更多嵌入式 Linux 学习资料，请关注：一口 Linux 回复关键字:1024

个人微信

公众号

一、数据成员

termios 函数族提供了一个常规的终端接口，用于控制非同步通信端口。 这个结构包含了至少下列成员：

```
struct termios
{
    unsigned short c_iflag; /* 输入模式标志*/
    unsigned short c_oflag; /* 输出模式标志*/
    unsigned short c_cflag; /* 控制模式标志*/
    unsigned short c_lflag; /*区域模式标志或本地模式标志或局部模式*/
    unsigned char c_line; /*行控制 line discipline */
    unsigned char c_cc[NCC]; /* 控制字符特性*/
};
```

二、作用

这个变量被用来提供一个健全的线路设置集 合，如果这个端口在被用户初始化前使用. 驱动初始化这个变量使用一个标准的数值集，它拷贝自 tty_std_termios 变量. tty_std_termos 在 tty 核心被定义为:

```
struct termios tty_std_termios = {
    .c_iflag = ICRNL | IXON,
    .c_oflag = OPOST | ONLCR,
    .c_cflag = B38400 | CS8 | CREAD | HUPCL,
    .c_lflag = ISIG | ICANON | ECHO | ECHOE | ECHOK | ECHOCTL | ECHOKE | IEXTEN,
```

```
    .c_cc = INIT_C_CC
};
```

这个 struct termios 结构用来持有所有的当前线路设置，给这个 tty 设备的一个特定端口. 这些线路设置控制当前波特率，数据大小，数据流控设置，以及许多其他值.

三、成员的值

（一）c_iflag 标志常量：Input mode（ 输入模式）

input mode 可以在输入值传给程序之前控制其处理的方式。其中输入值可能是由序列埠或键盘的终端驱动程序所接收到的字元。 我们可以利用 termios 结构的 c_iflag 的标志来加以控制，其定义的方式皆以 OR 来加以组合。 * IGNBRK ： 忽略输入中的 BREAK 状态。 （忽略命令行中的中断） * BRKINT ： （命令行出 现中断时，可产生一插断）如果设置了 GNBRK，将忽略 BREAK。如果没有设置，但是设置了 BRKINT，那么 BREAK 将使得输入和输出队列被刷新，如果终端是一个前台进程组的控制终端，这个进程组中所有进程将收到 SIGINT 信号。如果既未设置 IGNBRK 也未设置 BRKINT，BREAK 将视为与 NUL 字符同义，除非设置了 PARMRK，这种情况下它被视为序列;。
* IGNPAR ： 忽略桢错误和奇偶校验错。
* PARMRK ： 如果没有设置 IGNPAR，在有奇偶校验错或桢错误的字符前插入 377 。如果既没有设置 IGNPAR 也没有设置 PARMRK，将有奇偶校验错或桢错误的字符视为 。
* INPCK ： 启用输入奇偶检测。
* ISTRIP ： 去掉第八位。
* INLCR ：将输入中的 NL 翻译为 CR。（将收到 的换行符号转换为 Return）
* IGNCR ：忽略输入中的回车。
* ICRNL ：将输入中的回车翻译为新行 (除非设置了 IGNCR)(否则当输入信号有 CR 时不会终止输入)。
* IUCLC ：(不属于 POSIX) 将输入中的大写字母映射为小写字母。
* IXON ：启用输出的 XON/XOFF 流控制。
* IXANY ：(不属于 POSIX.1; XSI) 允许任何字符来重新开始输出。(?)
* IXOFF ：启用输入的 XON/XOFF 流控制。

* IMAXBEL:(不属于 POSIX) 当输入队列满时响零。Linux 没有实现这一位，总是将它视为已设置。

（二） c_oflag 标志常量：Output mode（ 输 出模式）

Output mode 主要负责控制输出字元的处理方式。输出字元在传送到序列埠或显示器之前是如何被程序来处理。 输出模式是利用 termios 结构的 c_oflag 的标志来加以控制，其定义的方式皆以 OR 来加以组合。 * OPOST ： 启用具体实现自行定义的输出处理。

