Linux 终端(TTY)

TTY 是 Teletype 或 Teletypewriter 的缩写，原来是指电传打字机，后来这种设备逐渐键盘和显示器取代。不管是电传打字机还是键盘显示器，都是作为计算机的终端设备存在的，所以 TTY 也泛指计算机的终端(terminal)设备。为了支持这些 TTY 设备，Linux 实现了一个叫做 TTY 的子系统。所以 TTY 既指终端，也指 Linux 的 TTY 子系统，当然 TTY 还有更丰富(混乱)的含义，本文试图把它们解释清楚。本文中演示部分使用的环境为 ubuntu 18.04。

硬件终端 terminal(TTY)

早期的终端(terminal) 是一台独立于计算机的机器(teletype 即, TTY)，大概像下面的样子：

它终端通过线缆与计算机连接，并完成计算机的输入输出功能：

现在物理终端实际上已经灭绝了，我们看到的所有 TTY 都是模拟视频终端，即软件仿真出来的终端。可以通过 toe -a 命令查看系统支持的终端类型，不要奇怪，这是一个挺长的列表。

控制台 console

提到终端就不能不提控制台 console。控制台的概念与终端含义非常相近，其实现在我们经常用它们表示相同的东西，但是在计算机的早期时代，它们确实是不同的东西。
一些数控设备(比如数控机床)的控制箱，通常会被称为控制台，顾名思义，控制台就是一个直接控制设备的面板，上面有很多控制按钮。在计算机里，把那套直接连接在电脑上的键盘和显示器就叫做控制台。而终端是通过串口连接上的，不是计算机自身的设备，而控制台是计算机本身就有的设备，一个计算机只有一个控制台。**计算机启动的时候，所有的信息都会显示到控制台上，而不会显示到终端上。**这同样说明，控制台是计算机的基本设备，而终端是附加设备。计算机操作系统中，与终端不相关的信息，比如内核消息，后台服务消息，都可以显示到控制台上，但不会显示到终端上。比如在启动和关闭 Linux 系统时，我们可以在控制台上看到很多的内核信息(下图来自 vSphere Client 中的 "Virtual Machine Console")：

```
[    3.990076] pci 0000:00:18.4:   bridge window [mem 0xea700000-0xea7fffff 64bi
t pref]
[    3.990268] pci 0000:00:18.5: PCI bridge to [bus 20]
[    3.990385] pci 0000:00:18.5:   bridge window [mem 0xfbe00000-0xfbefffff]
[    3.990521] pci 0000:00:18.5:   bridge window [mem 0xea300000-0xea3fffff 64bi
t pref]
[    3.990727] pci 0000:00:18.6: PCI bridge to [bus 21]
[    3.990844] pci 0000:00:18.6:   bridge window [mem 0xfba00000-0xfbafffff]
[    3.990965] pci 0000:00:18.6:   bridge window [mem 0xe9f00000-0xe9ffffff 64bi
t pref]
[    3.991158] pci 0000:00:18.7: PCI bridge to [bus 22]
[    3.991275] pci 0000:00:18.7:   bridge window [mem 0xfb600000-0xfb6fffff]
[    3.991394] pci 0000:00:18.7:   bridge window [mem 0xe9b00000-0xe9bfffff 64bi
t pref]
[    3.992097] NET: Registered protocol family 2
[    4.062349] TCP established hash table entries: 65536 (order: 7, 524288 bytes
)
[    4.085797] TCP bind hash table entries: 65536 (order: 8, 1048576 bytes)
[    4.087032] TCP: Hash tables configured (established 65536 bind 65536)
[    4.093080] UDP hash table entries: 4096 (order: 5, 131072 bytes)
[    4.100717] UDP-Lite hash table entries: 4096 (order: 5, 131072 bytes)
[    4.107506] NET: Registered protocol family 1
[    4.107630] pci 0000:00:00.0: Limiting direct PCI/PCI transfers
[    4.108194] Unpacking initramfs...
_
```

