程序 14-10 linux/include/termios.h

```
1  #ifndef _TERMIOS_H
2  #define _TERMIOS_H
3
4  #include <sys/types.h>
5
6  #define TTY_BUF_SIZE 1024            // tty 中的缓冲区长度。
7
8  /* 0x54 is just a magic number to make these relatively uniqe ('T') */
   /* 0x54 只是一个魔数，目的是为了使这些常数唯一('T') */
9
   // tty 设备的 ioctl 调用命令集。ioctl 将命令编码在低位字中。
   // 下面名称 TC[*]的含义是 tty 控制命令。
   // 取相应终端 termios 结构中的信息(参见 tcgetattr())。
10 #define TCGETS          0x5401
   // 设置相应终端 termios 结构中的信息(参见 tcsetattr(), TCSANOW)。
11 #define TCSETS          0x5402
   // 在设置终端 termios 的信息之前，需要先等待输出队列中所有数据处理完(耗尽)。对于修改参数
   // 会影响输出的情况，就需要使用这种形式(参见 tcsetattr(), TCSADRAIN 选项)。
12 #define TCSETSW         0x5403
   // 在设置 termios 的信息之前，需要先等待输出队列中所有数据处理完，并且刷新(清空)输入队列。
   // 再设置（参见 tcsetattr(), TCSAFLUSH 选项）。
13 #define TCSETSF         0x5404
   // 取相应终端 termio 结构中的信息(参见 tcgetattr())。
14 #define TCGETA          0x5405
   // 设置相应终端 termio 结构中的信息(参见 tcsetattr(), TCSANOW 选项)。
15 #define TCSETA          0x5406
   // 在设置终端 termio 的信息之前，需要先等待输出队列中所有数据处理完(耗尽)。对于修改参数
   // 会影响输出的情况，就需要使用这种形式(参见 tcsetattr(), TCSADRAIN 选项)。
16 #define TCSETAW         0x5407
   // 在设置 termio 的信息之前，需要先等待输出队列中所有数据处理完，并且刷新(清空)输入队列。
   // 再设置（参见 tcsetattr(), TCSAFLUSH 选项）。
17 #define TCSETAF         0x5408
   // 等待输出队列处理完毕(空)，若参数值是 0，则发送一个 break（参见 tcsendbreak(), tcdrain()）。
18 #define TCSBRK          0x5409
   // 开始/停止控制。如果参数值是 0，则挂起输出；如果是 1，则重新开启挂起的输出；如果是 2，
   // 则挂起输入；如果是 3，则重新开启挂起的输入（参见 tcflow()）。
19 #define TCXONC          0x540A
   // 刷新已写输出但还没发送或已收但还没有读数据。如果参数是 0，则刷新(清空)输入队列；如果
   // 是 1，则刷新输出队列；如果是 2，则刷新输入和输出队列（参见 tcflush()）。
20 #define TCFLSH          0x540B
   // 下面名称 TIOC[*]的含义是 tty 输入输出控制命令。
   // 设置终端串行线路专用模式。
21 #define TIOCEXCL        0x540C
   // 复位终端串行线路专用模式。
22 #define TIOCNXCL        0x540D
   // 设置 tty 为控制终端。(TIOCNOTTY - 禁止 tty 为控制终端)。
23 #define TIOCSCTTY       0x540E
   // 读取指定终端设备进程的组 id，参见 tcgetpgrp()。该常数符号名称是"Terminal IO Control
   // Get PGRP "的缩写。读取前台进程组 ID。
24 #define TIOCGPGRP       0x540F
   // 设置指定终端设备进程的组 id(参见 tcsetpgrp())。
25 #define TIOCSPGRP       0x5410
```

```
    // 返回输出队列中还未送出的字符数。
26  #define TIOCOUTQ        0x5411
    // 模拟终端输入。该命令以一个指向字符的指针作为参数，并假装该字符是在终端上键入的。用户
    // 必须在该控制终端上具有超级用户权限或具有读许可权限。
27  #define TIOCSTI         0x5412
    // 读取终端设备窗口大小信息（参见 winsize 结构）。
28  #define TIOCGWINSZ      0x5413
    // 设置终端设备窗口大小信息（参见 winsize 结构）。
29  #define TIOCSWINSZ      0x5414
    // 返回 modem 状态控制引线的当前状态比特位标志集（参见下面 185-196 行）。
30  #define TIOCMGET        0x5415
    // 设置单个 modem 状态控制引线的状态(true 或 false)(Individual control line Set)。
31  #define TIOCMBIS        0x5416
    // 复位单个 modem 状态控制引线的状态(Individual control line clear)。
32  #define TIOCMBIC        0x5417
    // 设置 modem 状态引线的状态。如果某一比特位置位，则 modem 对应的状态引线将置为有效。
33  #define TIOCMSET        0x5418
    // 读取软件载波检测标志(1 - 开启；0 - 关闭)。
    // 对于本地连接的终端或其他设备，软件载波标志是开启的，对于使用 modem 线路的终端或设备
    // 则是关闭的。为了能使用这两个 ioctl 调用，tty 线路应该是以 O_NDELAY 方式打开的，这样
    // open()就不会等待载波。
34  #define TIOCGSOFTCAR    0x5419
    // 设置软件载波检测标志(1 - 开启；0 - 关闭)。
35  #define TIOCSSOFTCAR    0x541A
    // 返回输入队列中还未取走字符的数目。