* OLCUC ： (不属于 POSIX) 将输出中的小写字母映射为大写字母。
* ONLCR ： (XSI) 将输出中的新行符映射为回车-换行。
* OCRNL ： 将输出中的回车映射为新行符
* ONOCR ： 不在第 0 列输出回车。
* ONLRET ： 不输出回车。
* OFILL ： 发送填充字符作为延时，而不是使用定时来延时。
* OFDEL ： (不属于 POSIX) 填充字符是 ASCII DEL (0177)。如果不设置，填充字符则是 ASCII NUL。
* NLDLY ： 新行延时掩码。取值为 NL0 和 NL1。
* CRDLY ： 回车延时掩码。取值为 CR0, CR1, CR2, 或 CR3。
* TABDLY ： 水平跳格延时掩码。取值为 TAB0, TAB1, TAB2, TAB3(或 XTABS)。取值为 TAB3，即 XTABS，将扩展跳格为空格 (每个跳格符填充 8 个空格)。(?)
* BSDLY ： 回退延时掩码。取值为 BS0 或 BS1。(从来没有被实现过)
* VTDLY ： 竖直跳格延时掩码。取值为 VT0 或 VT1。
* FFDLY ： 进表延时掩码。取值为 FF0 或 FF1。

（三） c_cflag 标志常量：Control mode（ 控制模式）

Control mode 主要用于控制终端设备的硬件设置。利用 termios 结构的 c_cflag 的标志来加以控制。控制模式用在序列线连接到数据设备，也可以用在与终 端设备的交谈。 一般来说，改变终端设备的组 态要比使用 termios 的控制模式来改变行(lines)的行为来得容易。 * CBAUD ： (不属于 POSIX) 波特率掩码 (4+1 位)。

* CBAUDEX ： (不属于 POSIX) 扩展的波特率掩码 (1 位)，包含在 CBAUD 中。
* (POSIX 规定波特率存储在 termios 结构中，并未精确指定它的位置，而是提供了函数 cfgetispeed() 和 cfsetispeed() 来存取它。一些系统使用 c_cflag 中 CBAUD 选择的位，其他系统使用单独的变量，例如 sg_ispeed 和

sg_ospeed 。）

* CSIZE：字符长度掩码（传送或接收字元时用的位数）。 取值为 CS5（传送或接收字元时用 5bits），CS6，CS7，或 CS8。

* CSTOPB ：设置两个停止位，而不是一个。

* CREAD ：打开接受者。

* PARENB ：允许输出产生奇偶信息以及输入的奇偶校验（启用同位产生与侦测）。

* PARODD ：输入和输出是奇校验（使用奇同位而非偶同位）。

* HUPCL ：在最后一个进程关闭设备后，降低 modem 控制线 (挂断)。(?)

* CLOCAL ：忽略 modem 控制线。

* LOBLK :(不属于 POSIX) 从非当前 shell 层阻塞输出(用于 shl)。(?)

* CIBAUD :(不属于 POSIX) 输入速度的掩码。CIBAUD 各位的值与 CBAUD 各位相同，左移了 IBSHIFT 位。

* CRTSCTS :(不属于 POSIX) 启用 RTS/CTS (硬件) 流控制。

（四） c_lflag 标志常量：Local mode（ 局部模式）

Local mode 主要用来控制终端设备不同的特色。利用 termios 结构里的 c_lflag 的标志来设定局部模式。在巨集中有两个比较重要的标志： *ECHO：它可以让你阻止键入字元的回应。 *ICANON(正规模式)标志，它可以对所接收的字元在两种不同的终端设备模式之间来回切 换。 * ISIG：当接受到字符 INTR，QUIT，SUSP，或 DSUSP 时，产生相应的信号。

* ICANON：启用标准模式 (canonical mode)。允许使用特殊字符 EOF，EOL，EOL2，ERASE，KILL，LNEXT，REPRINT，STATUS，和 WERASE，以及按行的缓冲。

* XCASE:(不属于 POSIX; Linux 下不被支持) 如果同时设置了 ICANON，终端只有大写。输入被转换为小写，除了有前缀的字符。输出时，大写字符被前缀（某些系统指定的特定字符） ，小写字符被转换成大写。

