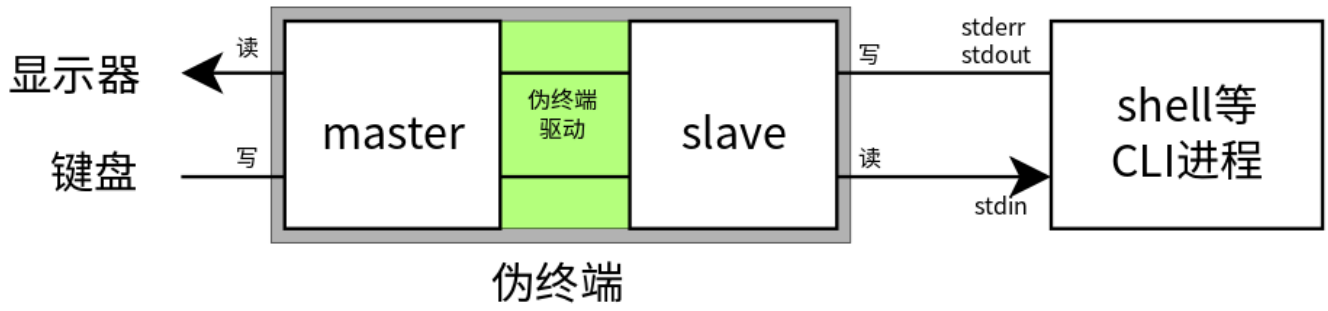
Linux 伪终端(pty)

通过《Linux 终端(TTY)》一文我们了解到：我们常说的终端分为终端 tty1-6 和伪终端。使用 tty1-6 的情况一般为 Linux 系统直接连了键盘和显示器，或者是使用了 vSphere console 等虚拟化方案，其它情况下使用的都是伪终端。本文将介绍伪终端的基本概念。本文中演示部分使用的环境为 ubuntu 18.04。

伪终端

伪终端(pseudo terminal，有时也被称为 pty)是指伪终端 master 和伪终端 slave 这一对字符设备。其中的 slave 对应 /dev/pts/ 目录下的一个文件，而 master 则在内存中标识为一个文件描述符(fd)。伪终端由终端模拟器提供，终端模拟器是一个运行在用户态的应用程序。

Master 端是更接近用户显示器、键盘的一端，slave 端是在虚拟终端上运行的 CLI(Command Line Interface，命令行接口)程序。Linux 的伪终端驱动程序，会把 master 端(如键盘)写入的数据转发给 slave 端供程序输入，把程序写入 slave 端的数据转发给 master 端供(显示器驱动等)读取。请参考下面的示意图(此图来自互联网)：



我们打开的终端桌面程序，比如 GNOME Terminal，其实是一种终端模拟软件。当终端模拟软件运行时，它通过打开 /dev/ptmx 文件创建了一个伪终端的 master 和 slave 对，并让 shell 运行在 slave 端。当用户在终端模拟软件中按下键盘按键时，它产生字节流并写入 master 中，shell 进程便可从 slave 中读取输入；shell 和它的子程序，将输出内容写入 slave 中，由终端模拟软件负责将字符打印到窗口中。

