_ENTRY_ADDR    0x100000
13:
14: #define KERNEL_CS             0x08    /* kernel code segment */
15: #define KERNEL_DS             0x10    /* kernel data segment */
16: #define USER_CS               0x18    /* user code segment */
17: #define USER_DS               0x20    /* user data segment */
18: #define TSS                   0x28    /* TSS segment */
19:
20:
21: #ifndef ASM_FILE
22:
23: #include <fiwix/types.h>
24:
25: #define NR_GDT_ENTRIES        6        /* entries in GDT descriptor */
26: #define NR_IDT_ENTRIES        256     /* entries in IDT descriptor */
27:
28: /* low flags of Segment Descriptors */
29: #define SD_DATA                0x02    /* DATA Read/Write */
30: #define SD_CODE                0x0A    /* CODE Exec/Read */
31:
32: #define SD_32INTRGATE          0x0E    /* 32-bit Interrupt Gate (0D110) */
33: #define SD_32TRAPGATE          0x0F    /* 32-bit Trap Gate (0D111) */
34:
35: #define SD_CD                  0x10    /* 0 = system / 1 = code/data */
36: #define SD_DPL0                0x00    /* priority level 0 (kernel) */
37: #define SD_DPL3                0x60    /* priority level 3 (user) */
38: #define SD_PRESENT             0x80    /* segment present or valid */
39:
40: /* high flags Segment Descriptors */
41: #define SD_OPSIZE32            0x04    /* 32-bit code and data segments */
42: #define SD_PAGE4KB             0x08    /* page granularity (4KB) */
43:
44: /* low flags of the TSS Descriptors */
45: #define SD_TSSPRESENT          0x89    /* TSS present and not busy flag */
46:
47: #define USR_PL                  3        /* User Privilege Level */
48:
49: /* EFLAGS */
50: #define EF_IOPL                 12     /* IOPL bit */
51:
52: struct desc_r {
53:     __u16 limit;
54:     __u32 base_addr;
55: } __attribute__((packed));
56:
57: struct seg_desc {
58:     unsigned sd_lolimit : 16;    /* segment limit 0-15 bits */
59:     unsigned sd_lobase  : 24;    /* base address 0-23 bits */
60:     unsigned sd_loflags : 8;     /* flags (P, DPL, S and TYPE) */
61:     unsigned sd_hilimit : 4;     /* segment limit 16-19 bits */
62:     unsigned sd_hiflags : 4;     /* flags (G, DB, 0 and AVL) */
63:     unsigned sd_hibase  : 8;     /* base address 24-31 bits */
64: } __attribute__((packed));
65:
66: struct gate_desc {
67:     unsigned gd_looffset: 16;    /* offset 0-15 bits */

```


include/fiwix/segments.h

Page 2/2

```
68:         unsigned gd_selector: 16;          /* segment selector */
69:         unsigned gd_flags   : 16;          /* flags (P, DPL, TYPE, 0 and NULL) */
70:         unsigned gd_hioffset: 16;          /* offset 16-31 bits */
71: } __attribute__((packed));
72:
73: void gdt_init(void);
74: void idt_init(void);
75:
76: #endif /* ! ASM_FILE */
77:
78: #endif /* _FIWIX_SEGMENTS_H */
```

include/fiwix/serial.h

Page 1/2

```

1: /*
2:  * fiwix/include/fiwix/serial.h
3:  *
4:  * Copyright 2020-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_SERIAL_H
9: #define _FIWIX_SERIAL_H
10:
11: #define SERIAL4_IRQ    4    /* IRQ for serial ports 1 and 3 */
12: #define SERIAL3_IRQ    3    /* IRQ for serial ports 2 and 4 */
13:
14: #define NR_SERIAL      4    /* maximum number of serial ttys */
15: #define SERIAL_MAJOR   4    /* major number for /dev/ttyS[n] */
16: #define SERIAL_MINORS  NR_SERIAL
17: #define SERIAL_MSF     6    /* serial minor shift factor */
18:
19: /* UART registers */
20: #define UART_TD        0    /* W: Transmitter Holding Buffer */
21: #define UART_RD        0    /* R: Receiver Buffer */
22: #define UART_DLL       0    /* RW: Divisor Latch Low Byte */
23: #define UART_DLH       1    /* RW: Divisor Latch High Byte */
24: #define UART_IER       1    /* RW: Interrupt Enable Register */
25: #define UART_IIR       2    /* R: Interrupt Identification Register */
26: #define UART_FCR       2    /* W: FIFO Control Register */
27: #define UART_LCR       3    /* RW: Line Control Register */
28: #define UART_MCR       4    /* RW: Modem Control Register */
29: #define UART_LSR       5    /* R: Line Status Register */
30: #define UART_MSR       6    /* R: Modem Status Register */
31: #define UART_SR       7    /* RW: Scratch Register */
32:
33: /* Interrupt Enable Register */
34: #define UART_IER_RDAI  0x1   /* enable Received Data Available Interrupt */
35: #define UART_IER_THREI 0x2   /* enable Transmitter Holding Register Empty Interrupt */
36: #define UART_IER_RLSI  0x4   /* enable Receiver Line Status Interrupt */
37: #define UART_IER_MSI   0x8   /* enable Modem Status Interrupt */
38:
39: /* Interrupt Identification Register */
40: #define UART_IIR_NOINT 0x01  /* no interrupts pending */
41: #define UART_IIR_MKINT 0x06  /* mask all interrupt flags */
42: #define UART_IIR_MSI   0x00  /* Modem Status Interrupt */
43: #define UART_IIR_THREI 0x02  /* Transmitter Holding Register Empty Interrupt */
44: #define UART_IIR_RDAI  0x04  /* Received Data Available Interrupt */
45: #define UART_IIR_RLSI  0x06  /* Receiver Line Status Interrupt */
46: #define UART_IIR_FIFOTO 0xC0  /* FIFO TimeOut interrupt */
47: #define UART_IIR_FIFO64 0x20  /* 64 byte FIFO enabled (16750 only) */
48: #define UART_IIR_FIFO  0x40  /* FIFO is enabled (still needs bit #7 on) */
49: #define UART_IIR_FIFOKO 0x80  /* FIFO is enabled, but unusable */
50:
51: /* FIFO Control Register */
52: #define UART_FCR_FIFO  0x07  /* enable FIFO (clear receive and transmit) */
53: #define UART_FCR_CRCVR 0x02  /* clear receiver */
54: #define UART_FCR_CXMTR 0x04  /* clear transmitter */
55: #define UART_FCR_DMA   0x08  /* DMA mode select */
56: #define UART_FCR_FIFO64 0x20  /* enable 64 byte FIFO (16750 only) */
57: #define UART_FCR_FIFO14 0xC0  /* set to 14 bytes 'trigger level' FIFO */
58:
59: /* Line Control Register */
60: #define UART_LCR_WL5   0x00  /* word length 5 bits */
61: #define UART_LCR_WL6   0x01  /* word length 6 bits */
62: #define UART_LCR_WL7   0x02  /* word length 7 bits */
63: #define UART_LCR_WL8   0x03  /* word length 8 bits */
64: #define UART_LCR_2STB  0x04  /* 2 stop bits */
65: #define UART_LCR_1STB  0x00  /* 1 stop bit */

```

include/fiwix/serial.h

Page 2/2

```

66: #define UART_LCR_NP      0x00    /* no parity */
67: #define UART_LCR_OP      0x08    /* odd parity */
68: #define UART_LCR_EP      0x18    /* even parity */
69: #define UART_LCR_SBRK    0x40    /* Set Break enable */
70: #define UART_LCR_DLAB    0x80    /* Divisor Latch Access Bit */
71:
72: /* Modem Control Register */
73: #define UART_MCR_DTR     0x1     /* Data Terminal Ready */
74: #define UART_MCR_RTS     0x2     /* Request To Send */
75: #define UART_MCR_OUT2    0x8     /* Auxiliary Output 2 */
76:
77: /* Line Status Register */
78: #define UART_LSR_RDA     0x01    /* Received Data Available */
79: #define UART_LSR_OE      0x02    /* Overrun Error */
80: #define UART_LSR_PE      0x04    /* Parity Error */
81: #define UART_LSR_FE      0x08    /* Framing Error */
82: #define UART_LSR_BI      0x10    /* Break Interrupt */
83: #define UART_LSR_THRE    0x20    /* Transmitter Holding Register Empty */
84: #define UART_LSR_EDHR    0x40    /* Empty Data Holding Registers TD and SH */
85: #define UART_LSR_EFIFO   0x80    /* Error in Received FIFO */
86:
87:
88: #define UART_FIFO_SIZE   16      /* 16 bytes */
89: #define UART_HAS_FIFO    0x02    /* has FIFO working */
90: #define UART_IS_8250     0x04    /* is a 8250 chip */
91: #define UART_IS_16450    0x08    /* is a 16450 chip */
92: #define UART_IS_16550    0x10    /* is a 16550 chip */
93: #define UART_IS_16550A   0x20    /* is a 16550A chip */
94:
95: struct serial {
96:     short int addr;          /* port I/O address */
97:     char irq;
98:     int baud;
99:     char *name;
100:    short int lctrl;         /* line control flags (8N1, 7E2, ...) */
101:    int flags;
102:    struct tty *tty;
103:    struct serial *next;
104: };
105:
106: int serial_open(struct tty *);
107: int serial_close(struct tty *);
108: int serial_ioctl(struct tty *, int, unsigned long int);
109: void serial_write(struct tty *);
110: void irq_serial(int, struct sigcontext *);
111: void irq_serial_bh(void);
112: void serial_init(void);
113:
114: #endif /* _FIWIX_SERIAL_H */

```

include/fiwix/sigcontext.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/sigcontext.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_SIGCONTEXT_H
9: #define _FIWIX_SIGCONTEXT_H
10:
11: struct sigcontext {
12:     unsigned int gs;
13:     unsigned int fs;
14:     unsigned int es;
15:     unsigned int ds;
16:     unsigned int edi;
17:     unsigned int esi;
18:     unsigned int ebp;
19:     unsigned int esp;
20:     int ebx;
21:     int edx;
22:     int ecx;
23:     int eax;
24:     int err;
25:     unsigned int eip;
26:     unsigned int cs;
27:     unsigned int eflags;
28:     unsigned int oldesp;
29:     unsigned int oldss;
30: };
31:
32: #endif /* _FIWIX_SIGCONTEXT_H */
```

include/fiwix/signal.h

Page 1/2

```

1: /*
2:  * fiwix/include/fiwix/signal.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_SIGNAL_H
9: #define _FIWIX_SIGNAL_H
10:
11: #define NSIG          32
12:
13: #define SIGHUP        1      /* Hangup or Reset */
14: #define SIGINT        2      /* Interrupt */
15: #define SIGQUIT       3      /* Quit */
16: #define SIGILL        4      /* Illegal Instruction */
17: #define SIGTRAP       5      /* Trace Trap */
18: #define SIGABRT       6      /* Abort Instruction */
19: #define SIGIOT        SIGABRT /* I/O Trap Instruction */
20: #define SIGBUS        7      /* Bus Error */
21: #define SIGFPE        8      /* Floating Point Exception */
22: #define SIGKILL       9      /* Kill */
23: #define SIGUSR1       10     /* User Defined #1 */
24: #define SIGSEGV       11     /* Segmentation Violation */
25: #define SIGUSR2       12     /* User Defined #2 */
26: #define SIGPIPE       13     /* Broken Pipe */
27: #define SIGALRM       14     /* Alarm Clock */
28: #define SIGTERM       15     /* Software Termination */
29: #define SIGSTKFLT     16     /* Stack Fault */
30: #define SIGCHLD       17     /* Child Termination */
31: #define SIGCONT       18     /* Continue */
32: #define SIGSTOP       19     /* Stop */
33: #define SIGTSTP       20     /* Terminal Stop */
34: #define SIGTTIN       21     /* Background Read */
35: #define SIGTTOU       22     /* Background Write */
36: #define SIGURG        23     /* Urgent Data */
37: #define SIGXCPU       24     /* CPU eXceeded */
38: #define SIGXFSZ       25     /* File Size eXceeded */
39: #define SIGVTALRM     26     /* Virtual Time Alarm */
40: #define SIGPROF       27     /* Profile Alarm */
41: #define SIGWINCH      28     /* Window Change */
42: #define SIGIO         29     /* I/O Asynchronous */
43: #define SIGPOLL       SIGIO
44: #define SIGPWR        30     /* Power Fault */
45: #define SIGUNUSED     31
46:
47: typedef unsigned long int __sigset_t;
48: typedef void (*__sighandler_t)(int);
49:
50: struct sigaction {
51:     __sighandler_t sa_handler;
52:     __sigset_t sa_mask;
53:     int sa_flags;
54:     void (*sa_restorer)(void);
55: };
56:
57: #define SIG_DFL        ((__sighandler_t) 0)
58: #define SIG_IGN        ((__sighandler_t) 1)
59: #define SIG_ERR        ((__sighandler_t) -1)
60:
61: /* bits in sa_flags */
62: #define SA_NOCLDSTOP   0x00000001 /* don't send SIGCHLD when children stop
*/
63: #define SA_NOCLDWAIT   0x00000002 /* don't create zombie on child death */
64: #define SA_ONSTACK     0x08000000 /* invoke handler on alternate stack */
65: #define SA_RESTART     0x10000000 /* automatically restart system call */
66: #define SA_INTERRUPT   0x20000000 /* unused */