* ECHO ：回显输入字符。

* ECHOE ：如果同时设置了 ICANON，字符 ERASE 擦除前一个输入字符，WERASE 擦除前一个词。

* ECHOK ：如果同时设置了 ICANON，字符 KILL 删除当前行。

* ECHONL ：如果同时设置了 ICANON，回显字符 NL，即使没有设置 ECHO。

* ECHOCTL ：(不属于 POSIX) 如果同时设置了 ECHO，除了 TAB,NL，START，和 STOP 之外的 ASCII 控制信号被回显为 ^X，这里 X 是比控制信号大 0x40

的 ASCII 码。例如，字符 0x08(BS) 被回显为 ^H。

* ECHOPRT ：(不属于 POSIX) 如果同时设置了 ICANON 和 IECHO，字符在删除的同时被打印。

* ECHOKE ：(不属于 POSIX) 如果同时设置了 ICANON，回显 KILL 时将删除一行中的每个字符，如同指定了 ECHOE 和 ECHOPRT 一样。

* DEFECHO :(不属于 POSIX) 只在一个进程读的时候回显。

* FLUSHO :(不属于 POSIX; Linux 下不被支持) 输出被刷新。这个标志可以通过键入字符 DISCARD 来开关。

* NOFLSH :禁止在产生 SIGINT, SIGQUIT 和 SIGSUSP 信号时刷新输入和输出队列，即关闭 queue 中的 flush。

* TOSTOP :向试图写控制终端的后台进程组发送 SIGTTOU 信号（传送欲写入的信息到后台 处理）。

* PENDIN :(不属于 POSIX; Linux 下不被支持) 在读入下一个字符时，输入队列中所有字符被重新输出。(bash 用它来处理 typeahead)

* IEXTEN :启用实现自定义的输入处理。这个标志必须与 ICANON 同时使用，才能解释特殊字符 EOL2，LNEXT，REPRINT 和 WERASE，IUCLC 标志才有效。

（五） c_cc 数组：特殊控制字元可提供使用者设定一些特殊的功能，如 Ctrl+C 的字元组合。

特殊控制字元主要是利用 termios 结构里 c_cc 的阵列成员 来做设定。c_cc 阵列主要用于正规与非正规两种环境，但要注意的是正规与非正规不可混为一谈。其定义了特殊的控制字符。符号下标（初始值）和意义为：

- VINTR：(003, ETX, Ctrl-C, or also 0177, DEL, rubout) 中断字符。发出 SIGINT 信号。当设置 ISIG 时可被识别，不再作为输入传递。
- VQUIT ：(034, FS, Ctrl-) 退出字符。发出 SIGQUIT 信号。当设置 ISIG 时可被识别，不再作为输入传递。
- VERASE ：(0177, DEL, rubout, or 010, BS, Ctrl-H, or also#) 删除字符。删除上一个还没有删掉的字符，但不删除上一个 EOF 或行首。当设置 ICANON 时可被识别，不再作为输入传递。
- VKILL ：(025, NAK, Ctrl-U, or Ctrl-X, or also @) 终止字符。删除自上一个 EOF 或行首以来的输入。当设置 ICANON 时可被识别，不再作为输入传递。
- VEOF ：(004, EOT, Ctrl-D) 文件尾字符。更精确地说，这个字符使得 tty 缓冲中的内容被送到等待输入的用户程序中，而不必等到 EOL。如果它是一行的第一个字符，那么用户程序的 read() 将返回 0，指示读到了 EOF。当设置 ICANON 时可被识别，不再作为输入传递。