36  #define FIONREAD        0x541B
37  #define TIOCINQ         FIONREAD
38
    // 窗口大小(Window size)属性结构。在窗口环境中可用于基于屏幕的应用程序。
    // ioctls 中的 TIOCGWINSZ 和 TIOCSWINSZ 可用来读取或设置这些信息。
39  struct winsize {
40          unsigned short ws_row;      // 窗口字符行数。
41          unsigned short ws_col;      // 窗口字符列数。
42          unsigned short ws_xpixel;   // 窗口宽度，象素值。
43          unsigned short ws_ypixel;   // 窗口高度，象素值。
44  };
45
    // AT&T 系统 V 的 termio 结构。
46  #define NCC 8                       // termio 结构中控制字符数组的长度。
47  struct termio {
48          unsigned short c_iflag;     /* input mode flags */      // 输入模式标志。
49          unsigned short c_oflag;     /* output mode flags */     // 输出模式标志。
50          unsigned short c_cflag;     /* control mode flags */    // 控制模式标志。
51          unsigned short c_lflag;     /* local mode flags */      // 本地模式标志。
52          unsigned char c_line;       /* line discipline */       // 线路规程（速率）。
53          unsigned char c_cc[NCC];    /* control characters */    // 控制字符数组。
54  };
55
    // POSIX 的 termios 结构。
56  #define NCCS 17                     // termios 结构中控制字符数组长度。
57  struct termios {
58          tcflag_t c_iflag;           /* input mode flags */      // 输入模式标志。
59          tcflag_t c_oflag;           /* output mode flags */     // 输出模式标志。
```

```
60          tcflag_t c_cflag;           /* control mode flags */    // 控制模式标志。
61          tcflag_t c_lflag;           /* local mode flags */      // 本地模式标志。
62          cc_t c_line;                /* line discipline */       // 线路规程（速率）。
63          cc_t c_cc[NCCS];            /* control characters */    // 控制字符数组。
64  };
65
    // 以下是控制字符数组 c_cc[]中项的索引值。该数组初始值定义在 include/linux/tty.h 中。
    // 程序可以更改这个数组中的值。如果定义了_POSIX_VDISABLE（\0），那么当数组某一项值
    // 等于 _POSIX_VDISABLE 的值时，表示禁止使用数组中相应的特殊字符。
66  /* c_cc characters */    /* c_cc 数组中的字符 */
67  #define VINTR 0          // c_cc[VINTR]    = INTR    (^C), \003, 中断字符。
68  #define VQUIT 1          // c_cc[VQUIT]    = QUIT    (^\), \034, 退出字符。
69  #define VERASE 2         // c_cc[VERASE]   = ERASE   (^H), \177, 擦出字符。
70  #define VKILL 3          // c_cc[VKILL]    = KILL    (^U), \025, 终止字符（删除行）。
71  #define VEOF 4           // c_cc[VEOF]     = EOF     (^D), \004, 文件结束字符。
72  #define VTIME 5          // c_cc[VTIME]    = TIME    (\0), \0,   定时器值(参见后面说明)。
73  #define VMIN 6           // c_cc[VMIN]     = MIN     (\1), \1,   定时器值。
74  #define VSWTC 7          // c_cc[VSWTC]    = SWTC    (\0), \0,   交换字符。
75  #define VSTART 8         // c_cc[VSTART]   = START   (^Q), \021, 开始字符。
76  #define VSTOP 9          // c_cc[VSTOP]    = STOP    (^S), \023, 停止字符。
77  #define VSUSP 10         // c_cc[VSUSP]    = SUSP    (^Z), \032, 挂起字符。
78  #define VEOL 11          // c_cc[VEOL]     = EOL     (\0), \0,   行结束字符。
79  #define VREPRINT 12      // c_cc[VREPRINT] = REPRINT (^R), \022, 重显示字符。
80  #define VDISCARD 13      // c_cc[VDISCARD] = DISCARD (^O), \017, 丢弃字符。
81  #define VWERASE 14       // c_cc[VWERASE]  = WERASE  (^W), \027, 单词擦除字符。
82  #define VLNEXT 15        // c_cc[VLNEXT]   = LNEXT   (^V), \026, 下一行字符。