```

include/fiwix/signal.h

Page 2/2

```
67:
68: /* don't automatically block signal when the handler is executing */
69: #define SA_NODEFER      0x40000000
70: #define SA_NOMASK      SA_NODEFER
71:
72: /* reset signal disposition to SIG_DFL before invoking handler */
73: #define SA_RESETHAND    0x80000000
74: #define SA_ONESHOT      SA_RESETHAND
75:
76: /* bits in the third argument to 'waitpid/wait4' */
77: #define WNOHANG         1      /* don't block waiting */
78: #define WUNTRACED      2      /* report status of stopped children */
79:
80: #define SIG_BLOCK       0      /* for blocking signals */
81: #define SIG_UNBLOCK    1      /* for unblocking signals */
82: #define SIG_SETMASK    2      /* for setting the signal mask */
83:
84: /* SIGKILL and SIGSTOP can't ever be set as blockable signals */
85: #define SIG_BLOCKABLE  (~(1 << (SIGKILL - 1)) | (1 << (SIGSTOP - 1)))
86:
87: #define SIG_MASK(sig)  (~(1 << ((sig) - 1)))
88:
89: int issig(void);
90: void psig(unsigned int);
91:
92: #endif /* _FIWIX_SIGNAL_H */
```

include/fiwix/sleep.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/sleep.h
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_SLEEP_H
9: #define _FIWIX_SLEEP_H
10:
11: #include <fiwix/process.h>
12:
13: #define AREA_BH 0x00000001
14: #define AREA_CALLOUT 0x00000002
15: #define AREA_TTY_READ 0x00000004
16: #define AREA_SERIAL_READ 0x00000008
17:
18: extern struct proc *proc_run_head;
19:
20: struct resource {
21:     char locked;
22:     char wanted;
23: };
24:
25: void runnable(struct proc *);
26: void not_runnable(struct proc *, int);
27: int sleep(void *, int);
28: void wakeup(void *);
29: void wakeup_proc(struct proc *);
30:
31: void lock_resource(struct resource *);
32: void unlock_resource(struct resource *);
33: int lock_area(unsigned int);
34: int unlock_area(unsigned int);
35:
36: void sleep_init(void);
37:
38: #endif /* _FIWIX_SLEEP_H */
```

include/fiwix/statbuf.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/statbuf.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_STATBUF_H
9: #define _FIWIX_STATBUF_H
10:
11: struct old_stat {
12:     __dev_t st_dev;
13:     unsigned short int st_ino;
14:     __mode_t st_mode;
15:     __nlink_t st_nlink;
16:     __uid_t st_uid;
17:     __gid_t st_gid;
18:     __dev_t st_rdev;
19:     unsigned int st_size;
20:     __time_t st_atime;
21:     __time_t st_mtime;
22:     __time_t st_ctime;
23: };
24:
25: struct new_stat {
26:     __dev_t st_dev;
27:     unsigned short int __pad1;
28:     __ino_t st_ino;
29:     __mode_t st_mode;
30:     __nlink_t st_nlink;
31:     __uid_t st_uid;
32:     __gid_t st_gid;
33:     __dev_t st_rdev;
34:     unsigned short int __pad2;
35:     __off_t st_size;
36:     __blk_t st_blksize;
37:     __blk_t st_blocks;
38:     __time_t st_atime;
39:     unsigned int __unused1;
40:     __time_t st_mtime;
41:     unsigned int __unused2;
42:     __time_t st_ctime;
43:     unsigned int __unused3;
44:     unsigned int __unused4;
45:     unsigned int __unused5;
46: };
47:
48: #endif /* _FIWIX_STATBUF_H */
```


include/fiwix/statfs.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/statfs.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_STATFS_H
9: #define _FIWIX_STATFS_H
10:
11: typedef struct {
12:     long int val[2];
13: } fsid_t;
14:
15: struct statfs {
16:     long int f_type;
17:     long int f_bsize;
18:     long int f_blocks;
19:     long int f_bfree;
20:     long int f_bavail;
21:     long int f_files;
22:     long int f_ffree;
23:     fsid_t f_fsid;
24:     long int f_namelen;
25:     long int f_spare[6];
26: };
27:
28: #endif /* _FIWIX_STATFS_H */
```

include/fiwix/stat.h

Page 1/1

```

1: #ifndef _FIWIX_STAT_H
2: #define _FIWIX_STAT_H
3:
4: #include <fiwix/statbuf.h>
5:
6: /* Encoding of the file mode.  These are the standard Unix values,
7:    but POSIX.1 does not specify what values should be used.  */
8:
9: #define S_IFMT          0170000      /* Type of file mask */
10:
11: /* File types.  */
12: #define S_IFIFO         0010000      /* Named pipe (fifo) */
13: #define S_IFCHR         0020000      /* Character special */
14: #define S_IFDIR         0040000      /* Directory */
15: #define S_IFBLK         0060000      /* Block special */
16: #define S_IFREG         0100000      /* Regular */
17: #define S_IFLNK         0120000      /* Symbolic link */
18: #define S_IFSOCK        0140000      /* Socket */
19:
20: /* Protection bits.  */
21: #define S_IXUSR         00100        /* USER  --x----- */
22: #define S_IWUSR         00200        /* USER  -w----- */
23: #define S_IRUSR         00400        /* USER  r----- */
24: #define S_IRWXU         00700        /* USER  rwx----- */
25:
26: #define S_IXGRP         00010        /* GROUP  -----x--- */
27: #define S_IWGRP         00020        /* GROUP  -----w--- */
28: #define S_IRGRP         00040        /* GROUP  ----r----- */
29: #define S_IRWXG         00070        /* GROUP  ---rwx--- */
30:
31: #define S_IXOTH         00001        /* OTHERS -----x */
32: #define S_IWOTH         00002        /* OTHERS -----w */
33: #define S_IROTH         00004        /* OTHERS -----r-- */
34: #define S_IRWXO         00007        /* OTHERS -----rwx */
35:
36: #define S_ISUID         0004000      /* set user id on execution */
37: #define S_ISGID         0002000      /* set group id on execution */
38: #define S_ISVTX         0001000      /* sticky bit */
39:
40: #define S_IREAD         S_IRUSR      /* Read by owner.  */
41: #define S_IWRITE        S_IWUSR      /* Write by owner.  */
42: #define S_IEXEC         S_IXUSR      /* Execute by owner.  */
43:
44: #define S_ISFIFO(m)     ((m) & S_IFMT) == S_IFIFO)
45: #define S_ISCHR(m)      ((m) & S_IFMT) == S_IFCHR)
46: #define S_ISDIR(m)      ((m) & S_IFMT) == S_IFDIR)
47: #define S_ISBLK(m)      ((m) & S_IFMT) == S_IFBLK)
48: #define S_ISREG(m)      ((m) & S_IFMT) == S_IFREG)
49: #define S_ISLNK(m)      ((m) & S_IFMT) == S_IFLNK)
50: #define S_ISSOCK(m)     ((m) & S_IFMT) == S_IFSOCK)
51:
52: #define TO_READ         4            /* test for read permission */
53: #define TO_WRITE        2            /* test for write permission */
54: #define TO_EXEC         1            /* test for execute permission */
55:
56: #endif /* _FIWIX_STAT_H */

```

include/fiwix/stdarg.h

Page 1/1

```
1: /*
2: Copyright (C) 1988 Free Software Foundation
3:
4: This file is part of GNU CC.
5:
6: GNU CC is distributed in the hope that it will be useful,
7: but WITHOUT ANY WARRANTY. No author or distributor
8: accepts responsibility to anyone for the consequences of using it
9: or for whether it serves any particular purpose or works at all,
10: unless he says so in writing. Refer to the GNU CC General Public
11: License for full details.
12:
13: Everyone is granted permission to copy, modify and redistribute
14: GNU CC, but only under the conditions described in the
15: GNU CC General Public License. A copy of this license is
16: supposed to have been given to you along with GNU CC so you
17: can know your rights and responsibilities. It should be in a
18: file named COPYING. Among other things, the copyright notice
19: and this notice must be preserved on all copies.
20: */
21:
22: #ifndef __stdarg_h
23: #define __stdarg_h
24:
25: typedef char *va_list;
26:
27: /* Amount of space required in an argument list for an arg of type TYPE.
28:    TYPE may alternatively be an expression whose type is used. */
29:
30: #define __va_rounded_size(TYPE) \
31:    (((sizeof (TYPE) + sizeof (int) - 1) / sizeof (int)) * sizeof (int))
32:
33: #define va_start(AP, LASTARG) \
34:    (AP = ((char *) &(LASTARG) + __va_rounded_size(LASTARG)))
35:
36: extern void va_end (va_list);
37: #define va_end(AP) /* Nothing */
38:
39: #define va_arg(AP, TYPE) (AP += __va_rounded_size (TYPE), \
40:    *((TYPE *) (AP - __va_rounded_size (TYPE))))
41:
42: #endif /* __stdarg_h */
```

include/fiwix/stdio.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/stdio.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _INCLUDE_STDIO_H
9: #define _INCLUDE_STDIO_H
10:
11: void register_console(void (*fn)(char *, unsigned int));
12: void printk(const char *, ...);
13: int sprintk(char *, const char *, ...);
14:
15: #endif /* _INCLUDE_STDIO_H */
```

include/fiwix/string.h

Page 1/1

```

1: /*
2:  * fiwix/include/fiwix/string.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _INCLUDE_STRING_H
9: #define _INCLUDE_STRING_H
10:
11: #include <fiwix/types.h>
12:
13: #ifndef NULL
14: #define NULL    '\0'    /* ((void *)0) */
15: #endif
16:
17: #define MIN(a,b)    ((a) < (b) ? (a) : (b))
18: #define MAX(a,b)    ((a) > (b) ? (a) : (b))
19:
20: #define IS_NUMERIC(c)    ((c) >= '0' && (c) <= '9')
21: #define IS_SPACE(c)    ((c) == ' ')
22:
23: void swap_asc_word(char *, int);
24: int strcmp(const char *, const char *);
25: int strncmp(const char *, const char *, __ssize_t);
26: char * strcpy(char *, const char *);
27: void strncpy(char *, const char *, int);
28: char * strcat(char *, const char *);
29: char * strncat(char *, const char *, __ssize_t);
30: int strlen(const char *);
31: char * get_basename(const char *);
32: char * remove_trailing_slash(char *);
33: int is_dir(const char *);
34: int atoi(const char *);
35: void memcpy_b(void *, const void *, unsigned int);
36: void memcpy_w(void *, const void *, unsigned int);
37: void memcpy_l(void *, const void *, unsigned int);
38: void memset_b(void *, unsigned char, unsigned int);
39: void memset_w(void *, unsigned short int, unsigned int);
40: void memset_l(void *, unsigned int, unsigned int);
41:
42: #endif /* _INCLUDE_STRING_H */

```

include/fiwix/syscalls.h

Page 1/3

```

1: /*
2:  * fiwix/include/fiwix/syscalls.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_SYSCALLS_H
9: #define _FIWIX_SYSCALLS_H
10:
11: #include <fiwix/types.h>
12: #include <fiwix/system.h>
13: #include <fiwix/time.h>
14: #include <fiwix/times.h>
15: #include <fiwix/timeb.h>
16: #include <fiwix/utime.h>
17: #include <fiwix/statbuf.h>
18: #include <fiwix/ustat.h>
19: #include <fiwix/signal.h>
20: #include <fiwix/utsname.h>
21: #include <fiwix/resource.h>
22: #include <fiwix/dirent.h>
23: #include <fiwix/statfs.h>
24: #include <fiwix/sigcontext.h>
25: #include <fiwix/mman.h>
26:
27: #define NR_SYSCALLS      (sizeof(syscall_table) / sizeof(unsigned int))
28:
29: int do_syscall(unsigned int, int, int, int, int, int, int, struct sigcontext);
30:
31: int sys_exit(int);
32: void do_exit(int);
33: int sys_fork(int, int, int, int, int, struct sigcontext *);
34: int sys_read(unsigned int, char *, int);
35: int sys_write(unsigned int, const char *, int);
36: int sys_open(const char *, int, __mode_t);
37: int sys_close(unsigned int);
38: int sys_waitpid(__pid_t, int *, int);
39: int sys_creat(const char *, __mode_t);
40: int sys_link(const char *, const char *);
41: int sys_unlink(const char *);
42: int sys_execve(const char *, char **, char **, int, int, struct sigcontext *);
43: int sys_chdir(const char *);
44: int sys_time(__time_t *);
45: int sys_mknod(const char *, __mode_t, __dev_t);
46: int sys_chmod(const char *, __mode_t);
47: int sys_chown(const char *, __uid_t, __gid_t);
48: int sys_stat(const char *, struct old_stat *);
49: int sys_lseek(unsigned int, __off_t, unsigned int);
50: int sys_getpid(void);
51: int sys_mount(const char *, const char *, const char *, unsigned long int, const
void *);
52: int sys_umount(const char *);
53: int sys_setuid(__uid_t);
54: int sys_getuid(void);
55: int sys_stime(__time_t *);
56: int sys_alarm(unsigned int);
57: int sys_fstat(unsigned int, struct old_stat *);
58: int sys_pause(void);
59: int sys_utime(const char *, struct utimbuf *);
60: int sys_access(const char *, __mode_t);
61: int sys_ftime(struct timeb *);
62: void sys_sync(void);
63: int sys_kill(__pid_t, __sigset_t);
64: int sys_rename(const char *, const char *);
65: int sys_mkdir(const char *, __mode_t);
66: int sys_rmdir(const char *);