伪终端的使用场景

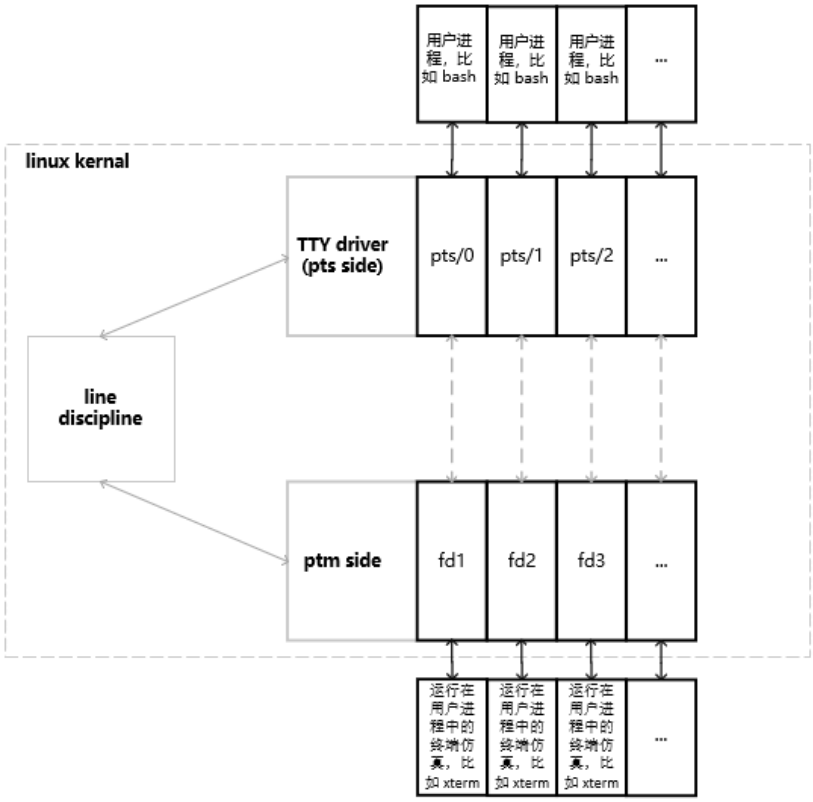
伪终端大概有三类使用场景：

- 像 xterm、gnome-terminal 等图形界面的终端模拟软件将键盘和鼠标事件转换为文本输入，并图形化地显示输出内容
- 远程 shell 应用程序(如 sshd)在客户机上的远程终端和服务器上的伪终端之间中继输入和输出
- 多路复用器应用,如 screen 和 tmux。它们把输入和输出从一个终端转播到另一个终端，使文本模式的应用程序从实际的终端上脱离

Linux 中为什么要提出伪终端这个概念呢？shell 等命令行程序不可以直接从显示器和键盘读取数据吗？
为了同屏运行多个终端模拟器、并实现远程登录，还真不能让 shell 直接跨过伪终端这一层。在操作系统的一大思想——虚拟化的指导下，为多个终端模拟器、远程用户分配多个虚拟的终端是有必要的。上图中的 shell 使用的 slave 端就是一个虚拟化的终端。Master 端是模拟用户一端的交互。之所以称为虚拟化的终端，是因为它除了转发数据流外，还要有点终端的样子。

伪终端原理

伪终端本质上是运行在用户态的终端模拟器创建的一对字符设备。其中的 slave 对应 /dev/pts/ 目录下的一个文件，而 master 则在内存中标识为一个文件描述符(fd)。对于伪终端来说，重点是软件仿真终端程序运行在用户空间，这是它与终端的本质区别，请参考下面的示意图：



/dev/ptmx 是一个字符设备文件，当进程打开 /dev/ptmx 文件时，进程会同时获得一个指向 pseudoterminal master(ptm)的文件描述符和一个在 /dev/pts 目录中创建的 pseudoterminal slave(pts) 设备。通过打开 /dev/ptmx 文件获得的每个文件描述符都是一个独立的 ptm，它有自己关联的 pts，ptmx(可以认为内存中有一个 ptmx 对象)在内部会维护该文件描述符和 pts 的对应关系，对这个文件描述符的读写都会被 ptmx 转发到对应的 pts。我们可以通过 lsof 命令查看 ptmx 打开的文件描述符：

```
$ sudo lsof /dev/ptmx
```

```
nick@esearch:~$ sudo lsof /dev/ptmx
COMMAND   PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
sshd    13117 root    5u   CHR    5,2      0t0   86 /dev/ptmx
sshd    13147 nick    9u   CHR    5,2      0t0   86 /dev/ptmx
sshd    13147 nick   11u   CHR    5,2      0t0   86 /dev/ptmx
sshd    13147 nick   12u   CHR    5,2      0t0   86 /dev/ptmx
```

进程默认的 IO

一般情况下我们通过远程连接的方式执行命令时，进程的标准输入、标准输出和标准错误输出都会绑定到伪终端上，下面是一个简单的 demo 程序：

```
#include <stdio.h>
#include <unistd.h>
int main()
{
    printf("PID : %d\n", getpid());
    sleep(200);

    printf("\n");
    return 0;
}
```

把这段代码保存在文件 mydemo.c 中，然后执行下面的命令编译并执行该程序：

```
$ gcc -Wall mydemo.c -o demo
$ ./demo
```

```
nick@esearch:~/demo$ ./demo
PID : 17981
```

demo 程序输出了自己进程的 PID，现在另外开一个终端执行 lsof 命令：

```
$ lsof -p 17981
```

```
nick@esearch:~$ lsof -p 17981
COMMAND   PID USER   FD   TYPE DEVICE SIZE/OFF    NODE NAME
demo    17981 nick  cwd    DIR  253,0     4096 4852063 /home/nick/demo
demo    17981 nick  rtd    DIR  253,0     4096       2 /
demo    17981 nick  txt    REG  253,0     8432 4851970 /home/nick/demo/demo
demo    17981 nick  mem    REG  253,0  2030544  139476 /lib/x86_64-linux-gn
demo    17981 nick  mem    REG  253,0   170960  139464 /lib/x86_64-linux-gn
demo    17981 nick    0u   CHR  136,0      0t0       3 /dev/pts/0
demo    17981 nick    1u   CHR  136,0      0t0       3 /dev/pts/0
demo    17981 nick    2u   CHR  136,0      0t0       3 /dev/pts/0
```

可以看到进程的 0u(标准输入)、1u(标准输出)、2u(标准错误输出)都绑定到了伪终端 /dev/pts/0 上。

参考：

Linux TTY/PTS概述
The TTY demystified
伪终端 pts man page
伪终端 pty man page