83  #define VEOL2 16         // c_cc[VEOL2]    = EOL2    (\0), \0,   行结束字符 2。
84
    // termios 结构输入模式字段 c_iflag 各种标志的符号常数。
85  /* c_iflag bits */       /* c_iflag 比特位 */
86  #define IGNBRK  0000001          // 输入时忽略 BREAK 条件。
87  #define BRKINT  0000002          // 在 BREAK 时产生 SIGINT 信号。
88  #define IGNPAR  0000004          // 忽略奇偶校验出错的字符。
89  #define PARMRK  0000010          // 标记奇偶校验错。
90  #define INPCK   0000020          // 允许输入奇偶校验。
91  #define ISTRIP  0000040          // 屏蔽字符第 8 位。
92  #define INLCR   0000100          // 输入时将换行符 NL 映射成回车符 CR。
93  #define IGNCR   0000200          // 忽略回车符 CR。
94  #define ICRNL   0000400          // 在输入时将回车符 CR 映射成换行符 NL。
95  #define IUCLC   0001000          // 在输入时将大写字符转换成小写字符。
96  #define IXON    0002000          // 允许开始/停止（XON/XOFF）输出控制。
97  #define IXANY   0004000          // 允许任何字符重启输出。
98  #define IXOFF   0010000          // 允许开始/停止（XON/XOFF）输入控制。
99  #define IMAXBEL 0020000          // 输入队列满时响铃。
100
    // termios 结构中输出模式字段 c_oflag 各种标志的符号常数。
101 /* c_oflag bits */       /* c_oflag 比特位 */
102 #define OPOST   0000001          // 执行输出处理。
103 #define OLCUC   0000002          // 在输出时将小写字符转换成大写字符。
104 #define ONLCR   0000004          // 在输出时将换行符 NL 映射成回车-换行符 CR-NL。
105 #define OCRNL   0000010          // 在输出时将回车符 CR 映射成换行符 NL。
106 #define ONOCR   0000020          // 在 0 列不输出回车符 CR。
107 #define ONLRET  0000040          // 换行符 NL 执行回车符的功能。
```

```
108 #define OFILL   0000100         // 延迟时使用填充字符而不使用时间延迟。
109 #define OFDEL   0000200         // 填充字符是 ASCII 码 DEL。如果未设置，则使用 ASCII NULL。
110 #define NLDLY   0000400         // 选择换行延迟。
111 #define   NL0   0000000         // 换行延迟类型 0。
112 #define   NL1   0000400         // 换行延迟类型 1。
113 #define CRDLY   0003000         // 选择回车延迟。
114 #define   CR0   0000000         // 回车延迟类型 0。
115 #define   CR1   0001000         // 回车延迟类型 1。
116 #define   CR2   0002000         // 回车延迟类型 2。
117 #define   CR3   0003000         // 回车延迟类型 3。
118 #define TABDLY  0014000         // 选择水平制表延迟。
119 #define   TAB0  0000000         // 水平制表延迟类型 0。
120 #define   TAB1  0004000         // 水平制表延迟类型 1。
121 #define   TAB2  0010000         // 水平制表延迟类型 2。
122 #define   TAB3  0014000         // 水平制表延迟类型 3。
123 #define   XTABS 0014000         // 将制表符 TAB 换成空格，该值表示空格数。
124 #define BSDLY   0020000         // 选择退格延迟。
125 #define   BS0   0000000         // 退格延迟类型 0。
126 #define   BS1   0020000         // 退格延迟类型 1。
127 #define VTDLY   0040000         // 纵向制表延迟。
128 #define   VT0   0000000         // 纵向制表延迟类型 0。
129 #define   VT1   0040000         // 纵向制表延迟类型 1。
130 #define FFDLY   0040000         // 选择换页延迟。
131 #define   FF0   0000000         // 换页延迟类型 0。
132 #define   FF1   0040000         // 换页延迟类型 1。
133
    // termios 结构中控制模式标志字段 c_cflag 标志的符号常数（8 进制数）。
134 /* c_cflag bit meaning */     /* c_cflag 比特位的含义 */
135 #define CBAUD   0000017         // 传输速率位屏蔽码。
136 #define  B0     0000000         /* hang up */  /* 挂断线路 */
137 #define  B50    0000001         // 波特率 50。
138 #define  B75    0000002         // 波特率 75。
139 #define  B110   0000003         // 波特率 110。