```

include/fiwix/syscalls.h

Page 2/3

```

67: int sys_dup(unsigned int);
68: int sys_pipe(int *);
69: int sys_times(struct tms *);
70: int sys_brk(unsigned int);
71: int sys_setgid(__gid_t);
72: int sys_getgid(void);
73: unsigned int sys_signal(__sigset_t, void(*sighandler)(int));
74: int sys_geteuid(void);
75: int sys_getegid(void);
76: int sys_umount2(const char *, int);
77: int sys_ioctl(unsigned int, int, unsigned long int);
78: int sys_fcntl(int, int, unsigned long int);
79: int sys_setpgid(__pid_t, __pid_t);
80: int sys_olduname(struct oldold_utsname *);
81: int sys_umask(__mode_t);
82: int sys_chroot(const char *);
83: int sys_ustat(__dev_t, struct ustat *);
84: int sys_dup2(int, int);
85: int sys_getppid(void);
86: int sys_getpgrp(void);
87: int sys_setsid(void);
88: int sys_sigaction(__sigset_t, const struct sigaction *, struct sigaction *);
89: int sys_sgetmask(void);
90: int sys_ssetmask(int);
91: int sys_setreuid(__uid_t, __uid_t);
92: int sys_setregid(__gid_t, __gid_t);
93: int sys_sigsuspend(__sigset_t *);
94: int sys_sigpending(__sigset_t *);
95: int sys_sethostname(const char *, int);
96: int sys_setrlimit(int, const struct rlimit *);
97: int sys_getrlimit(int, struct rlimit *);
98: int sys_getrusage(int, struct rusage *);
99: int sys_gettimeofday(struct timeval *, struct timezone *);
100: int sys_settimeofday(const struct timeval *, const struct timezone *);
101: int sys_getgroups(__ssize_t, __gid_t *);
102: int sys_setgroups(__ssize_t, const __gid_t *);
103: int old_select(unsigned long int *);
104: int sys_symlink(const char *, const char *);
105: int sys_lstat(const char *, struct old_stat *);
106: int sys_readlink(const char *, char *, __size_t);
107: int sys_reboot(int, int, int);
108: int old_mmap(struct mmap *);
109: int sys_munmap(unsigned int, __size_t);
110: int sys_truncate(const char *, __off_t);
111: int sys_ftruncate(int, __off_t);
112: int sys_fchmod(int, __mode_t);
113: int sys_fchown(int, __uid_t, __gid_t);
114: int sys_statfs(const char *, struct statfs *);
115: int sys_fstatfs(unsigned int, struct statfs *);
116: int sys_ioperm(unsigned long int, unsigned long int, int);
117: int sys_socketcall(int, unsigned long int *);
118: int sys_setitimer(int, const struct itimerval *, struct itimerval *);
119: int sys_getitimer(int, struct itimerval *);
120: int sys_newstat(const char *, struct new_stat *);
121: int sys_newlstat(const char *, struct new_stat *);
122: int sys_newfstat(unsigned int, struct new_stat *);
123: int sys_uname(struct old_utsname *);
124: int sys_iopl(int, int, int, int, int, struct sigcontext *);
125: int sys_wait4(__pid_t, int *, int, struct rusage *);
126: int sys_sysinfo(struct sysinfo *);
127: int sys_fsync(int);
128: int sys_sigreturn(unsigned int, int, int, int, int, int, struct sigcontext *);
129: int sys_setdomainname(const char *, int);
130: int sys_newuname(struct new_utsname *);
131: int sys_mprotect(unsigned int, __size_t, int);
132: int sys_sigprocmask(int, const __sigset_t *, __sigset_t *);
133: int sys_getpgid(__pid_t);

```

include/fiwix/syscalls.h

Page 3/3

```
134: int sys_fchdir(unsigned int);
135: int sys_personality(unsigned long int);
136: int sys_setfsuid(__uid_t);
137: int sys_setfsgid(__gid_t);
138: int sys_llseek(unsigned int, unsigned long int, unsigned long int, __loff_t *, u
nsigned int);
139: int sys_getdents(unsigned int, struct dirent *, unsigned int);
140: int sys_select(int, fd_set *, fd_set *, fd_set *, struct timeval *);
141: int sys_flock(int, int);
142: int sys_getsid(__pid_t);
143: int sys_fdatasync(int);
144: int sys_nanosleep(const struct timespec *, struct timespec *);
145: int sys_getcwd(char *, __size_t);
146:
147: #endif /* _FIWIX_SYSCALLS_H */
```


include/fiwix/sysrq.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/sysrq.h
3:  *
4:  * Copyright 2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_SYSRQ_H
9: #define _FIWIX_SYSRQ_H
10:
11: /* the key combination consists of Alt+SysRq and another key, defined below */
12: #define SYSRQ_STACK      0x00000002    /* 'l' -> stack backtrace */
13: #define SYSRQ_TASKS     0x00000004    /* 't' -> task list */
14: #define SYSRQ_UNDEF     0x80000000    /* Undefined operation */
15:
16: void do_sysrq(int);
17:
18: #endif /* _FIWIX_SYSRQ_H */
```

include/fiwix/system.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/system.h
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_SYSTEM_H
9: #define _FIWIX_SYSTEM_H
10:
11: #define UTS_SYSNAME      "Fiwix"
12: #define UTS_NODENAME    "(none)"
13: #define UTS_RELEASE     "1.3.0"
14: #define UTS_DOMAINNAME  "(none)"
15:
16: struct sysinfo {
17:     long int uptime;           /* seconds since boot */
18:     unsigned long int loads[3]; /* load average (1, 5 and 15 minutes) */
19:     unsigned long int totalram; /* total usable main memory size */
20:     unsigned long int freeram;  /* available memory size */
21:     unsigned long int sharedram; /* amount of shared memory */
22:     unsigned long int bufferram; /* amount of memory used by buffers */
23:     unsigned long int totalswap; /* total swap space size */
24:     unsigned long int freeswap; /* available swap space */
25:     unsigned short int procs;   /* number of current processes */
26:     char _f[22];              /* pads structure to 64 bytes */
27: };
28:
29: #endif /* _FIWIX_SYSTEM_H */
```

include/fiwix/termbits.h

Page 1/3

```

1: /*
2:  * fiwix/include/fiwix/termbits.h
3:  *
4:  */
5:
6: #ifndef _FIWIX_TERMBITS_H
7: #define _FIWIX_TERMBITS_H
8:
9: /* These definitions match those used by the 4.4 BSD kernel.
10:    If the operating system has termios system calls or ioctls that
11:    correctly implement the POSIX.1 behavior, there should be a
12:    system-dependent version of this file that defines 'struct termios',
13:    'tcflag_t', 'cc_t', 'speed_t' and the 'TC*' constants appropriately. */
14:
15: /* Type of terminal control flag masks. */
16: typedef unsigned long int tcflag_t;
17:
18: /* Type of control characters. */
19: typedef unsigned char cc_t;
20:
21: /* Type of baud rate specifiers. */
22: typedef long int speed_t;
23:
24: /* c_iflag bits */
25: #define IGNBRK 0000001 /* Ignore break condition */
26: #define BRKINT 0000002 /* Signal interrupt on break */
27: #define IGNPAR 0000004 /* Ignore characters with parity errors */
28: #define PARMRK 0000010 /* Mark parity and framing errors */
29: #define INPCK 0000020 /* Enable input parity check */
30: #define ISTRIP 0000040 /* Strip 8th bit off characters */
31: #define INLCR 0000100 /* Map NL to CR on input */
32: #define IGNCR 0000200 /* Ignore CR */
33: #define ICRNL 0000400 /* Map CR to NL on input */
34: #define IUCLC 0001000 /* Convert to lowercase */
35: #define IXON 0002000 /* Enable start/stop output control */
36: #define IXANY 0004000 /* Any character will restart after stop */
37: #define IXOFF 0010000 /* Enable start/stop input control */
38: #define IMAXBEL 0020000 /* Ring bell when input queue is full */
39:
40: /* c_oflag bits */
41: #define OPOST 0000001 /* Perform output processing */
42: #define OLCUC 0000002
43: #define ONLCR 0000004 /* Map NL to CR-NL on output */
44: #define OCRNL 0000010
45: #define ONOCR 0000020
46: #define ONLRET 0000040
47: #define OFILL 0000100
48: #define OFDEL 0000200
49: #define NLDLY 0000400
50: #define NL0 0000000
51: #define NL1 0000400
52: #define CRDLY 0003000
53: #define CR0 0000000
54: #define CR1 0001000
55: #define CR2 0002000
56: #define CR3 0003000
57: #define TABDLY 0014000
58: #define TAB0 0000000
59: #define TAB1 0004000
60: #define TAB2 0010000
61: #define TAB3 0014000
62: #define XTABS 0014000
63: #define BSDLY 0020000
64: #define BS0 0000000
65: #define BS1 0020000
66: #define VTDLY 0040000
67: #define VT0 0000000

```

include/fiwix/termbits.h

Page 2/3

```

68: #define VT1 0040000
69: #define FFDLY 0100000
70: #define FF0 0000000
71: #define FF1 0100000
72:
73: /* c_cflag bit meaning */
74: #define CBAUD 0010017
75: #define B0 0000000 /* hang up */
76: #define B50 0000001 /* 50 baud */
77: #define B75 0000002 /* 75 baud */
78: #define B110 0000003 /* 110 baud */
79: #define B134 0000004 /* 134 baud */
80: #define B150 0000005 /* 150 baud */
81: #define B200 0000006 /* 200 baud */
82: #define B300 0000007 /* 300 baud */
83: #define B600 0000010 /* 600 baud */
84: #define B1200 0000011 /* 1200 baud */
85: #define B1800 0000012 /* 1800 baud */
86: #define B2400 0000013 /* 2400 baud */
87: #define B4800 0000014 /* 4800 baud */
88: #define B9600 0000015 /* 9600 baud */
89: #define B19200 0000016 /* 19200 baud */
90: #define B38400 0000017 /* 38400 baud */
91: #define EXTA B19200
92: #define EXTB B38400
93: #define CSIZE 0000060 /* Number of bits per byte (mask) */
94: #define CS5 0000000 /* 5 bits per byte */
95: #define CS6 0000020 /* 6 bits per byte */
96: #define CS7 0000040 /* 7 bits per byte */
97: #define CS8 0000060 /* 8 bits per byte */
98: #define CSTOPB 0000100 /* Two stop bits instead of one */
99: #define CREAD 0000200 /* Enable receiver */
100: #define PARENB 0000400 /* Parity enable */
101: #define PARODD 0001000 /* Odd parity instead of even */
102: #define HUPCL 0002000 /* Hang up on last close */
103: #define CLOCAL 0004000 /* Ignore modem status lines */
104: #define CBAUDEX 0010000
105: #define B57600 0010001
106: #define B115200 0010002
107: #define B230400 0010003
108: #define B460800 0010004
109: #define B500000 0010005
110: #define B576000 0010006
111: #define B921600 0010007
112: #define B1000000 0010010
113: #define B1152000 0010011
114: #define B1500000 0010012
115: #define B2000000 0010013
116: #define B2500000 0010014
117: #define B3000000 0010015
118: #define B3500000 0010016
119: #define B4000000 0010017
120: #define CIBAUD 002003600000 /* input baud rate (not used) */
121: #define CMSPAR 010000000000 /* mark or space (stick) parity */
122: #define CRTSCTS 020000000000 /* flow control */
123:
124: /* c_lflag bits */
125: #define ISIG 0000001 /* Enable signals */
126: #define ICANON 0000002 /* Do erase and kill processing */
127: #define XCASE 0000004
128: #define ECHO 0000010 /* Enable echo */
129: #define ECHOE 0000020 /* Visual erase for ERASE */
130: #define ECHOK 0000040 /* Echo NL after KILL */
131: #define ECHONL 0000100 /* Echo NL even if echo is OFF */
132: #define NOFLSH 0000200 /* Disable flush after interrupt */
133: #define TOSTOP 0000400 /* Send SIGTTOU for background output */
134: #define ECHOCTL 0001000 /* Echo control characters as ^X */

```

include/fiwix/termbits.h

Page 3/3

```

135: #define ECHOPRT 0002000 /* Hardcopy visual erase */
136: #define ECHOKE 0004000 /* Visual erase for KILL */
137: #define FLUSHO 0010000 /* Output being flushed (state) */
138: #define PENDIN 0040000 /* Retype pending input (state) */
139: #define IEXTEN 0100000 /* Enable DISCARD and LNEXT */
140:
141: /* c_cc characters */
142: #define VINTR 0 /* Interrupt character [ISIG] */
143: #define VQUIT 1 /* Quit character [ISIG] */
144: #define VERASE 2 /* Erase character [ICANON] */
145: #define VKILL 3 /* Kill-line character [ICANON] */
146: #define VEOF 4 /* End-of-file character [ICANON] */
147: #define VTIME 5 /* Time-out value (1/10 secs) [!ICANON] */
148: #define VMIN 6 /* Minimum # of bytes read at once [!ICANON] */
149: #define VSWTC 7
150: #define VSTART 8 /* Start (X-ON) character [IXON, IXOFF] */
151: #define VSTOP 9 /* Stop (X-OFF) character [IXON, IXOFF] */
152: #define VSUSP 10 /* Suspend character [ISIG] */
153: #define VEOL 11 /* End-of-line character [ICANON] */
154: #define VREPRINT 12 /* Reprint-line character [ICANON] */
155: #define VDISCARD 13 /* Discard character [IEXTEN] */
156: #define VWERASE 14 /* Word-erase character [ICANON] */
157: #define VLNEXT 15 /* Literal-next character [IEXTEN] */
158: #define VEOL2 16 /* Second EOL character [ICANON] */
159:
160: /* Values for the ACTION argument to 'tcflow'. */
161: #define TCOOFF 0 /* Suspend output */
162: #define TCOON 1 /* Restart suspended output */
163: #define TCIOFF 2 /* Send a STOP character */
164: #define TCION 3 /* Send a START character */
165:
166: /* Values for the QUEUE_SELECTOR argument to 'tcflush'. */
167: #define TCIFLUSH 0 /* Discard data received but not yet read */
168: #define TCOFLUSH 1 /* Discard data written but not yet sent */
169: #define TCIOFLUSH 2 /* Discard all pending data */
170:
171: /* Values for the OPTIONAL_ACTIONS argument to 'tcsetattr'. */
172: #define TCSANOW 0 /* Change immediately */
173: #define TCSADRAIN 1 /* Change when pending output is written */
174: #define TCSAFLUSH 2 /* Flush pending input before changing */
175:
176: #endif /* _FIWIX_TERMBITS_H */

