Linux ~ termios 串口编程 - Burden

Burden 关注 - 0 粉丝 - 6 +加关注

ermios 结构是在POSIX规范中定义的标准接口，它类似于系统V中的termio接口，通过设置termios类型的数据结构中的值和使用一小

组函数调用，你就可以对终端接口进行控制。

可以被调整来影响终端的值按照不同的模式被分为如下几组：

1.输入模式

2.输出模式

3.控制模式

4.本地模式

5.特殊控制模式

最小的termios结构的典型定义如下：

```
struct termios
{
    tcflag_t c_iflag;
    tcflag_t c_oflag;
    tcflag_t c_cflag;
    tcflag_t c_lflag;
    cc_t     c_cc[NCCS];
};
```

结构成员的名称与上面列出的5种参数类型相对应。

你可以调用函数tcgetattr来初始化一个终端对应的termios结构，该函数的原型如下：

```
#include<termios.h>
int tcgetattr(int fd, struct termios *termios_p);
```

这个函数调用把当前终端接口变量的值写入**termios_p**参数指向的结构。如果这些值其后被修改了，你可以通过调用函数**tcsetattr**来重新配置终端接口。

```
1
2
#include<termios.h>

int tcsetattr(int fd , int actions , const struct termios
```

```
*termios_h);
```

参数**actions**控制修改方式，共有三种修改方式，如下所示。

1.TCSANOW：立刻对值进行修改

2.TCSADRAIN：等当前的输出完成后再对值进行修改。

3.TCSAFLUSH：等当前的输出完成之后，再对值进行修改，但丢弃还未从read调用返回的当前的可用的任何输入。

接下来我们将分别对五种模式进行介绍。

一 输入模式

输入模式控制输入数据在传递给程序之前的处理方式。你通过设置termios结构中的c_iflag成员的标志对它们进行控制。所有的标志都被定义为

宏，并可通过按位或的方式结合起来。

可用于c_iflag成员的宏如下所示：

BRKINT：当在输入行中检测到一个终止状态时，产生一个中断。

TGNBRK：忽略输入行中的终止状态。

TCRNL：将接受到的回车符转换为新行符。

TGNCR：忽略接受到的新行符。

INLCR：将接受到的新行符转换为回车符。

IGNPAR：忽略奇偶校检错误的字符。

INPCK：对接收到的字符执行奇偶校检。

PARMRK：对奇偶校检错误作出标记。

ISTRIP：将所有接收的字符裁减为7比特。

IXOFF：对输入启用软件流控。

IXON：对输出启用软件流控。

如果BRKINT和TGNBRK标志都未被设置，则输入行中的终止状态就被读取为NULL（0X00）字符。

三.输出模式

输出模式控制输出字符的处理方式,即由程序发出的字符在传递到串行口或屏幕之前如何处理.通过设置c_oflag成员的标识对输出模式进行控制.

OPSOT:打开输出处理功能

ONLCR:将输出中的换行符转换为回车符

OCRNL:将回车符转换为换行符

ONOCR:第 0 行不输出回车符

ONLRET:不输出回车符

NLDLY:换行符延时选择

CRDLY:回车符延时

TABDLY:制表符延时

...

输出模式用得也不多

四.控制模式

控制模式控制终端的硬件特性,通过c_cflag成员标识配置.

CLOCAL:忽略所有调制解调器的状态行

CREAD:启用字符接收器

CS5/6/7/8:发送或接收字符时使用 5 / 6 / 7 / 8 比特

CSTOPB:每个字符使用两停止位

HUPCL:关闭时挂断调制解调器

PARENB:启用奇偶校验码的生成和检测功能

PARODD:只使用奇检验而不用偶校验

一般也不用这种方式,通常直接修改终端配置文件来修改硬件特性要容易一些

五.本地模式

通过c_lflag成员控制终端的某些特性

ECHO:启用输入字符的本地回显功能

ECHONL:回显换行符

ICANON:启用标准输入处理

ISIG:启用信号

...