140 #define  B134   0000004         // 波特率 134。
141 #define  B150   0000005         // 波特率 150。
142 #define  B200   0000006         // 波特率 200。
143 #define  B300   0000007         // 波特率 300。
144 #define  B600   0000010         // 波特率 600。
145 #define  B1200  0000011         // 波特率 1200。
146 #define  B1800  0000012         // 波特率 1800。
147 #define  B2400  0000013         // 波特率 2400。
148 #define  B4800  0000014         // 波特率 4800。
149 #define  B9600  0000015         // 波特率 9600。
150 #define  B19200 0000016         // 波特率 19200。
151 #define  B38400 0000017         // 波特率 38400。
152 #define EXTA B19200             // 扩展波特率 A。
153 #define EXTB B38400             // 扩展波特率 B。

154 #define CSIZE   0000060         // 字符位宽度屏蔽码。
155 #define   CS5   0000000         // 每字符 5 比特位。
156 #define   CS6   0000020         // 每字符 6 比特位。
157 #define   CS7   0000040         // 每字符 7 比特位。
158 #define   CS8   0000060         // 每字符 8 比特位。
```

```
159 #define CSTOPB  0000100         // 设置两个停止位，而不是 1 个。
160 #define CREAD   0000200         // 允许接收。
161 #define PARENB  0000400         // 开启输出时产生奇偶位、输入时进行奇偶校验。
162 #define PARODD  0001000         // 输入/输入校验是奇校验。
163 #define HUPCL   0002000         // 最后进程关闭后挂断。
164 #define CLOCAL  0004000         // 忽略调制解调器(modem)控制线路。
165 #define CIBAUD  03600000        /* input baud rate (not used) */  /* 输入波特率(未使用) */
166 #define CRTSCTS 020000000000    /* flow control */  /* 流控制 */
167
    // termios 结构中本地模式标志字段 c_lflag 的符号常数。
168 /* c_lflag bits */      /* c_lflag 比特位 */
169 #define ISIG    0000001         // 当收到字符 INTR、QUIT、SUSP 或 DSUSP，产生相应的信号。
170 #define ICANON  0000002         // 开启规范模式（熟模式）。
171 #define XCASE   0000004         // 若设置了 ICANON，则终端是大写字符的。
172 #define ECHO    0000010         // 回显输入字符。
173 #define ECHOE   0000020         // 若设置了 ICANON，则 ERASE/WERASE 将擦除前一字符/单词。
174 #define ECHOK   0000040         // 若设置了 ICANON，则 KILL 字符将擦除当前行。
175 #define ECHONL  0000100         // 如设置了 ICANON，则即使 ECHO 没有开启也回显 NL 字符。
176 #define NOFLSH  0000200         // 当生成 SIGINT 和 SIGQUIT 信号时不刷新输入输出队列，当
                                    // 生成 SIGSUSP 信号时，刷新输入队列。
177 #define TOSTOP  0000400         // 发送 SIGTTOU 信号到后台进程的进程组，该后台进程试图写
                                    // 自己的控制终端。
178 #define ECHOCTL 0001000         // 若设置了 ECHO，则除 TAB、NL、START 和 STOP 以外的 ASCII
                                    // 控制信号将被回显成象^X 式样，X 值是控制符+0x40。
179 #define ECHOPRT 0002000         // 若设置了 ICANON 和 IECHO，则字符在擦除时将显示。
180 #define ECHOKE  0004000         // 若设置了 ICANON，则 KILL 通过擦除行上的所有字符被回显。
181 #define FLUSHO  0010000         // 输出被刷新。通过键入 DISCARD 字符，该标志被翻转。
182 #define PENDIN  0040000         // 当下一个字符是读时，输入队列中的所有字符将被重显。
183 #define IEXTEN  0100000         // 开启实现时定义的输入处理。
184
185 /* modem lines */    /* modem 线路信号符号常数 */
186 #define TIOCM_LE        0x001       // 线路允许(Line Enable)。
187 #define TIOCM_DTR       0x002       // 数据终端就绪(Data Terminal Ready)。
188 #define TIOCM_RTS       0x004       // 请求发送(Request to Send)。
189 #define TIOCM_ST        0x008       // 串行数据发送(Serial Transfer)。[??]