```

include/fiwix/termios.h

Page 1/1

```

1:  /*
2:  *  fiwix/include/fiwix/termios.h
3:  *
4:  *  Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  *  Distributed under the terms of the Fiwix License.
6:  */
7:
8:  #ifndef _FIWIX_TERMIOS_H
9:  #define _FIWIX_TERMIOS_H
10:
11:  #include <fiwix/termbits.h>
12:
13:  struct winsize {
14:      unsigned short int ws_row;
15:      unsigned short int ws_col;
16:      unsigned short int ws_xpixel;
17:      unsigned short int ws_ypixel;
18:  };
19:
20:
21:  #define NCC      8
22:
23:  /* old terminal control structure */
24:  struct termio {
25:      unsigned short int c_iflag;      /* input mode flags */
26:      unsigned short int c_oflag;      /* output mode flags */
27:      unsigned short int c_cflag;      /* control mode flags */
28:      unsigned short int c_lflag;      /* local mode flags */
29:      unsigned char c_line;            /* line discipline */
30:      unsigned char c_cc[NCC];        /* control characters */
31:  };
32:
33:
34:  #define NCCS 19
35:
36:  /* new terminal control structure */
37:  struct termios {
38:      tcflag_t c_iflag;      /* input mode flags */
39:      tcflag_t c_oflag;      /* output mode flags */
40:      tcflag_t c_cflag;      /* control mode flags */
41:      tcflag_t c_lflag;      /* local mode flags */
42:      cc_t c_line;           /* line discipline */
43:      cc_t c_cc[NCCS];      /* control characters */
44:  };
45:
46:  #endif /* _FIWIX_TERMIOS_H */

```

include/fiwix/timeb.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/timeb.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_TIMEB_H
9: #define _FIWIX_TIMEB_H
10:
11: struct timeb {
12:     unsigned int time;           /* in seconds since Epoch */
13:     unsigned short int millitm; /* additional milliseconds */
14:     short int timezone;         /* minutes west of GMT */
15:     short int dstflag;         /* nonzero if DST is used */
16: };
17:
18: #endif /* _FIWIX_TIMEB_H */
```

include/fiwix/time.h

Page 1/1

```

1:  /*
2:  *  fiwix/include/fiwix/time.h
3:  *
4:  *  Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  *  Distributed under the terms of the Fiwix License.
6:  */
7:
8:  #ifndef _FIWIX_TIME_H
9:  #define _FIWIX_TIME_H
10:
11:  #define ITIMER_REAL    0
12:  #define ITIMER_VIRTUAL 1
13:  #define ITIMER_PROF    2
14:
15:  struct timespec {
16:      long int tv_sec;          /* seconds since 00:00:00, 1 Jan 1970 UTC */
17:      long int tv_nsec;        /* nanoseconds (1000000000ns = 1sec) */
18:  };
19:
20:  struct timeval {
21:      long int tv_sec;          /* seconds since 00:00:00, 1 Jan 1970 UTC */
22:      long int tv_usec;        /* microseconds (1000000us = 1sec) */
23:  };
24:
25:  struct timezone {
26:      int tz_minuteswest;      /* minutes west of GMT */
27:      int tz_dsttime;          /* type of DST correction */
28:  };
29:
30:  struct itimerval {
31:      struct timeval it_interval;
32:      struct timeval it_value;
33:  };
34:
35:  struct mt {
36:      int mt_sec;
37:      int mt_min;
38:      int mt_hour;
39:      int mt_day;
40:      int mt_month;
41:      int mt_year;
42:  };
43:
44:  unsigned long int tv2ticks(const struct timeval *);
45:  void ticks2tv(long int, struct timeval *);
46:  int setitimer(int, const struct itimerval *, struct itimerval *);
47:  unsigned long int mktime(struct mt *);
48:
49:  #endif /* _FIWIX_TIME_H */

```


include/fiwix/timer.h

Page 1/1

```

1: /*
2:  * fiwix/include/fiwix/timer.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_TIMER_H
9: #define _FIWIX_TIMER_H
10:
11: #include <fiwix/types.h>
12: #include <fiwix/sigcontext.h>
13:
14: #define TIMER_IRQ      0
15: #define HZ              100      /* kernel's Hertz rate (100 = 10ms) */
16: #define TICK           (1000000 / HZ)
17:
18: #define UNIX_EPOCH     1970
19:
20: #define LEAP_YEAR(y)   ((y % 4) == 0 && ((y % 100) != 0 || (y % 400) == 0))
21: #define DAYS_PER_YEAR(y) ((LEAP_YEAR(y)) ? 366 : 365)
22:
23: #define SECS_PER_MIN   60
24: #define SECS_PER_HOUR  (SECS_PER_MIN * 60)
25: #define SECS_PER_DAY   (SECS_PER_HOUR * 24)
26:
27: #define INFINITE_WAIT  0xFFFFFFFF
28:
29: struct callout {
30:     int expires;
31:     void (*fn)(unsigned int);
32:     unsigned int arg;
33:     struct callout *next;
34: };
35:
36: struct callout_req {
37:     void (*fn)(unsigned int);
38:     unsigned int arg;
39: };
40:
41: void add_callout(struct callout_req *, unsigned int);
42: void del_callout(struct callout_req *);
43: void irq_timer(int, struct sigcontext *);
44: void irq_timer_bh(void);
45: void do_callouts_bh(void);
46: void get_system_time(void);
47: void set_system_time(__time_t);
48: void timer_init(void);
49:
50: #endif /* _FIWIX_TIMER_H */

```

include/fiwix/times.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/times.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_TIMES_H
9: #define _FIWIX_TIMES_H
10:
11: struct tms {
12:     __clock_t tms_utime;    /* CPU time spent in user-mode */
13:     __clock_t tms_stime;    /* CPU time spent in kernel-mode */
14:     __clock_t tms_cutime;   /* (tms_utime + tms_cutime) of children */
15:     __clock_t tms_cstime;   /* (tms_stime + tms_cstime) of children */
16: };
17:
18: #endif /* _FIWIX_TIMES_H */
```

include/fiwix/traps.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/traps.h
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_TRAPS_H
9: #define _FIWIX_TRAPS_H
10:
11: #include <fiwix/sigcontext.h>
12:
13: #define NR_EXCEPTIONS    32
14:
15: struct traps {
16:     char *name;
17:     void (*handler)(unsigned int, struct sigcontext *);
18:     char errcode;
19: };
20:
21: void do_divide_error(unsigned int, struct sigcontext *);
22: void do_debug(unsigned int, struct sigcontext *);
23: void do_nmi_interrupt(unsigned int, struct sigcontext *);
24: void do_breakpoint(unsigned int, struct sigcontext *);
25: void do_overflow(unsigned int, struct sigcontext *);
26: void do_bound(unsigned int, struct sigcontext *);
27: void do_invalid_opcode(unsigned int, struct sigcontext *);
28: void do_no_math_coprocessor(unsigned int, struct sigcontext *);
29: void do_double_fault(unsigned int, struct sigcontext *);
30: void do_coprocessor_segment_overrun(unsigned int, struct sigcontext *);
31: void do_invalid_tss(unsigned int, struct sigcontext *);
32: void do_segment_not_present(unsigned int, struct sigcontext *);
33: void do_stack_segment_fault(unsigned int, struct sigcontext *);
34: void do_general_protection(unsigned int, struct sigcontext *);
35: void do_page_fault(unsigned int, struct sigcontext *);
36: void do_reserved(unsigned int, struct sigcontext *);
37: void do_floating_point_error(unsigned int, struct sigcontext *);
38: void do_alignment_check(unsigned int, struct sigcontext *);
39: void do_machine_check(unsigned int, struct sigcontext *);
40: void do_simd_fault(unsigned int, struct sigcontext *);
41:
42: void trap_handler(unsigned int, struct sigcontext);
43:
44: const char * elf_lookup_symbol(unsigned int addr);
45: void stack_backtrace(void);
46: int dump_registers(unsigned int, struct sigcontext *);
47:
48: #endif /* _FIWIX_TRAPS_H */
```

include/fiwix/tty.h

Page 1/2

```

1: /*
2:  * fiwix/include/fiwix/tty.h
3:  *
4:  * Copyright 2018-2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_TTY_H
9: #define _FIWIX_TTY_H
10:
11: #include <fiwix/termios.h>
12: #include <fiwix/fs.h>
13: #include <fiwix/console.h>
14: #include <fiwix/serial.h>
15:
16: #define NR_TTYS          NR_VCONSOLES + NR_SERIAL
17:
18: #define CBSIZE          32      /* number of characters in cblock */
19: #define NR_CB_QUEUE    8      /* number of cblocks per queue */
20: #define CB_POOL_SIZE   128    /* number of cblocks in the central pool */
21:
22: #define TAB_SIZE       8
23: #define MAX_TAB_COLS   132    /* maximum number of tab stops */
24:
25: #define LAST_CHAR(q)   ((q)->tail ? (q)->tail->data[(q)->tail->end_off - 1] : N
ULL)
26:
27: struct clist {
28:     unsigned short int count;
29:     unsigned short int cb_num;
30:     struct cblock *head;
31:     struct cblock *tail;
32: };
33:
34: struct cblock {
35:     unsigned short int start_off;
36:     unsigned short int end_off;
37:     unsigned char data[CBSIZE];
38:     struct cblock *prev;
39:     struct cblock *next;
40: };
41:
42: struct kbd_state {
43:     char mode;
44: };
45:
46: struct tty {
47:     __dev_t dev;
48:     struct clist read_q;
49:     struct clist cooked_q;
50:     struct clist write_q;
51:     short int count;
52:     struct termios termios;
53:     struct winsize winsize;
54:     struct kbd_state kbd;
55:     __pid_t pid, pgid, sid;
56:     unsigned char lnext;
57:     void *driver_data;
58:     int canon_data;
59:     char tab_stop[132];
60:     int column;
61:
62:     /* formerly tty driver operations */
63:     void (*stop)(struct tty *);
64:     void (*start)(struct tty *);
65:     void (*deltab)(struct tty *);
66:     void (*reset)(struct tty *);

```

include/fiwix/tty.h

Page 2/2

```
67:         void (*input)(struct tty *);
68:         void (*output)(struct tty *);
69:         int (*open)(struct tty *);
70:         int (*close)(struct tty *);
71:         void (*set_termios)(struct tty *);
72:     };
73: extern struct tty tty_table[];
74:
75: int register_tty(__dev_t);
76: struct tty * get_tty(__dev_t);
77: void disassociate_ctty(struct tty *);
78: void termios_reset(struct tty *);
79: void do_cook(struct tty *);
80: int tty_putchar(struct tty *, unsigned char);
81: int tty_open(struct inode *, struct fd *);
82: int tty_close(struct inode *, struct fd *);
83: int tty_read(struct inode *, struct fd *, char *, __size_t);
84: int tty_write(struct inode *, struct fd *, const char *, __size_t);
85: int tty_ioctl(struct inode *, int cmd, unsigned long int);
86: int tty_lseek(struct inode *, __off_t);
87: int tty_select(struct inode *, int);
88: void tty_init(void);
89:
90: int tty_queue_putchar(struct tty *, struct clist *, unsigned char);
91: int tty_queue_unputchar(struct clist *);
92: unsigned char tty_queue_getchar(struct clist *);
93: void tty_queue_flush(struct clist *);
94: int tty_queue_room(struct clist *q);
95: void tty_queue_init(void);
96:
97: int vt_ioctl(struct tty *, int, unsigned long int);
98:
99: #endif /* _FIWIX_TTY_H */
```

include/fiwix/types.h

Page 1/1

```

1: /*
2:  * fiwix/include/fiwix/types.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_TYPES_H
9: #define _FIWIX_TYPES_H
10:
11: typedef __signed__ char __s8;
12: typedef unsigned char __u8;
13: typedef __signed__ short int __s16;
14: typedef unsigned short int __u16;
15: typedef __signed__ int __s32;
16: typedef unsigned int __u32;
17:
18: typedef __u16 __uid_t;
19: typedef __u16 __gid_t;
20: typedef __u32 __ino_t;
21: typedef __u16 __mode_t;
22: typedef __u16 __nlink_t;
23: typedef __u32 __off_t;
24: typedef __s32 __pid_t;
25: typedef __s32 __ssize_t;
26: typedef __u32 __size_t;
27: typedef unsigned long int __clock_t;
28: typedef __u32 __time_t;
29: typedef __u16 __dev_t;
30: typedef __u16 __key_t;
31: typedef __s32 __blk_t;          /* must be signed in order to return -EIO */
32: typedef __s32 __daddr_t;
33: typedef unsigned long long int __loff_t;
34:
35: /* number of descriptors that can fit in an 'fd_set' */
36: /* WARNING: this value must be the same as in the C Library */
37: #define __FD_SETSIZE      64
38:
39: #define __NFDBITS          (sizeof(unsigned long int) * 8)
40: #define __FDELT(d)        ((d) / __NFDBITS)
41: #define __FDMASK(d)       (1 << ((d) % __NFDBITS))
42:
43: /* define the fd_set structure for select() */
44: typedef struct {
45:     unsigned long int fds_bits[__FD_SETSIZE / __NFDBITS];
46: } fd_set;
47:
48: #define __FD_ZERO(set)      (memset_b((void *) (set), 0, sizeof(fd_set)))
49: #define __FD_SET(d, set)    ((set)->fds_bits[__FDELT(d)] |= __FDMASK(d))
50: #define __FD_CLR(d, set)   ((set)->fds_bits[__FDELT(d)] &= ~__FDMASK(d))
51: #define __FD_ISSET(d, set) ((set)->fds_bits[__FDELT(d)] & __FDMASK(d))
52:
53: #endif /* _FIWIX_TYPES_H */

```

include/fiwix/unistd.h

Page 1/4