- VMIN :非 canonical 模式读的最小字符数（MIN 主要是表示能满足 read 的最小字元数）。
- VEOL :(0, NUL) 附加的行尾字符。当设置 ICANON 时可被识别。
- VTIME : 非 canonical 模式读时的延时，以十分之一秒为单位。
- VEOL2 : (not in POSIX; 0, NUL) 另一个行尾字符。当设置 ICANON 时可被识别。
- VSWTCH : (not in POSIX; not supported under Linux; 0,NUL) 开关字符。(只为 shl 所用。)
- VSTART : (021, DC1, Ctrl-Q) 开始字符。重新开始被 Stop 字符中止的输出。当设置 IXON 时可被识别，不再作为输入传递。
- VSTOP : (023, DC3, Ctrl-S) 停止字符。停止输出，直到键入 Start 字符。当设置 IXON 时可被识别，不再作为输入传递。
- VSUSP : (032, SUB, Ctrl-Z) 挂起字符。发送 SIGTSTP 信号。当设置 ISIG 时可被识别，不再作为输入传递。
- VDSUSP : (not in POSIX; not supported under Linux; 031,EM, Ctrl-Y) 延时挂起信号。当用户程序读到这个字符时，发送 SIGTSTP 信号。当设置 IEXTEN 和 ISIG，并且系统支持作业管理时可被识别，不再作为输入传递。
- VLNEXT : (not in POSIX; 026, SYN, Ctrl-V) 字面上的下一个。引用下一个输入字符，取消它的任何特殊含义。当设置 IEXTEN 时可被识别，不再作为输入传递。
- VWERASE : (not in POSIX; 027, ETB, Ctrl-W) 删除词。当设置 ICANON 和 IEXTEN 时可被识别，不再作为输入传递。
- VREPRINT : (not in POSIX; 022, DC2, Ctrl-R) 重新输出未读的字符。当设置 ICANON 和 IEXTEN 时可被识别，不再作为输入传递。
- VDISCARD : (not in POSIX; not supported under Linux;017, SI, Ctrl-O) 开关：开始/结束丢弃未完成的输出。当设置 IEXTEN 时可被识别，不再作为输入传递。
- VSTATUS : (not in POSIX; not supported under Linux;status request: 024, DC4, Ctrl-T).
- 这些符号下标值是互不相同的，除了 VTIME，VMIN 的值可能分别与 VEOL，VEOF 相同。 (在 non-canonical 模式下，特殊字符的含义更改为延时含义。MIN 表示应当被读入的最小字符数。TIME 是以十分之一秒为单位的计时器。如果同时设置了它们，read 将等待直到至少读入一个字符，一旦读入 MIN 个字符或者 从上次读入字符开始经过了 TIME 时间就立即返回。如果只设置了 MIN，read 在读入 MIN 个字符之前不会返回。如果只设置了 TIME，read 将在至少读入一个字符，或者计时器超时的时候立即返回。如果都没有设置，read 将立即返回，只给出当前准备好的字符。)

MIN 与 TIME 组合有以下四种：

1. MIN = 0 , TIME =0 有 READ 立即回传否则传回 0 ,不读取任何字元

2. MIN = 0 ， TIME >0 READ 传回读到的字元,或在十分之一秒后传回 TIME 若来不及读到任何字元,则传回 0
3. MIN > 0 ， TIME =0 READ 会等待,直到 MIN 字元可读
4. MIN > 0 ， TIME > 0 每一格字元之间计时器即会被启动 READ 会在读到 MIN 字元,传回值或 TIME 的字元计时(1/10 秒)超过时将值 传回

四、 与此结构体相关的函数

（一）tcgetattr()

1.原型

```
int tcgetattr(int fd,struct termois & termios_p);
```

2.功能 取得终端介质（fd）初始值，并把其值 赋给 temios_p;函数可以从后台进程中调用；但是，终端属性可能被后来的前 台进程所改变。

（二）tcsetattr()

1.原型

```
int tcsetattr(int fd,int actions,const struct termios *termios_p);
```

2.功能 设置与终端相关的参数 (除非需要底层支持却无法满足)，使用 termios_p 引用的 termios 结构。optional_actions （tcsetattr 函数的第二个参数）指定了什么时候改变会起作用： * TCSANOW：改变立即发生
* TCSADRAIN：改变在所有写入 fd 的输出都被传输后生效。这个函数应当用于修改影响输出的参数时使用。(当前输出完成时将值改变)
* TCSAFLUSH ：改变在所有写入 fd 引用的对象的输出都被传输后生效，所有已接受但未读入的输入都在改变发生前丢弃(同 TCSADRAIN，但会舍弃当前所有值)。

（三）tcsendbreak()