190 #define TIOCM_SR        0x010       // 串行数据接收(Serial Receive)。[??]
191 #define TIOCM_CTS       0x020       // 清除发送(Clear To Send)。
192 #define TIOCM_CAR       0x040       // 载波监测(Carrier Detect)。
193 #define TIOCM_RNG       0x080       // 响铃指示(Ring indicate)。
194 #define TIOCM_DSR       0x100       // 数据设备就绪(Data Set Ready)。
195 #define TIOCM_CD        TIOCM_CAR
196 #define TIOCM_RI        TIOCM_RNG
197
198 /* tcflow() and TCXONC use these */   /* tcflow()和 TCXONC 使用这些符号常数 */
199 #define TCOOFF          0           // 挂起输出（是"Terminal Control Output OFF"的缩写）。
200 #define TCOON           1           // 重启被挂起的输出。
201 #define TCIOFF          2           // 系统传输一个 STOP 字符，使设备停止向系统传输数据。
202 #define TCION           3           // 系统传输一个 START 字符，使设备开始向系统传输数据。
203
204 /* tcflush() and TCFLSH use these */  /* tcflush()和 TCFLSH 使用这些符号常数 */
205 #define TCIFLUSH        0           // 清接收到的数据但不读。
206 #define TCOFLUSH        1           // 清已写的数据但不传送。
207 #define TCIOFLUSH       2           // 清接收到的数据但不读。清已写的数据但不传送。
```

```
208
209 /* tcsetattr uses these */            /* tcsetattr()使用这些符号常数 */
210 #define TCSANOW         0           // 改变立即发生。
211 #define TCSADRAIN       1           // 改变在所有已写的输出被传输之后发生。
212 #define TCSAFLUSH       2           // 改变在所有已写的输出被传输之后并且在所有接收到但
                                        // 还没有读取的数据被丢弃之后发生。
213
    // 以下这些函数在编译环境的函数库 libc.a 中实现，内核中没有。在函数库实现中，这些函数通过
    // 调用系统调用 ioctl()来实现。有关 ioctl()系统调用，请参见 fs/ioctl.c 程序。
    // 返回 termios_p 所指 termios 结构中的接收波特率。
214 extern speed_t cfgetispeed(struct termios *termios_p);
    // 返回 termios_p 所指 termios 结构中的发送波特率。
215 extern speed_t cfgetospeed(struct termios *termios_p);
    // 将 termios_p 所指 termios 结构中的接收波特率设置为 speed。
216 extern int cfsetispeed(struct termios *termios_p, speed_t speed);
    // 将 termios_p 所指 termios 结构中的发送波特率设置为 speed。
217 extern int cfsetospeed(struct termios *termios_p, speed_t speed);
    // 等待 fildes 所指对象已写输出数据被传送出去。
218 extern int tcdrain(int fildes);
    // 挂起/重启 fildes 所指对象数据的接收和发送。
219 extern int tcflow(int fildes, int action);
    // 丢弃 fildes 指定对象所有已写但还没传送以及所有已收到但还没有读取的数据。
220 extern int tcflush(int fildes, int queue_selector);
    // 获取与句柄 fildes 对应对象的参数，并将其保存在 termios_p 所指的地方。
221 extern int tcgetattr(int fildes, struct termios *termios_p);
    // 如果终端使用异步串行数据传输，则在一定时间内连续传输一系列 0 值比特位。
222 extern int tcsendbreak(int fildes, int duration);
    // 使用 termios 结构指针 termios_p 所指的数据，设置与终端相关的参数。
223 extern int tcsetattr(int fildes, int optional_actions,
224         struct termios *termios_p);
225
226 #endif
227
```