```

1: /*
2:  * fiwix/include/fiwix/unistd.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_UNISTD_H
9: #define _FIWIX_UNISTD_H
10:
11: /* Linux 2.0.40 ABI (one syscall from Linux 2.2.26) */
12: /* #define SYS_setup */
13: #define SYS_exit 1
14: #define SYS_fork 2
15: #define SYS_read 3
16: #define SYS_write 4
17: #define SYS_open 5
18: #define SYS_close 6
19: #define SYS_waitpid 7
20: #define SYS_creat 8
21: #define SYS_link 9
22: #define SYS_unlink 10
23: #define SYS_execve 11
24: #define SYS_chdir 12
25: #define SYS_time 13
26: #define SYS_mknod 14
27: #define SYS_chmod 15
28: #define SYS_chown 16
29: #define SYS_break 17 /* -ENOSYS */
30: #define SYS_oldstat 18
31: #define SYS_lseek 19
32: #define SYS_getpid 20
33: #define SYS_mount 21
34: #define SYS_umount 22
35: #define SYS_setuid 23
36: #define SYS_getuid 24
37: #define SYS_stime 25
38: /* #define SYS_ptrace */
39: #define SYS_alarm 27
40: #define SYS_oldfstat 28
41: #define SYS_pause 29
42: #define SYS_utime 30
43: #define SYS_stty 31 /* -ENOSYS */
44: #define SYS_gtty 32 /* -ENOSYS */
45: #define SYS_access 33
46: /* #define SYS_nice */
47: #define SYS_ftime 35
48: #define SYS_sync 36
49: #define SYS_kill 37
50: #define SYS_rename 38
51: #define SYS_mkdir 39
52: #define SYS_rmdir 40
53: #define SYS_dup 41
54: #define SYS_pipe 42
55: #define SYS_times 43
56: #define SYS_prof 44 /* -ENOSYS */
57: #define SYS_brk 45
58: #define SYS_setgid 46
59: #define SYS_getgid 47
60: #define SYS_signal 48
61: #define SYS_geteuid 49
62: #define SYS_getegid 50
63: /* #define SYS_acct */
64: #define SYS_umount2 52 /* from 2.2.26, it was sys_phys() */
65: #define SYS_lock 53 /* -ENOSYS */
66: #define SYS_ioctl 54
67: #define SYS_fcntl 55

```

include/fiwix/unistd.h

Page 2/4

```

68: #define SYS_mpx                56                /* -ENOSYS */
69: #define SYS_setpgid            57
70: #define SYS_ulimit             58                /* -ENOSYS */
71: #define SYS_olduname           59
72: #define SYS_umask              60
73: #define SYS_chroot             61
74: #define SYS_ustat              62
75: #define SYS_dup2              63
76: #define SYS_getppid           64
77: #define SYS_getpgrp           65
78: #define SYS_setsid            66
79: #define SYS_sigaction         67
80: #define SYS_sgetmask          68
81: #define SYS_ssetmask          69
82: #define SYS_setreuid          70
83: #define SYS_setregid          71
84: #define SYS_sigsuspend        72
85: #define SYS_sigpending        73
86: #define SYS_sethostname       74
87: #define SYS_setrlimit         75
88: #define SYS_getrlimit         76
89: #define SYS_getrusage         77
90: #define SYS_gettimeofday      78
91: #define SYS_settimeofday      79
92: #define SYS_getgroups         80
93: #define SYS_setgroups         81
94: #define SYS_oldselect         82
95: #define SYS_symlink           83
96: #define SYS_oldlstat          84
97: #define SYS_readlink          85
98: /* #define SYS_uselib */
99: /* #define SYS_swapon */
100: #define SYS_reboot            88
101: /* #define SYS_oldreaddir */
102: #define SYS_old_mmap          90
103: #define SYS_munmap            91
104: #define SYS_truncate          92
105: #define SYS_ftruncate         93
106: #define SYS_fchmod            94
107: #define SYS_fchown            95
108: /* #define SYS_getpriority */
109: /* #define SYS_setpriority */
110: /* #define SYS_profil */
111: #define SYS_statfs            99
112: #define SYS_fstatfs          100
113: #define SYS_ioperm           101
114: #define SYS_socketcall        102
115: /* #define SYS_syslog */
116: #define SYS_setitimer         104
117: #define SYS_getitimer         105
118: #define SYS_newstat           106
119: #define SYS_newlstat          107
120: #define SYS_newfstat          108
121: #define SYS_uname             109
122: #define SYS_iopl              110
123: /* #define SYS_vhangup */
124: /* #define SYS_idle            112                -ENOSYS */
125: /* #define SYS_vm86old */
126: #define SYS_wait4             114
127: /* #define SYS_swapoff */
128: #define SYS_sysinfo           116
129: /* #define SYS_ipc */
130: #define SYS_fsync             118
131: #define SYS_sigreturn         119
132: /* #define SYS_clone */
133: #define SYS_setdomainname     121
134: #define SYS_newuname          122

```


include/fiwix/unistd.h

Page 3/4

```

135: /* #define SYS_modify_ldt */
136: /* #define SYS_adjtimex */
137: #define SYS_mprotect          125
138: #define SYS_sigprocmask      126
139: /* #define SYS_create_module */
140: /* #define SYS_init_module */
141: /* #define SYS_delete_module */
142: /* #define SYS_get_kernel_syms */
143: /* #define SYS_quotactl */
144: #define SYS_getpgid          132
145: #define SYS_fchdir           133
146: /* #define SYS_bdflush */
147: /* #define SYS_sysfs */
148: #define SYS_personality      136
149: /* #define afs_syscall */
150: #define SYS_setfsuid         138
151: #define SYS_setfsgid         139
152: #define SYS_llseek           140
153: #define SYS_getdents         141
154: #define SYS_select           142
155: #define SYS_flock            143
156: /* #define SYS_msync */
157: /* #define SYS_readv */
158: /* #define SYS_writev */
159: #define SYS_getsid           147
160: #define SYS_fdatasync        148
161: /* #define SYS_sysctl */
162: /* #define SYS_mlock */
163: /* #define SYS_munlock */
164: /* #define SYS_mlockall */
165: /* #define SYS_munlockall */
166: /* #define SYS_sched_setparam */
167: /* #define SYS_sched_getparam */
168: /* #define SYS_sched_setscheduler */
169: /* #define SYS_sched_getscheduler */
170: /* #define SYS_sched_yield */
171: /* #define SYS_sched_get_priority_max */
172: /* #define SYS_sched_get_priority_min */
173: /* #define SYS_sched_rr_get_interval */
174: #define SYS_nanosleep        162
175: /* #define SYS_mremap */
176:
177: /* extra system calls from Linux 2.2.26 */
178: /* #define SYS_mremap */
179: /* #define SYS_setresuid */
180: /* #define SYS_getresuid */
181: /* #define SYS_ni_syscall */
182: /* #define SYS_query_module */
183: /* #define SYS_poll */
184: /* #define SYS_nfsservctl */
185: /* #define SYS_setresgid */
186: /* #define SYS_getresgid */
187: /* #define SYS_prctl */
188: /* #define SYS_rt_sigreturn_wrapper */
189: /* #define SYS_rt_sigaction */
190: /* #define SYS_rt_sigprocmask */
191: /* #define SYS_rt_sigpending */
192: /* #define SYS_rt_sigtimedwait */
193: /* #define SYS_rt_sigqueueinfo */
194: /* #define SYS_rt_sigsuspend_wrapper */
195: /* #define SYS_pread */
196: /* #define SYS_pwrite */
197: /* #define SYS_chown */
198: #define SYS_getcwd           183
199: /* #define SYS_capget */
200: /* #define SYS_capset */
201: /* #define SYS_sigaltstack_wrapper */

```

include/fiwix/unistd.h

Page 4/4

```
202: /* #define SYS_sendfile */
203: /* #define SYS_ni_syscall */
204: /* #define SYS_ni_syscall */
205: #define SYS_vfork                190
206:
207: #endif /* _FIWIX_UNISTD_H */
```

include/fiwix/ustat.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/ustat.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_USTAT_H
9: #define _FIWIX_USTAT_H
10:
11: #include <fiwix/types.h>
12:
13: struct ustat {
14:     __daddr_t f_tfree;           /* total free blocks */
15:     __ino_t f_tinode;          /* number of free inodes */
16:     char f_fname;              /* filesystem name */
17:     char f_fpack;             /* filesystem pack name */
18: };
19:
20: #endif /* _FIWIX_USTAT_H */
```

include/fiwix/utime.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/utime.h
3:  *
4:  * Copyright 2018, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_UTIME_H
9: #define _FIWIX_UTIME_H
10:
11: #include <fiwix/types.h>
12:
13: struct utimbuf {
14:     __time_t actime;           /* access time */
15:     __time_t modtime;        /* modification time */
16: };
17:
18: #endif /* _FIWIX_UTIME_H */
```

include/fiwix/utsname.h

Page 1/2

```

1:  /* Copyright (C) 1991, 1992, 1994, 1996 Free Software Foundation, Inc.
2:    This file is part of the GNU C Library.
3:
4:    The GNU C Library is free software; you can redistribute it and/or
5:    modify it under the terms of the GNU Library General Public License as
6:    published by the Free Software Foundation; either version 2 of the
7:    License, or (at your option) any later version.
8:
9:    The GNU C Library is distributed in the hope that it will be useful,
10:   but WITHOUT ANY WARRANTY; without even the implied warranty of
11:   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
12:   Library General Public License for more details.
13:
14:   You should have received a copy of the GNU Library General Public
15:   License along with the GNU C Library; see the file COPYING.LIB.  If not,
16:   write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330,
17:   Boston, MA 02111-1307, USA.  */
18:
19:  /*
20:   *      POSIX Standard: 4.4 System Identification      <fiwix/utsname.h>
21:   */
22:
23:  #ifndef _FIWIX_UTSNAME_H
24:  #define _FIWIX_UTSNAME_H
25:
26:  #define _OLD_UTSNAME_LENGTH      9
27:  #define _UTSNAME_LENGTH         65
28:
29:  #ifndef _UTSNAME_NODENAME_LENGTH
30:  #define _UTSNAME_NODENAME_LENGTH _UTSNAME_LENGTH
31:  #endif
32:
33:  /* Very OLD structure describing the system and machine.  */
34:  struct oldold_utsname
35:  {
36:      char sysname[_OLD_UTSNAME_LENGTH];
37:      char nodename[_OLD_UTSNAME_LENGTH];
38:      char release[_OLD_UTSNAME_LENGTH];
39:      char version[_OLD_UTSNAME_LENGTH];
40:      char machine[_OLD_UTSNAME_LENGTH];
41:  };
42:
43:  /* OLD structure describing the system and machine.  */
44:  struct old_utsname
45:  {
46:      char sysname[_UTSNAME_LENGTH];
47:      char nodename[_UTSNAME_NODENAME_LENGTH];
48:      char release[_UTSNAME_LENGTH];
49:      char version[_UTSNAME_LENGTH];
50:      char machine[_UTSNAME_LENGTH];
51:  };
52:
53:  /* NEW structure describing the system and machine.  */
54:  struct new_utsname
55:  {
56:      /* Name of the implementation of the operating system.  */
57:      char sysname[_UTSNAME_LENGTH];
58:
59:      /* Name of this node on the network.  */
60:      char nodename[_UTSNAME_NODENAME_LENGTH];
61:
62:      /* Current release level of this implementation.  */
63:      char release[_UTSNAME_LENGTH];
64:      /* Current version level of this release.  */
65:      char version[_UTSNAME_LENGTH];
66:
67:      /* Name of the hardware type the system is running on.  */

```

include/fiwix/utsname.h

Page 2/2

```
68:     char machine[_UTSNAME_LENGTH];
69:     char domainname[_UTSNAME_LENGTH];
70: };
71:
72: extern struct new_utsname sys_utsname;
73: extern char UTS_MACHINE[_UTSNAME_LENGTH];
74:
75: #endif /* _FIWIX_UTSNAME_H */
```

include/fiwix/version.h

Page 1/1

```
1: #define UTS_VERSION "Wed Dec  8 07:40:17 GMT 2021"
```

include/fiwix/vgacon.h

Page 1/1