现在终端和控制台都由硬件概念，逐渐演化成了软件的概念。简单的说，**能直接显示系统消息的那个终端称为控制台，其他的则称为终端(控制台也是一个终端)。或者我们在平时的使用中压根就不区分 Linux 中的终端与控制台。**

下面的例子是通过 /dev/console 文件向控制台发送消息，这个这个例子我们可以看到控制台与终端的一点点不同之处。
打开 vSphere Client 中的 "Virtual Machine Console"(即控制台)，默认显示的是 tty1：

```
Ubuntu 18.04.1 LTS esearch tty1

esearch login:
```

通过其他的终端向 /dev/console 中写入字符串 "hello world"：
root@esearch:~# echo "hello world" > /dev/console

```
Ubuntu 18.04.1 LTS esearch tty1

esearch login: hello world
_
```

字符串显示在了控制台中。然后通过 Ctrl + Alt + F2 把控制台中的终端切换到 tty2，再次向 /dev/console 写入字符串：
root@esearch:~# echo "hello world" > /dev/console

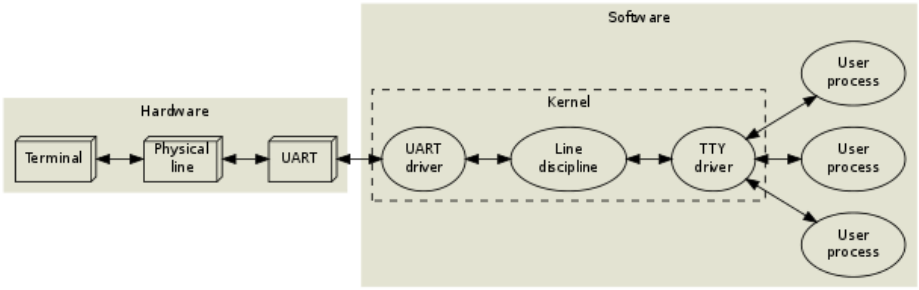
```
Ubuntu 18.04.1 LTS esearch tty2

esearch login: hello world
```

这次字符串写到了 tty2 中，这说明 Linux 总是把写入 /dev/console 的内容会显示在控制台中当前的虚拟终端(tty1-tty6)里。

TTY 设备

从历史上看，终端刚开始就是终端机，配有打印机，键盘，带有一个串口，通过串口传送数据到主机端，然后主机处理完交给终端打印出来。电传打字机(teletype)可以被看作是这类设备的统称，因此终端也被简称为 TTY(teletype 的缩写)。
如下图所示(下图来自互联网)：



UART 驱动

如上图所示，物理终端通过电缆连接到计算机上的 UART(通用异步接收器和发射器)。操作系统中有一个 UART 驱动程序用于管理字节的

物理传输。

行规范

上图中内核中的 Line discipline(行规范)用来提供一个编辑缓冲区和一些基本的编辑命令(退格，清除单个单词，清除行，重新打印)，主要用来支持用户在输入时的行为(比如输错了，需要退格)。