传送连续的 0 值比特流，持续一段时间，如果终端使用异步串行数据传输的话。如果 duration 是 0，它至少传输 0.25 秒，不会超过 0.5 秒。如果 duration 非零，它发送的时间长度由实现定义。如果终端并非使用异步串行数据传输，tcsendbreak() 什么都不做。

（四）tcdrain()

等待直到所有写入 fd 引用的对象的输出都被传输。

（五）tcflush()

丢弃要写入引用的对象，但是尚未传输的数据，或者收到但是尚未读取的数据，取决于 queue_selector 的值：

- TCIFLUSH ： 刷新收到的数据但是不读
- TCOFLUSH ： 刷新写入的数据但是不传送
- TCIOFLUSH ： 同时刷新收到的数据但是不读，并且刷新写入的数据但是不传送

（六）tcflow()

挂起 fd 引用的对象上的数据传输或接收，取决于 action 的值：

- TCOOFF ： 挂起输出
- TCOON ： 重新开始被挂起的输出
- TCIOFF ： 发送一个 STOP 字符，停止终端设备向系统传送数据
- TCION ： 发送一个 START 字符，使终端设备向系统传输数据打开一个终端设备时的默认设置是输入和输出都没有挂起。
-

（七） 波特率函数

被用来获取和设置 termios 结构中，输入和输出波特率的值。新值不会马上生效，直到成功调用了 tcsetattr() 函数。设置速度为 B0 使得 modem "挂机"。与 B38400 相应的实际比特率可以用 setserial(8) 调整。 输入和输出波特率被保存于 termios 结构中。 cfmakeraw 设置终端属性如下：

```
termios_p->c_iflag &= ~(IGNBRK|BRKINT|PARMRK|ISTRIP|INLCR|IGNCR|ICRNL|IXON);
termios_p->c_oflag &= ~OPOST;
termios_p->c_lflag &= ~(ECHO|ECHONL|ICANON|ISIG|IEXTEN);
termios_p->c_cflag &= ~(CSIZE|PARENB);
termios_p->c_cflag |= CS8;
```

- cfgetospeed() 返回 termios_p 指向的 termios 结构中存储的输出波特率

- cfsetospeed() 设置 termios_p 指向的 termios 结构中存储的输出波特率为 speed。取值必须是以下常量之一：

```
B0      B50     B75     B110    B134    B150
B200    B300    B600    B1200   B1800
B2400   B4800   B9600   B19200  B38400
B57600  B115200 B230400 </span>
```

其中： 零值 B0 用来中断连接。如果指定了 B0，不应当再假定存在连接。通常，这样将断开连接。CBAUDEX 是一个掩码，指示高于 POSIX.1 定义的速度的那一些(57600 及以上)。因此，B57600 & CBAUDEX 为非零。

- cfgetispeed() 返回 termios 结构中存储的输入波特率。
- cfsetispeed() 设置 termios 结构中存储的输入波特率为 speed。如果输入波特率被设为 0，实际输入波特率将等于输出波特率。

五、 RETURN VALUE 返回值

- cfgetispeed() 返回 termios 结构中存储的输入波特率。
- cfgetospeed() 返回 termios 结构中存储的输出波特率。

其他函数返回： 0：成 功 -1:失 败，并且为 errno 置值来指示错误。注意 tcsetattr() 返回成功，如果任何所要求的修改可以实现的话。因此，当进行多重修改时，应当在这个函数之后再次调用 tcgetattr() 来检测是否所有修改都成功实现。

六、NOTES 注意

Unix V7 以及很多后来的系统有一个波特率的列表，在十四个值 B0, ..., B9600 之后可以看到两个常数 EXTA, EXTB ("External A" and "External B")。

很多系统将这个列表扩展为更高的波特率。tcsendbreak 中非零的 duration 有不同的效果。SunOS 指定中断 duration*N 秒，其中 N 至少为 0.25，不高于 0.5 。Linux, AIX, DU, Tru64 发送 duration 微秒的 break 。

FreeBSD, NetBSD, HP-UX 以及 MacOS 忽略 duration 的值。在 Solaris 和 Unixware 中， tcsendbreak 搭配非零的 duration 效果类似于 tcdrain。
SEE ALSO 参见 stty(1), setserial(8)