```

1: /*
2:  * fiwix/include/fiwix/vgacon.h
3:  *
4:  * Copyright 2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_VGACON_H
9: #define _FIWIX_VGACON_H
10:
11: #include <fiwix/console.h>
12:
13: #define MONO_ADDR          0xB0000L
14: #define COLOR_ADDR        0xB8000L
15:
16: #define MONO_6845_ADDR    0x3B4    /* i/o address (+1 for data register) */
17: #define COLOR_6845_ADDR  0x3D4    /* i/o address (+1 for data register) */
18:
19: #define ATTR_CONTROLLER   0x3C0    /* attribute controller register */
20: #define ATTR_CONTROLLER_PAS 0x20    /* palette address source */
21: #define INPUT_STAT1      0x3DA    /* input status #1 register */
22:
23: #define CRT_INDEX         0
24: #define CRT_DATA          1
25: #define CRT_CURSOR_STR    0xA
26: #define CRT_CURSOR_END    0xB
27: #define CRT_START_ADDR_HI 0xC
28: #define CRT_START_ADDR_LO 0xD
29: #define CRT_CURSOR_POS_HI 0xE
30: #define CRT_CURSOR_POS_LO 0xF
31:
32: #define CURSOR_MASK       0x1F
33: #define CURSOR_DISABLE   0x20
34:
35: void vgacon_put_char(struct vconsole *, unsigned char);
36: void vgacon_insert_char(struct vconsole *);
37: void vgacon_delete_char(struct vconsole *);
38: void vgacon_update_curpos(struct vconsole *);
39: void vgacon_show_cursor(struct vconsole *, int);
40: void vgacon_get_curpos(struct vconsole *);
41: void vgacon_write_screen(struct vconsole *, int, int, short int);
42: void vgacon_blank_screen(struct vconsole *);
43: void vgacon_scroll_screen(struct vconsole *, int, int);
44: void vgacon_restore_screen(struct vconsole *);
45: void vgacon_screen_on(struct vconsole *);
46: void vgacon_screen_off(unsigned int);
47: void vgacon_buf_scroll(struct vconsole *, int);
48: void vgacon_cursor_blink(unsigned int);
49: void vgacon_init(void);
50:
51: #endif /* _FIWIX_VGACON_H */

```


include/fiwix/video.h

Page 1/1

```
1: /*
2:  * fiwix/include/fiwix/video.h
3:  *
4:  * Copyright 2021, Jordi Sanfeliu. All rights reserved.
5:  * Distributed under the terms of the Fiwix License.
6:  */
7:
8: #ifndef _FIWIX_VIDEO_H
9: #define _FIWIX_VIDEO_H
10:
11: void video_init(void);
12:
13: #endif /* _FIWIX_VIDEO_H */
```

include/fiwix/vt.h

Page 1/1

```

1: #ifndef VT_H
2: #define VT_H
3:
4: /* prefix 0x56 is 'V', to avoid collision with termios and kd */
5:
6: #define VT_OPENQRY      0x5600 /* find available vt */
7:
8: struct vt_mode {
9:     char mode; /* vt mode */
10:    char waitv; /* if set, hang on writes if not active */
11:    short int relsig; /* signal to raise on release req */
12:    short int acqsig; /* signal to raise on acquisition */
13:    short int frsig; /* unused (set to 0) */
14: };
15: #define VT_GETMODE      0x5601 /* get mode of active vt */
16: #define VT_SETMODE      0x5602 /* set mode of active vt */
17: #define VT_AUTO         0x00 /* auto vt switching */
18: #define VT_PROCESS      0x01 /* process controls switching */
19: #define VT_ACKACQ      0x02 /* acknowledge switch */
20:
21: struct vt_stat {
22:    unsigned short int v_active; /* active vt */
23:    unsigned short int v_signal; /* signal to send */
24:    unsigned short int v_state; /* vt bitmask */
25: };
26: #define VT_GETSTATE     0x5603 /* get global vt state info */
27: #define VT_SENDSIG     0x5604 /* signal to send to bitmask of vts */
28:
29: #define VT_RELDISP     0x5605 /* release display */
30:
31: #define VT_ACTIVATE    0x5606 /* make vt active */
32: #define VT_WAITACTIVE  0x5607 /* wait for vt active */
33: #define VT_DISALLOCATE 0x5608 /* free memory associated to vt */
34:
35: struct vt_sizes {
36:    unsigned short int v_rows; /* number of rows */
37:    unsigned short int v_cols; /* number of columns */
38:    unsigned short int v_scrollsize; /* number of lines of scrollbar */
39: };
40: #define VT_RESIZE      0x5609 /* set kernel's idea of screensize */
41:
42: struct vt_consize {
43:    unsigned short int v_rows; /* number of rows */
44:    unsigned short int v_cols; /* number of columns */
45:    unsigned short int v_vlin; /* number of pixel rows on screen */
46:    unsigned short int v_clin; /* number of pixel rows per character */
47:    unsigned short int v_vcol; /* number of pixel columns on screen */
48:    unsigned short int v_ccol; /* number of pixel columns per character
*/
49: };
50: #define VT_RESIZEX     0x560A /* set kernel's idea of screensize + more */
51: #define VT_LOCKSWITCH  0x560B /* disallow vt switching */
52: #define VT_UNLOCKSWITCH 0x560C /* allow vt switching */
53:
54: #endif /* VT_H */

```

README.md

Page 1/2

```

1: Fiwix
2: =====
3: Fiwix is an operating system kernel written from scratch, based on the UNIX arch
itecture and fully focused on being POSIX compatible. It is designed and developed main
ly as a hobby OS and, since it serves also for educational purposes, the kernel code is
kept as simple as possible for the benefit of students and OS enthusiasts. It is small
in size (less than 50K lines of code), runs on the i386 hardware platform and is compa
tible with a good base of existing GNU applications.
4:
5: Features
6: -----
7: - Mostly written in C language (Assembly only used in the needed parts).
8: - GRUB Multiboot Specification v1 compliant.
9: - Full 32bit protected mode non-preemptive kernel.
10: - For i386 processors and higher.
11: - Preemptive multitasking.
12: - Protected task environment (independent memory address per process).
13: - POSIX-compliant (mostly).
14: - Process groups, sessions and job control.
15: - Interprocess communication with pipes and signals.
16: - BSD file locking mechanism (POSIX restricted to file and advisory only).
17: - Virtual memory management up to 4GB (1GB physical only and no swapping yet).
18: - Demand paging with Copy-On-Write feature.
19: - Linux 2.0 ABI system calls compatibility (mostly).
20: - ELF-386 executable format support (statically and dynamically linked).
21: - Round Robin based scheduler algorithm (no priorities yet).
22: - VFS abstraction layer.
23: - EXT2 filesystem support with 1KB, 2KB and 4KB block sizes.
24: - Minix v1 and v2 filesystem support.
25: - Linux-like PROC filesystem support (read only).
26: - PIPE pseudo-filesystem support.
27: - ISO9660 filesystem support with Rock Ridge extensions.
28: - RAMdisk device support.
29: - Initial RAMdisk (initrd) image support.
30: - SVGAlib based applications support.
31: - Keyboard driver with Linux keymaps support.
32: - Frame buffer device support for VESA VBE 2.0+ compliant graphic cards.
33: - Serial port (RS-232) driver support.
34: - Remote serial console support.
35: - Parallel port printer driver support.
36: - Basic implementation of a Pseudo-Random Number Generator.
37: - Floppy disk device driver and DMA management.
38: - IDE/ATA ATAPI CD-ROM device driver.
39: - IDE/ATA hard disk device driver.
40:
41: Compiling
42: -----
43: The command needed to build the Fiwix kernel is `make clean ; make`. This will
create the files **fiwix** (the kernel itself) and **System.map.gz** (the symbol table)
in the root directory of the source code tree.
44:
45: Before compiling you might want to tweak the kernel configuration by changing th
e default values in `include/fiwix/config.h`.
46:
47: Keep in mind that the kernel doesn't do anything on its own, you need to create
a user-space environment to make use of it. Upon booting, the kernel mounts the root fi
lesystem and tries to run `/sbin/init` on it, so you would need to provide this program
yourself. Fortunately, FiwixOS provides a full user-space UNIX-like environment to te
st the Fiwix kernel.
48:
49: Installing
50: -----
51: You can proceed to install [FiwixOS] (https://www.fiwix.org/downloads.html) on a
hard disk either by booting from the CD-ROM or from a floppy. If you chosen the latter,
you will also need the Installation CD-ROM inserted in order to install the packages t
hat form all the system environment.
52:

```

README.md

Page 2/2

```
53: Let the system boot and when you are ready, just type `install.sh`.
54:
55: The minimal hardware requirements are as follows:
56:
57: - Standard IBM PC-AT architecture.
58: - i386 processor or higher.
59: - 3MB of RAM memory (128MB recommended).
60: - IDE/ATAPI CD-ROM or floppy disk (3.5", 1.44MB).
61: - 1GB ATA hard disk.
62:
63: Please keep in mind that this is a kernel in its very early stages and may well
have serious bugs and broken features which have not yet been identified or resolved.
64:
65: Let me repeat that.
66:
67: Please keep in mind that this is a kernel in its very early stages and may well
have serious bugs and broken features which have not yet been identified or resolved.
68:
69: *****
70: *** USE AT YOUR OWN RISK! ***
71: *****
72:
73: References
74: -----
75: - [Website] (https://www.fiwix.org)
76: - [Mailing List] (https://lists.sourceforge.net/lists/listinfo/fiwix-general)
77: - [IRC] (https://web.libera.chat/)
78:
79: License
80: -----
81: Fiwix is free software licensed under the terms of the MIT License, see the LICE
NSE file for more details.
82: Copyright (C) 2018-2021, Jordi Sanfeliu.
83:
84: Credits
85: -----
86: Fiwix was created by [Jordi Sanfeliu] (https://www.fibranet.cat).
87: You can contact me at [jordi@fibranet.cat] (mailto:jordi@fibranet.cat).
88:
89:
```

LICENSE

Page 1/1

1: MIT License
2:
3: **Copyright** (c) 2018 Jordi Sanfeliu
4:
5: Permission is hereby granted, free of charge, to any person obtaining a copy
6: of this software and associated documentation **files** (the "**Software**"), to deal
7: in the Software without restriction, including without limitation the rights
8: to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
9: copies of the Software, and to permit persons to whom the Software is
10: furnished to **do** so, subject to the following conditions:
11:
12: The above copyright notice and this permission notice shall be included in all
13: copies or substantial portions of the Software.
14:
15: THE SOFTWARE IS PROVIDED "**AS IS**", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
16: IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
17: FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
18: AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
19: LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
20: OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
21: SOFTWARE.
22:

CODING

Page 1/1