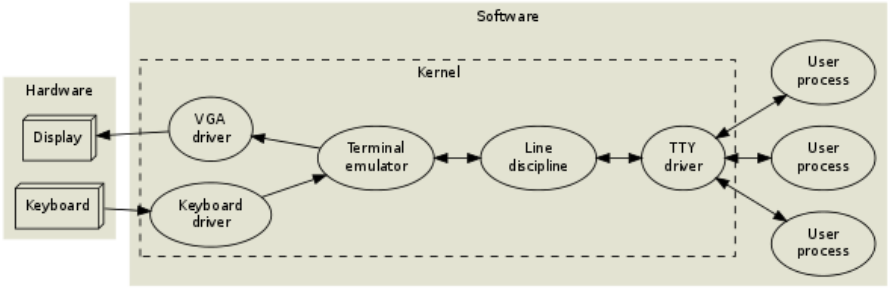
TTY 驱动

TTY 驱动用来进行会话管理，并且处理各种终端设备。

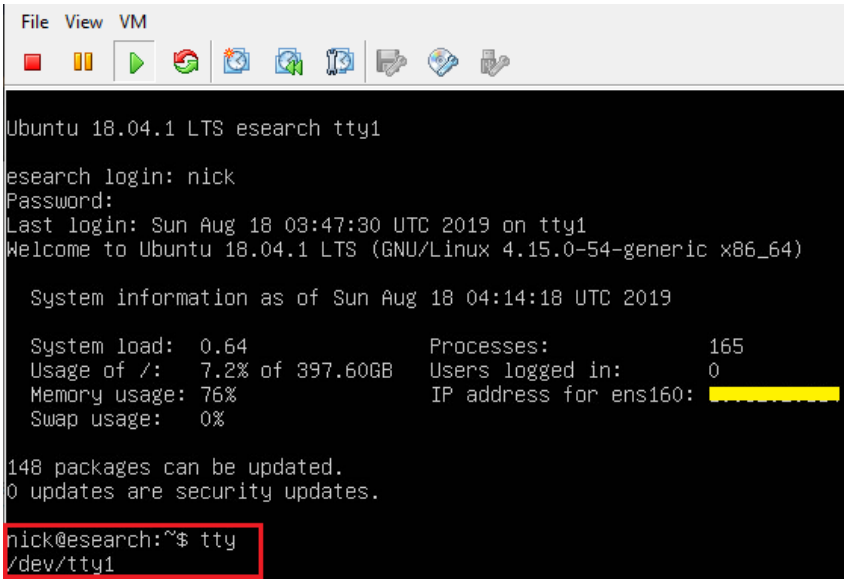
UART 驱动、行规范和 TTY 驱动都位于内核中，它们的一端是终端设备，另一端是用户进程。因为在 Linux 下所有的设备都是文件，所以它们三个加在一起被称为 "TTY 设备"，即我们常说的 TTY。

从软件仿真终端到伪终端

后来的终端慢慢演变成了键盘 + 显示器。如果我们要把内容输出到显示器，只要把这些内容写入到显示器对应的 TTY 设备就可以了，然后由 TTY 层负责匹配合适的驱动完成输出，这也是 Linux 控制台的工作原理(下图来自互联网)：

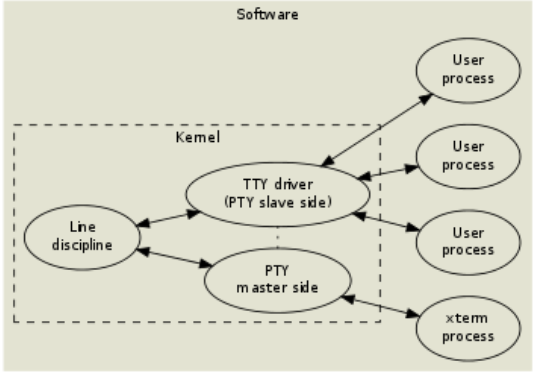


上图中，TTY 驱动和行规范的行为与前面的示例类似，但不再有 UART 或物理终端。相反，软件仿真出视频终端，并最终被渲染到 VGA 显示器。注意，这里出现了软件仿真终端，它们是运行在内核态的。显示器和 vSphere Client "Virtual Machine Console" 中的 tty1-tty6 都是**软件仿真终端**：