最常用的是ECHO和ICANON标志,前者抑制键入字符的回显(抑制??),后者如
说明

六.特殊的控制字符

标准模式和非标准模式下,c_cc数组的下标有不同的值:

标准模式:

VEOF: E O F 字符

VEOL: E O L 字符

VERASE:ERASE字符

VINTR:INTR字符

VKILL:KILL字符

VQUIT:QUIT字符

VSTART:START字符

VSTOP:STOP字符

非标准模式:

VINTR:INTR字符

VMIN:MIN值

VQUIT:QUIT字符

VSUSP:SUSP字符

VTIME:TIME值

VSTART:START字符

VSTOP:STOP字符

1.字符

INTR:该字符使终端驱动程序向与终端相连的进程以送SIGINT信号

QUIT:该字符使终端驱动程序向与终端相连的进程发送SIGQUIT信号

EOF;该字符使终端驱动程序将输入行中的全部字符传递给正在读取输入的应用程序.如果输入行为空,read调用将返回０,就好像在文件尾调用read一样

...

2.TIME和**MIN**值

这两个值只用于非标准模式,两者结合共同控制对输入的读取方式,还能控制在一个程序试图与一个终端关联的文件描述符时将发生的情况

MIN = 0, TIME = 0时:read立即返回,如果有待处理的字符,它们就会被返回,如果没有,read调用返回0,且不读取任何字符

MIN = 0, TIME > 0时:有字符处理或经过TIME个0.1秒后返回

MIN > 0, TIME = 0时:read一直等待,直到有ＭＩＮ个字符可以读取,返回值是字符的数量.到达文件尾时返回0

MIN > 0, TIME > 0时:read调用时,它会等待接收一个字符.在接收到第一个字符及其后续的每个字符后,启用一个字符间隔定时器.当有ＭＩＮ个字符可读或两字符间的时间间隔超进TIME个0.1秒时,read返回

通过设置ＭＩＮ和ＴＩＭＥ值,我们可以逐个字符地对输入进行处理

3.通过**shell**访问终端模式

stty -a:这个命令用来查看当前终端的设置情况

stty sane:如果不小心设错了终端模式,可用这个命令恢复,另一种恢复办法是在设置之前保存当前stty设置,在需要时再读出

stty -g > save_stty:将当前设置保存到文件save_atty中

stty $(cat save_stty):读出save_atty文件,恢复原终端设置

第三种恢复的办法是重新打下一个终端模拟器.查看死掉的终端进程,kill掉它

4.在命令行模式下设置终端模式

比如想让shell脚本读取单个字符,就需要关闭标准模式,同时将ＭＩＮ设为１,ＴＩＭＥ设为0:

stty -icanon min1 time 0

另一个例子是关闭输入密码时的回显功能:

atty -echo

使用完这个命令后要执行atty echo,将回显功能再次恢复

5.终端速度

termios结构中没有关于终端速度的成员和标识,但我们可以通过一组函数来实现.注意输入和输出是分开的,应使用不同的函数

```
#include <termios.h>

speed_t cfgetispeed(const struct termios *);
speed_t cfgetospeed(const struct termios *);
int cfsetispeed(struct termios *, speed_t speed);
int cfseospeed(struct termios *, speed_t speed);
```

这些函数只作用于termios结构,因此需要先调用tcgetattr()获得termios结构,再调用以上函数之一设置终端速度,最后调用tcsetattr()使设置生效

上面的speed参数可设的值,其中比较重要的几个:

B0:挂起终端

B1200:1200波特

B2400:2400波特

B9600:9600波特

B19200:19200波特

B38400:38400波特

6.其他函数

这些函数直接作用于文件描述符,不需要读写termios结构:

```
1
2
3
#include <termios.h>

int tcdrain(int fd);让调用程序一直等待,直到所有排队的输出都发送完毕
```

4	int tcflow(int, int flowtype);暂停或重新开始输出 int tcflush(int fd, int in_out_selector);清空输入,输出或两者都清空