```
1: Fiwix kernel coding standards
2: -----
3: It's easier on everyone if all authors working on a shared code base are
4: consistent in the way they write their programs. Fiwix has the following
5: conventions in its code:
6:
7: - Use of snake_case (multi-word names are lower_case_with_underscores) for
8:   everything except for macro and constant names.
9:
10: - No space after the name of a function in a call.
11:   For example, printk("hello") not printk ("hello").
12:
13: - Optional space after keywords "if", "for", "while", "switch".
14:   For example, if(x) or if (x).
15:
16: - Space before braces (except in function declarations).
17:   For example, if(x) { not if(x){.
18:
19: - Space between operands.
20:   For example, for(n = 0; n < 10; n++), not for(n=0;n<10;n++).
21:
22: - Beginning-of-line indentation via tabs, not spaces.
23:
24: - Preprocessor macros are always UPPERCASE.
25:
26: - Pointer types have spaces: (uint16_t *) not (uint16_t*).
27:
28: - Comments in code are always as in C89 /* ... */.
29:
30: - Multiline comments start always with a capital letter and the last sentence
31:   ends with a period.
32:
33: - Functions that take no arguments are declared as f(void) not f().
34:
35: - The open parenthesis is always on the same line as the function name.
36:
37: - There is never a space between the function name and the open parenthesis.
38:
39: - There is never a space between the parentheses and the parameters.
40:
41: - The open curly brace is always at the next line of the function declaration.
42:
43: - Conditionals should always include curly braces, even if there is only a
44:   single statement within the conditional.
45:
46:
47: Recommended for reading
48: -----
49: - http://skull.piratehaven.org/~bapper/298c/cstyle.html
50:
```

THANKS

Page 1/1

1: THANKS!
2:
3: A huge THANKS to all people who created their own hobby, and not so hobby,
4: operating systems and made them freely available on Internet.
5:
6: From those simple "**Hello world!\n**" kernels to a real self-hosting operating
7: system, their projects were a invaluable source of information and inspiration
8: to create the Fiwix kernel.
9:
10: A special thanks to OSDEV Community **for** their tutorials, documents, wikis, etc.
11:
12: Jordi Sanfeliu
13: **https://www.fiwix.org**
14:

Table of Contents

1	Makefile	1	pages	55	lines	21/12/06	21:01:05
2	fiwix.ld	1	pages	55	lines	21/02/12	19:47:07
3	kernel/boot.S	3	pages	152	lines	21/10/09	18:01:27
4	kernel/cmos.c	1	pages	57	lines	18/04/21	21:19:46
5	kernel/core386.S	10	pages	644	lines	21/12/06	21:03:28
6	kernel/cpu.c	5	pages	263	lines	19/10/23	23:20:36
7	kernel/gdt.c	1	pages	53	lines	21/09/30	08:58:57
8	kernel/idt.c	2	pages	60	lines	21/10/28	09:13:34
9	kernel/init.c	2	pages	122	lines	21/09/07	18:19:33
10	kernel/main.c	3	pages	159	lines	21/12/08	08:37:25
11	kernel/Makefile	1	pages	21	lines	21/05/24	16:22:05
12	kernel/multiboot.c	5	pages	310	lines	21/10/14	10:52:43
13	kernel/pic.c	4	pages	220	lines	21/12/04	21:37:08
14	kernel/pit.c	1	pages	29	lines	18/04/21	21:19:46
15	kernel/process.c	7	pages	410	lines	21/11/13	20:25:12
16	kernel/sched.c	2	pages	90	lines	21/12/06	21:09:05
17	kernel/signal.c	4	pages	219	lines	21/11/15	17:12:21
18	kernel/sleep.c	4	pages	231	lines	21/12/06	20:26:50
19	kernel/syscalls	0	pages	0	lines	21/12/08	07:48:05
20	kernel/syscalls.c	7	pages	407	lines	21/08/28	19:46:09
21	kernel/timer.c	7	pages	451	lines	21/12/06	21:09:18
22	kernel/traps.c	6	pages	348	lines	21/05/25	08:29:57
23	kernel/syscalls/access.c	1	pages	61	lines	18/04/21	21:19:46
24	kernel/syscalls/alarm.c	1	pages	39	lines	18/04/21	21:19:46
25	kernel/syscalls/brk.c	1	pages	54	lines	21/08/10	09:45:52
26	kernel/syscalls/chdir.c	1	pages	49	lines	18/04/21	21:19:46
27	kernel/syscalls/chmod.c	1	pages	56	lines	19/12/20	18:21:44
28	kernel/syscalls/chown.c	2	pages	68	lines	21/10/18	16:36:01
29	kernel/syscalls/chroot.c	1	pages	44	lines	18/04/21	21:19:46
30	kernel/syscalls/close.c	1	pages	39	lines	21/07/24	19:55:49
31	kernel/syscalls/creat.c	1	pages	23	lines	18/04/21	21:19:46
32	kernel/syscalls/dup2.c	1	pages	41	lines	18/04/21	21:19:46
33	kernel/syscalls/dup.c	1	pages	37	lines	18/04/21	21:19:46
34	kernel/syscalls/execve.c	6	pages	378	lines	21/08/25	14:01:35
35	kernel/syscalls/exit.c	2	pages	117	lines	21/11/25	11:32:54
36	kernel/syscalls/fchdir.c	1	pages	34	lines	18/04/21	21:19:46
37	kernel/syscalls/fchmod.c	1	pages	42	lines	18/04/21	21:19:46
38	kernel/syscalls/fchown.c	1	pages	53	lines	18/04/21	21:19:46
39	kernel/syscalls/fcntl.c	2	pages	64	lines	18/04/21	21:19:46
40	kernel/syscalls/fdatasync.c	1	pages	22	lines	18/04/21	21:19:46
41	kernel/syscalls/flock.c	1	pages	28	lines	18/04/21	21:19:46
42	kernel/syscalls/fork.c	3	pages	149	lines	21/09/27	17:06:45
43	kernel/syscalls/fstat.c	1	pages	44	lines	18/04/21	21:19:46
44	kernel/syscalls/fstatfs.c	1	pages	36	lines	18/04/21	21:19:46
45	kernel/syscalls/fsync.c	1	pages	39	lines	18/04/21	21:19:46
46	kernel/syscalls/ftime.c	1	pages	34	lines	18/04/21	21:19:46
47	kernel/syscalls/ftruncate.c	1	pages	53	lines	18/04/21	21:19:46
48	kernel/syscalls/getcwd.c	1	pages	29	lines	18/04/21	21:19:46
49	kernel/syscalls/getdents.c	1	pages	45	lines	18/04/21	21:19:46
50	kernel/syscalls/getegid.c	1	pages	20	lines	18/04/21	21:19:46
51	kernel/syscalls/geteuid.c	1	pages	20	lines	18/04/21	21:19:46
52	kernel/syscalls/getgid.c	1	pages	20	lines	18/04/21	21:19:46
53	kernel/syscalls/getgroups.c	1	pages	54	lines	20/01/11	19:01:23
54	kernel/syscalls/getitimer.c	1	pages	48	lines	18/04/21	21:19:46
55	kernel/syscalls/getpgid.c	1	pages	38	lines	21/12/02	17:09:32
56	kernel/syscalls/getpgrp.c	1	pages	20	lines	18/04/21	21:19:46
57	kernel/syscalls/getpid.c	1	pages	20	lines	18/04/21	21:19:46
58	kernel/syscalls/getppid.c	1	pages	20	lines	18/04/21	21:19:46
59	kernel/syscalls/getrlimit.c	1	pages	35	lines	18/04/21	21:19:46
60	kernel/syscalls/getrusage.c	1	pages	40	lines	18/04/21	21:19:46
61	kernel/syscalls/getsid.c	1	pages	39	lines	21/12/02	17:09:16
62	kernel/syscalls/gettimeofday.c	1	pages	41	lines	20/05/24	19:04:39
63	kernel/syscalls/getuid.c	1	pages	21	lines	21/01/05	19:19:30
64	kernel/syscalls/ioctl.c	1	pages	41	lines	18/04/21	21:19:46
65	kernel/syscalls/ioperm.c	1	pages	28	lines	20/06/07	10:35:30
66	kernel/syscalls/iopl.c	1	pages	58	lines	18/04/21	21:19:46
67	kernel/syscalls/kill.c	1	pages	47	lines	21/12/02	17:08:59

Table of Contents

Page -2/5

68	kernel/syscalls/link.c	2 pages	113 lines	18/04/21	21:19:46
69	kernel/syscalls/llseek.c	1 pages	57 lines	20/09/15	11:56:46
70	kernel/syscalls/lseek.c	1 pages	58 lines	18/04/21	21:19:46
71	kernel/syscalls/lstat.c	1 pages	51 lines	18/04/21	21:19:46
72	kernel/syscalls/Makefile	1 pages	20 lines	18/04/21	21:19:46
73	kernel/syscalls/mkdir.c	1 pages	66 lines	18/04/21	21:19:46
74	kernel/syscalls/mknod.c	2 pages	71 lines	18/04/21	21:19:46
75	kernel/syscalls/mount.c	4 pages	221 lines	18/04/21	21:19:46
76	kernel/syscalls/mprotect.c	1 pages	49 lines	21/08/22	18:44:35
77	kernel/syscalls/munmap.c	1 pages	22 lines	21/07/14	16:36:23
78	kernel/syscalls/nanosleep.c	1 pages	62 lines	21/01/05	19:13:06
79	kernel/syscalls/newfstat.c	1 pages	56 lines	21/10/20	11:15:01
80	kernel/syscalls/newlstat.c	1 pages	64 lines	21/10/20	11:28:43
81	kernel/syscalls/newstat.c	1 pages	64 lines	21/10/20	11:16:22
82	kernel/syscalls/newuname.c	1 pages	34 lines	21/05/15	08:13:36
83	kernel/syscalls/old_mmap.c	1 pages	52 lines	21/07/08	22:15:15
84	kernel/syscalls/old_select.c	1 pages	42 lines	18/04/21	21:19:46
85	kernel/syscalls/olduname.c	1 pages	39 lines	18/04/21	21:19:46
86	kernel/syscalls/open.c	3 pages	152 lines	21/09/05	20:23:55
87	kernel/syscalls/pause.c	1 pages	30 lines	18/04/21	21:19:46
88	kernel/syscalls/personality.c	1 pages	19 lines	18/04/21	21:19:46
89	kernel/syscalls/pipe.c	2 pages	72 lines	19/11/09	10:13:27
90	kernel/syscalls/read.c	1 pages	49 lines	21/09/04	19:34:37
91	kernel/syscalls/readlink.c	1 pages	57 lines	21/01/07	20:22:53
92	kernel/syscalls/reboot.c	1 pages	49 lines	18/06/11	13:05:20
93	kernel/syscalls/rename.c	2 pages	117 lines	21/07/06	18:09:29
94	kernel/syscalls/rmdir.c	2 pages	87 lines	18/04/21	21:19:46
95	kernel/syscalls/select.c	3 pages	166 lines	21/08/07	17:27:06
96	kernel/syscalls/setdomainname.c	1 pages	38 lines	18/04/21	21:19:46
97	kernel/syscalls/setfsuid.c	1 pages	21 lines	18/04/21	21:19:46
98	kernel/syscalls/setfsuid.c	1 pages	21 lines	18/04/21	21:19:46
99	kernel/syscalls/setgid.c	1 pages	32 lines	18/04/21	21:19:46
100	kernel/syscalls/setgroups.c	1 pages	41 lines	18/04/21	21:19:46
101	kernel/syscalls/sethostname.c	1 pages	42 lines	18/04/21	21:19:46
102	kernel/syscalls/setitimer.c	1 pages	31 lines	18/04/21	21:19:46
103	kernel/syscalls/setpgid.c	2 pages	68 lines	21/12/02	17:08:36
104	kernel/syscalls/setregid.c	1 pages	49 lines	20/05/05	16:52:26
105	kernel/syscalls/setreuid.c	1 pages	53 lines	20/05/05	16:50:01
106	kernel/syscalls/setrlimit.c	1 pages	42 lines	18/04/21	21:19:46
107	kernel/syscalls/setsid.c	1 pages	37 lines	21/12/02	17:08:10
108	kernel/syscalls/settimeofday.c	1 pages	46 lines	18/04/21	21:19:46
109	kernel/syscalls/setuid.c	1 pages	31 lines	18/04/21	21:19:46
110	kernel/syscalls/sgetmask.c	1 pages	20 lines	18/04/21	21:19:46
111	kernel/syscalls/sigaction.c	1 pages	54 lines	20/01/13	11:58:09
112	kernel/syscalls/signal.c	1 pages	56 lines	21/08/25	13:59:44
113	kernel/syscalls/sigpending.c	1 pages	30 lines	18/04/21	21:19:46
114	kernel/syscalls/sigprocmask.c	1 pages	51 lines	21/08/27	16:04:10
115	kernel/syscalls/sigreturn.c	1 pages	31 lines	20/12/07	17:08:39
116	kernel/syscalls/sigsuspend.c	1 pages	44 lines	18/04/21	21:19:46
117	kernel/syscalls/socketcall.c	1 pages	24 lines	18/10/20	11:31:46
118	kernel/syscalls/ssetmask.c	1 pages	26 lines	20/12/03	18:52:06
119	kernel/syscalls/stat.c	1 pages	52 lines	18/04/21	21:19:46
120	kernel/syscalls/statfs.c	1 pages	47 lines	18/04/21	21:19:46
121	kernel/syscalls/stime.c	1 pages	35 lines	18/04/21	21:19:46
122	kernel/syscalls/symlink.c	2 pages	73 lines	18/04/21	21:19:46
123	kernel/syscalls/sync.c	1 pages	27 lines	18/04/21	21:19:46
124	kernel/syscalls/sysinfo.c	1 pages	51 lines	21/12/02	17:07:36
125	kernel/syscalls/time.c	1 pages	37 lines	18/04/21	21:19:46
126	kernel/syscalls/times.c	1 pages	37 lines	18/04/21	21:19:46
127	kernel/syscalls/truncate.c	1 pages	66 lines	18/04/21	21:19:46
128	kernel/syscalls/umask.c	1 pages	27 lines	18/04/21	21:19:46
129	kernel/syscalls/umount2.c	2 pages	115 lines	21/10/16	09:02:19
130	kernel/syscalls/umount.c	1 pages	22 lines	18/04/21	21:19:46
131	kernel/syscalls/uname.c	1 pages	34 lines	21/05/10	18:43:05
132	kernel/syscalls/unlink.c	2 pages	78 lines	18/04/21	21:19:46
133	kernel/syscalls/ustat.c	1 pages	44 lines	18/04/21	21:19:46
134	kernel/syscalls/utime.c	2 pages	72 lines	18/04/21	21:19:46

Table of Contents

Page -3/5

135	kernel/syscalls/wait4.c	2 pages	99 lines	21/12/02	17:06:09