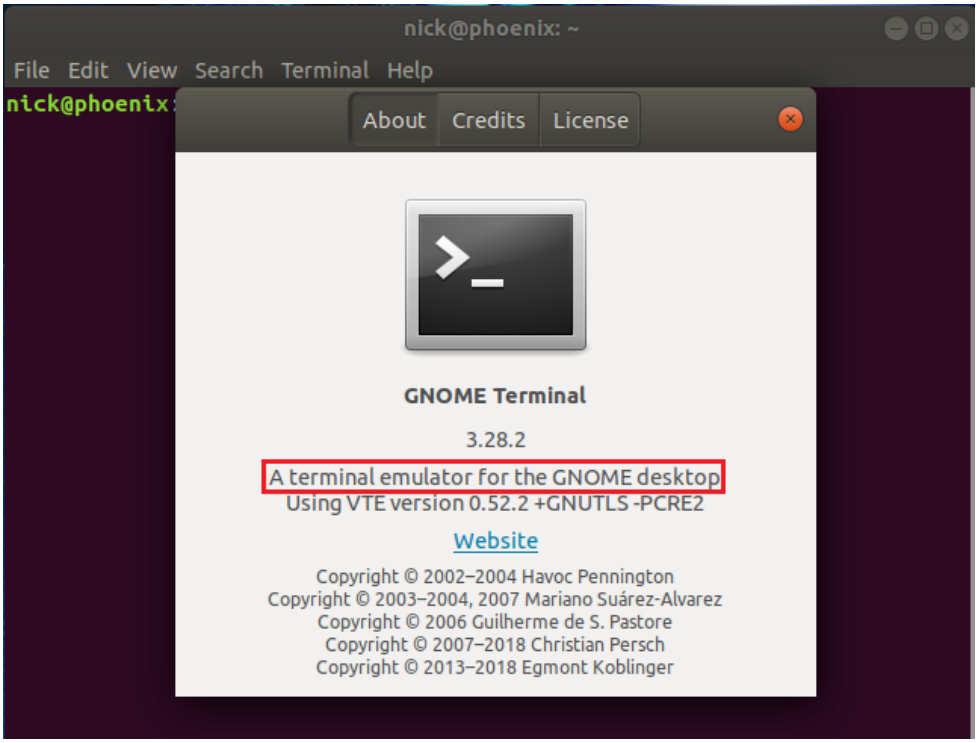


/dev/tty1-/dev/tty6 是这些仿真终端在文件系统中的表示，程序通过对这些文件的读写实现对仿真终端的读写。

如果我们在用户空间也进行终端仿真，情况会变得更加灵活，下图是 xterm 及其克隆的工作方式(下图来自互联网)：



为了便于将终端仿真移入用户空间，同时仍保持 TTY 子系统(TTY 子系统指 TTY 驱动和行规范)的完整，伪终端被发明了出来(pseudo terminal 或 pty)。伪终端在内核中分为两部分，分别是 master side 和 在 TTY 驱动中实现的 slave side。注意上图中的 xterm，这是一个运行在用户态的终端仿真程序，比如 Ubuntu Desktop 中的 GNOME Terminal：



当创建一个伪终端时，会在 /dev/pts 目录下创建一个设备文件：

```
nick@phoenix:~$ tty
/dev/pts/7
```

如果是通过 PuTTY 等终端仿真程序通过 SSH 的方式远程连接 Linux，那么终端仿真程序通过 SSH 与 PTY master side 交换数据。

终端与伪终端的区别

至此我们可以得出这样的结论：现在所说的终端已经不是硬件终端了，而是软件仿真终端(终端模拟软件)。
关于终端和伪终端，可以简单的理解如下：

- 真正的硬件终端基本上已经看不到了，现在所说的终端、伪终端都是软件仿真终端(即终端模拟软件)
- 一些连接了键盘和显示器的系统中，我们可以接触到运行在内核态的软件仿真终端(tty1-tty6)
- 通过 SSH 等方式建立的连接中使用的都是伪终端
- 伪终端是运行在用户态的软件仿真终端

总结

通过本文我们可以了解到，真正的硬件终端基本上已经看不到了。在一些连接了键盘和显示器的系统中(当然也包括一些 vsphere 等虚拟环境)，我们可以接触到运行在内核态的软件仿真终端。而我们使用最多的则是伪终端。

参考：

解密TTY
Linux TTY/PTS概述
The TTY demystified
What is stored in /dev/pts files and can we open them?
终端、虚拟终端、shell、控制台、tty的区别