程序 12-11 linux/fs/char_dev.c

```
1  /*
2   *  linux/fs/char_dev.c
3   *
4   *  (C) 1991  Linus Torvalds
5   */
6
7  #include <errno.h>        // 错误号头文件。包含系统中各种出错号。
8  #include <sys/types.h>    // 类型头文件。定义了基本的系统数据类型。
9
10 #include <linux/sched.h>  // 调度程序头文件，定义任务结构 task_struct、任务 0 数据等。
11 #include <linux/kernel.h> // 内核头文件。含有一些内核常用函数的原形定义。
12
13 #include <asm/segment.h>  // 段操作头文件。定义了有关段寄存器操作的嵌入式汇编函数。
14 #include <asm/io.h>       // io 头文件。定义硬件端口输入/输出宏汇编语句。
15
16 extern int tty_read(unsigned minor,char * buf,int count);     // 终端读。
17 extern int tty_write(unsigned minor,char * buf,int count);    // 终端写。
18
   // 定义字符设备读写函数指针类型。
19 typedef (*crw_ptr)(int rw,unsigned minor,char * buf,int count,off_t * pos);
20
   //// 串口终端读写操作函数。
   // 参数：rw - 读写命令；minor - 终端子设备号；buf - 缓冲区；cout - 读写字节数；
   // pos - 读写操作当前指针，对于终端操作，该指针无用。
   // 返回：实际读写的字节数。若失败则返回出错码。
21 static int rw_ttyx(int rw,unsigned minor,char * buf,int count,off_t * pos)
22 {
23         return ((rw==READ)?tty_read(minor,buf,count):
24                 tty_write(minor,buf,count));
25 }
26
   //// 终端读写操作函数。
   // 同上 rw_ttyx()，只是增加了对进程是否有控制终端的检测。
27 static int rw_tty(int rw,unsigned minor,char * buf,int count, off_t * pos)
28 {
   // 若进程没有对应的控制终端，则返回出错号。否则调用终端读写函数 rw_ttyx()，并返回
   // 实际读写字节数。
29         if (current->tty<0)
30                 return -EPERM;
31         return rw_ttyx(rw,current->tty,buf,count,pos);
32 }
33
   //// 内存数据读写。未实现。
34 static int rw_ram(int rw,char * buf, int count, off_t *pos)
35 {
36         return -EIO;
37 }
38
   //// 物理内存数据读写操作函数。未实现。
39 static int rw_mem(int rw,char * buf, int count, off_t * pos)
40 {
41         return -EIO;
```

```
42 }
43
   //// 内核虚拟内存数据读写函数。未实现。
44 static int rw_kmem(int rw,char * buf, int count, off_t * pos)
45 {
46         return -EIO;
47 }
48
   // 端口读写操作函数。
   // 参数：rw - 读写命令；buf - 缓冲区；cout - 读写字节数；pos - 端口地址。
   // 返回：实际读写的字节数。
49 static int rw_port(int rw,char * buf, int count, off_t * pos)
50 {
51         int i=*pos;
52
   // 对于所要求读写的字节数，并且端口地址小于 64k 时，循环执行单个字节的读写操作。
   // 若是读命令，则从端口 i 中读取一字节内容并放到用户缓冲区中。若是写命令，则从用
   // 户数据缓冲区中取一字节输出到端口 i。
53         while (count-->0 && i<65536) {
54                 if (rw==READ)
55                         put_fs_byte(inb(i),buf++);
56                 else
57                         outb(get_fs_byte(buf++),i);
58                 i++;                                    // 前移一个端口。[??]
59         }
   // 然后计算读/写的字节数，调整相应读写指针，并返回读/写的字节数。
60         i -= *pos;
61         *pos += i;
62         return i;
63 }
64
   //// 内存读写操作函数。内存主设备号是 1。这里仅给出对 0-5 子设备的处理。
65 static int rw_memory(int rw, unsigned minor, char * buf, int count, off_t * pos)
66 {
   // 根据内存设备子设备号，分别调用不同的内存读写函数。
67         switch(minor) {
68                 case 0:         // 对应设备文件名是 /dev/ram0 或/dev/ramdisk。
69                         return rw_ram(rw,buf,count,pos);
70                 case 1:         // /dev/ram1 或/dev/mem 或 ram。
71                         return rw_mem(rw,buf,count,pos);
72                 case 2:         // /dev/ram2 或/dev/kmem。
73                         return rw_kmem(rw,buf,count,pos);
74                 case 3:         // /dev/null。
75                         return (rw==READ)?0:count;      /* rw_null */
76                 case 4:         // /dev/port。
77                         return rw_port(rw,buf,count,pos);
78                 default:
79                         return -EIO;
80         }
81 }
82
   // 定义系统中设备种数。
83 #define NRDEVS ((sizeof (crw_table))/(sizeof (crw_ptr)))
```

```
84
   // 字符设备读写函数指针表。
85 static crw_ptr crw_table[]={
86         NULL,           /* nodev */             /* 无设备(空设备) */
87         rw_memory,      /* /dev/mem etc */      /* /dev/mem 等 */
88         NULL,           /* /dev/fd */           /* /dev/fd 软驱 */
89         NULL,           /* /dev/hd */           /* /dev/hd 硬盘 */
90         rw_ttyx,        /* /dev/ttyx */         /* /dev/ttyx 串口终端 */
91         rw_tty,         /* /dev/tty */          /* /dev/tty 终端 */
92         NULL,           /* /dev/lp */           /* /dev/lp 打印机 */
93         NULL};          /* unnamed pipes */     /* 未命名管道 */
94
   //// 字符设备读写操作函数。
   // 参数：rw -读写命令；dev -设备号；buf -缓冲区；count -读写字节数；pos -读写指针。
   // 返回：实际读/写字节数。
95 int rw_char(int rw,int dev, char * buf, int count, off_t * pos)
96 {
97         crw_ptr call_addr;
98
   // 如果设备号超出系统设备数，则返回出错码。如果该设备没有对应的读/写函数，也返回出
   // 错码。否则调用对应设备的读写操作函数，并返回实际读/写的字节数。
99         if (MAJOR(dev)>=NRDEVS)
100                return -ENODEV;
101        if (!(call_addr=crw_table[MAJOR(dev)]))
102                return -ENODEV;
103        return call_addr(rw,MINOR(dev),buf,count,pos);
104 }
105
```