136	kernel/syscalls/waitpid.c	1 pages	23 lines	20/08/13	09:31:29
137	kernel/syscalls/write.c	1 pages	49 lines	21/08/27	22:56:45
138	drivers/block/dma.c	2 pages	93 lines	18/04/21	21:19:46
139	drivers/block/floppy.c	15 pages	885 lines	21/05/24	12:53:08
140	drivers/block/ide.c	15 pages	852 lines	21/11/30	12:25:45
141	drivers/block/ide_cd.c	9 pages	518 lines	21/09/21	18:13:56
142	drivers/block/ide_hd.c	9 pages	522 lines	21/10/26	09:41:48
143	drivers/block/Makefile	1 pages	19 lines	21/05/06	21:14:52
144	drivers/block/part.c	1 pages	34 lines	18/04/21	21:19:46
145	drivers/block/ramdisk.c	3 pages	194 lines	21/09/09	22:16:43
146	drivers/char/console.c	18 pages	1057 lines	21/10/19	11:58:58
147	drivers/char/defkeymap.c	5 pages	149 lines	21/05/24	12:49:58
148	drivers/char/fb.c	3 pages	145 lines	21/05/28	11:09:23
149	drivers/char/keyboard.c	12 pages	786 lines	21/10/05	09:14:09
150	drivers/char/lp.c	4 pages	216 lines	20/03/29	10:35:31
151	drivers/char/Makefile	1 pages	20 lines	21/05/08	18:14:03
152	drivers/char/memdev.c	9 pages	524 lines	21/10/20	10:36:01
153	drivers/char/serial.c	9 pages	570 lines	21/05/24	12:55:25
154	drivers/char/sysrq.c	1 pages	67 lines	21/12/06	21:02:12
155	drivers/char/tty.c	16 pages	1001 lines	21/12/06	20:46:22
156	drivers/char/tty_queue.c	4 pages	261 lines	21/05/24	12:56:45
157	drivers/char/vt.c	4 pages	218 lines	21/06/16	18:30:22
158	drivers/video/fbcon.c	10 pages	589 lines	21/10/02	19:21:37
159	drivers/video/font-lat9-8x10.c	47 pages	3097 lines	21/10/02	19:29:23
160	drivers/video/font-lat9-8x12.c	54 pages	3611 lines	21/10/02	19:30:08
161	drivers/video/font-lat9-8x14.c	62 pages	4125 lines	21/10/02	19:31:11
162	drivers/video/font-lat9-8x16.c	70 pages	4639 lines	21/10/02	19:17:24
163	drivers/video/font-lat9-8x8.c	39 pages	2583 lines	21/10/02	19:23:04
164	drivers/video/fonts.c	1 pages	37 lines	21/05/05	21:42:08
165	drivers/video/Makefile	1 pages	19 lines	21/05/07	17:35:04
166	drivers/video/vgacon.c	6 pages	350 lines	21/10/14	10:55:10
167	drivers/video/video_init.c	1 pages	30 lines	21/05/23	19:52:22
168	fs/buffer.c	8 pages	478 lines	21/12/01	12:07:33
169	fs/devices.c	6 pages	365 lines	21/08/26	09:01:50
170	fs/elf.c	10 pages	582 lines	21/09/09	18:58:24
171	fs/ext2	0 pages	0 lines	21/12/08	07:48:05
172	fs/fd.c	1 pages	50 lines	18/04/21	21:19:46
173	fs/filesystems.c	2 pages	81 lines	18/04/21	21:19:46
174	fs/inode.c	7 pages	441 lines	21/09/30	17:06:54
175	fs/iso9660	0 pages	0 lines	21/12/08	07:48:05
176	fs/locks.c	4 pages	208 lines	21/08/26	08:50:09
177	fs/Makefile	1 pages	25 lines	19/11/23	11:39:20
178	fs/minix	0 pages	0 lines	21/12/08	07:48:05
179	fs/namei.c	3 pages	179 lines	21/10/18	16:46:14
180	fs/pipefs	0 pages	0 lines	21/12/08	07:48:05
181	fs/procfs	0 pages	0 lines	21/12/08	07:48:05
182	fs/script.c	2 pages	73 lines	19/12/24	18:45:20
183	fs/super.c	4 pages	227 lines	21/10/16	08:57:34
184	fs/ext2/bitmaps.c	5 pages	309 lines	19/11/09	10:14:21
185	fs/ext2/dir.c	3 pages	149 lines	21/12/04	17:41:38
186	fs/ext2/file.c	2 pages	131 lines	21/08/25	14:00:47
187	fs/ext2/inode.c	8 pages	456 lines	21/08/31	10:53:11
188	fs/ext2/Makefile	1 pages	19 lines	19/06/15	17:54:13
189	fs/ext2/namei.c	13 pages	798 lines	21/09/16	10:40:49
190	fs/ext2/super.c	4 pages	217 lines	21/11/30	08:31:18
191	fs/ext2/symlink.c	3 pages	134 lines	19/06/15	18:13:30
192	fs/iso9660/dir.c	4 pages	176 lines	21/08/26	08:52:06
193	fs/iso9660/file.c	2 pages	78 lines	18/04/21	21:19:46
194	fs/iso9660/inode.c	3 pages	179 lines	21/07/14	17:52:21
195	fs/iso9660/Makefile	1 pages	19 lines	18/04/21	21:19:46
196	fs/iso9660/namei.c	2 pages	121 lines	21/07/07	18:17:10
197	fs/iso9660/rrip.c	6 pages	340 lines	18/04/21	21:19:46
198	fs/iso9660/super.c	4 pages	224 lines	18/04/21	21:19:46
199	fs/iso9660/symlink.c	2 pages	108 lines	21/08/26	08:54:54
200	fs/minix/bitmaps.c	3 pages	168 lines	19/08/16	17:17:09
201	fs/minix/dir.c	3 pages	146 lines	21/12/04	17:41:16

Table of Contents

Page -4/5

202	fs/minix/file.c	2	pages	131	lines	21/08/25	14:00:54
203	fs/minix/inode.c	2	pages	75	lines	19/11/09	10:08:51
204	fs/minix/Makefile	1	pages	19	lines	18/04/21	21:19:46
205	fs/minix/namei.c	12	pages	724	lines	21/09/16	10:41:15
206	fs/minix/super.c	5	pages	268	lines	21/10/22	09:05:05
207	fs/minix/symlink.c	3	pages	136	lines	18/04/21	21:19:46
208	fs/minix/v1_inode.c	7	pages	407	lines	21/07/23	07:44:32
209	fs/minix/v2_inode.c	8	pages	470	lines	21/07/23	07:44:01
210	fs/pipefs/fifo.c	2	pages	73	lines	21/10/20	15:48:31
211	fs/pipefs/Makefile	1	pages	19	lines	18/04/21	21:19:46
212	fs/pipefs/pipe.c	4	pages	217	lines	21/10/19	11:56:53
213	fs/pipefs/super.c	2	pages	113	lines	19/11/09	10:10:58
214	fs/procfs/data.c	14	pages	831	lines	21/10/07	07:45:58
215	fs/procfs/dir.c	4	pages	247	lines	21/09/14	06:58:18
216	fs/procfs/file.c	2	pages	122	lines	21/08/25	14:01:07
217	fs/procfs/inode.c	2	pages	88	lines	21/06/22	18:56:28
218	fs/procfs/Makefile	1	pages	19	lines	18/04/21	21:19:46
219	fs/procfs/namei.c	2	pages	87	lines	21/09/14	06:58:52
220	fs/procfs/super.c	2	pages	82	lines	18/04/21	21:19:46
221	fs/procfs/symlink.c	3	pages	148	lines	21/06/22	18:49:30
222	fs/procfs/tree.c	2	pages	115	lines	21/01/07	20:03:50
223	mm/alloc.c	1	pages	35	lines	20/09/14	14:18:41
224	mm/bios_map.c	3	pages	153	lines	21/10/09	10:41:36
225	mm/fault.c	5	pages	263	lines	21/10/07	09:29:24
226	mm/Makefile	1	pages	19	lines	18/04/21	21:19:46
227	mm/memory.c	8	pages	468	lines	21/12/07	19:00:47
228	mm/mmap.c	8	pages	516	lines	21/08/22	20:01:16
229	mm/page.c	7	pages	425	lines	21/09/30	17:12:52
230	mm/swapper.c	1	pages	65	lines	21/09/29	11:31:24
231	lib/ctype.c	3	pages	140	lines	20/01/26	21:03:16
232	lib/Makefile	1	pages	19	lines	18/04/21	21:19:46
233	lib/printk.c	6	pages	360	lines	21/10/09	10:46:31
234	lib/strings.c	5	pages	280	lines	18/04/21	21:19:46
235	include/fiwix/asm.h	3	pages	136	lines	21/05/24	12:47:55
236	include/fiwix/bios.h	1	pages	26	lines	21/05/24	16:18:23
237	include/fiwix/buffer.h	1	pages	47	lines	21/10/05	17:31:15
238	include/fiwix/cmos.h	1	pages	58	lines	18/04/21	21:19:46
239	include/fiwix/config.h	1	pages	61	lines	21/12/03	19:41:52
240	include/fiwix/console.h	3	pages	185	lines	21/09/10	19:51:40
241	include/fiwix/cpu.h	2	pages	68	lines	18/04/21	21:19:46
242	include/fiwix/ctype.h	1	pages	38	lines	18/04/21	21:19:46
243	include/fiwix/devices.h	1	pages	49	lines	20/03/30	10:42:21
244	include/fiwix/dirent.h	1	pages	21	lines	18/04/21	21:19:46
245	include/fiwix/dma.h	1	pages	33	lines	18/04/21	21:19:46
246	include/fiwix/errno.h	3	pages	132	lines	18/04/21	21:19:46
247	include/fiwix/fbcon.h	1	pages	29	lines	21/05/19	10:59:40
248	include/fiwix/fb.h	1	pages	25	lines	21/02/28	15:02:13
249	include/fiwix/fcntl.h	1	pages	67	lines	18/04/21	21:19:46
250	include/fiwix/filesystems.h	3	pages	168	lines	19/11/09	10:04:45
251	include/fiwix/floppy.h	2	pages	101	lines	20/04/09	19:06:41
252	include/fiwix/font.h	1	pages	27	lines	21/10/02	19:17:58
253	include/fiwix/fs_ext2.h	4	pages	251	lines	19/08/18	18:47:16
254	include/fiwix/fs.h	4	pages	263	lines	21/07/12	19:01:27
255	include/fiwix/fs_iso9660.h	4	pages	216	lines	18/04/21	21:19:46
256	include/fiwix/fs_minix.h	2	pages	131	lines	19/11/09	10:05:13
257	include/fiwix/fs_pipe.h	1	pages	23	lines	18/04/21	21:19:46
258	include/fiwix/fs_proc.h	2	pages	91	lines	21/01/06	11:19:23
259	include/fiwix/i386elf.h	5	pages	279	lines	18/04/21	21:19:46
260	include/fiwix/ide_cd.h	1	pages	24	lines	18/04/21	21:19:46
261	include/fiwix/ide.h	5	pages	279	lines	21/11/30	09:05:26
262	include/fiwix/ide_hd.h	1	pages	23	lines	18/04/21	21:19:46
263	include/fiwix/ioctl.h	2	pages	89	lines	18/04/21	21:19:46
264	include/fiwix/kd.h	3	pages	167	lines	18/04/21	21:19:46
265	include/fiwix/kernel.h	2	pages	92	lines	21/10/07	09:13:42
266	include/fiwix/keyboard.h	3	pages	157	lines	21/05/24	16:17:39
267	include/fiwix/kparms.h	1	pages	67	lines	20/05/21	13:20:00
268	include/fiwix/limits.h	1	pages	27	lines	20/05/21	13:24:50

Table of Contents

Page -5/5

269	include/fiwix/locks.h	1 pages	29 lines	18/04/21	21:19:46
270	include/fiwix/lp.h	1 pages	48 lines	18/04/21	21:19:46
271	include/fiwix/memdev.h	1 pages	57 lines	18/04/25	22:41:18
272	include/fiwix/mman.h	1 pages	59 lines	21/03/01	18:54:01
273	include/fiwix/mm.h	2 pages	107 lines	21/09/27	17:05:00
274	include/fiwix/multiboot1.h	6 pages	351 lines	21/04/25	18:49:58
275	include/fiwix/part.h	1 pages	38 lines	18/04/21	21:19:46
276	include/fiwix/pic.h	1 pages	58 lines	21/12/04	21:26:21
277	include/fiwix/pit.h	1 pages	51 lines	18/04/21	21:19:46
278	include/fiwix/process.h	3 pages	189 lines	21/11/13	20:45:37
279	include/fiwix/ramdisk.h	1 pages	33 lines	20/12/19	17:50:11
280	include/fiwix/reboot.h	1 pages	23 lines	18/04/21	21:19:46
281	include/fiwix/resource.h	2 pages	81 lines	18/04/21	21:19:46
282	include/fiwix/sched.h	1 pages	55 lines	21/09/13	17:28:34
283	include/fiwix/segments.h	2 pages	78 lines	21/09/28	13:24:12
284	include/fiwix/serial.h	2 pages	114 lines	21/05/24	16:19:11
285	include/fiwix/sigcontext.h	1 pages	32 lines	18/04/21	21:19:46
286	include/fiwix/signal.h	2 pages	92 lines	18/04/21	21:19:46
287	include/fiwix/sleep.h	1 pages	38 lines	21/09/14	09:48:58
288	include/fiwix/statbuf.h	1 pages	48 lines	18/04/21	21:19:46
289	include/fiwix/statfs.h	1 pages	28 lines	18/04/21	21:19:46
290	include/fiwix/stat.h	1 pages	56 lines	18/04/21	21:19:46
291	include/fiwix/stdarg.h	1 pages	42 lines	18/04/21	21:19:46
292	include/fiwix/stdio.h	1 pages	15 lines	18/04/21	21:19:46
293	include/fiwix/string.h	1 pages	42 lines	18/04/21	21:19:46
294	include/fiwix/syscalls.h	3 pages	147 lines	21/08/09	16:53:58
295	include/fiwix/sysrq.h	1 pages	18 lines	21/10/22	12:35:29
296	include/fiwix/system.h	1 pages	29 lines	21/12/08	07:48:24
297	include/fiwix/termbits.h	3 pages	176 lines	20/04/13	11:47:52
298	include/fiwix/termios.h	1 pages	46 lines	20/04/13	11:47:49
299	include/fiwix/timeb.h	1 pages	18 lines	18/04/21	21:19:46
300	include/fiwix/time.h	1 pages	49 lines	18/04/21	21:19:46
301	include/fiwix/timer.h	1 pages	50 lines	20/04/09	19:13:22
302	include/fiwix/times.h	1 pages	18 lines	18/04/21	21:19:46
303	include/fiwix/traps.h	1 pages	48 lines	21/05/24	16:17:18
304	include/fiwix/tty.h	2 pages	99 lines	21/05/24	16:18:50
305	include/fiwix/types.h	1 pages	53 lines	18/04/21	21:19:46
306	include/fiwix/unistd.h	4 pages	207 lines	18/04/21	21:19:46
307	include/fiwix/ustat.h	1 pages	20 lines	18/04/21	21:19:46
308	include/fiwix/utime.h	1 pages	18 lines	18/04/21	21:19:46
309	include/fiwix/utsname.h	2 pages	75 lines	18/04/21	21:19:46
310	include/fiwix/version.h	1 pages	1 lines	21/12/08	08:40:17
311	include/fiwix/vgacon.h	1 pages	51 lines	21/05/19	10:59:57
312	include/fiwix/video.h	1 pages	13 lines	21/04/12	18:42:39
313	include/fiwix/vt.h	1 pages	54 lines	18/04/21	21:19:46
314	README.md	2 pages	88 lines	21/12/08	07:48:36
315	LICENSE	1 pages	22 lines	20/03/20	16:27:21
316	CODING	1 pages	50 lines	20/03/23	10:47:38
317	THANKS	1 pages	13 lines	20/03/26	18:47:04