Linux TTY framework(3)_从应用的角度看TTY设备

作者：wowo 发布于：2016-10-14 22:13 分类：TTY子系统

1. 前言

可以毫不夸张的说，我们在使用Linux系统的过程中，每时每刻都在和TTY打交道，显示输出、键盘输入、用户登录、shell终端、等等。

与此同时，作为软件工程师的我们，也会或多或少的困惑：这些习以为常的行为，怎么和kernel中的这些冷冰冰的代码联系起来的？

因此，在Linux TTY framework分析工作正式开始之前，让我们带着上面的疑问，以这些熟悉的应用场景为视角，进一步理解TTY有关的概念。这就是本文的目的。

2. 再谈终端（Terminal）设备

我们在“Linux TTY framework(1)_基本概念”中简单的提过，终端设备是指那些帮助我们和计算机进行人机交互的设备，所谓的人机交互，可简单的总结为：

> 输入（input），向计算机发送指令；
>
> 输出（output），计算机将执行结果显示出来。

与此同时，关于输入/输出，可提出如下疑问（为了简单，将输入、输出看作一个统一的整体，即终端）：

> 输入、输出设备是什么？
>
> 计算机怎么接收和输出？
>
> 人怎么接收和输出（这个就不用回答了，大家都知道，呵呵）。

关于这两个疑问，下面我们分别讨论。

2.1 终端的类型

根据不同的输入、输出设备类型，我们常见的终端有如下几类：

1）控制台终端（console）

> 这类终端的输入设备通常是键盘，输出设备通常是显示器；
>
> 输入、输出设备通过各类总线直接和计算机相连；
>
> “终端”其实是这些设备的一个逻辑抽象。

站内搜索

请输入关键词 搜索

功能

留言板
评论列表

最新评论

luke
因公司网络限制，文章里的图片不能显示，看的好辛苦啊！！
linuxer
@hit201j：已经修改，多谢！
bigpillow
Hi Linuxer, 我们这边有写一套完整的GPIO &...
hit201j
错别字：“还以一个就是bootloader传递过来的bl...
GrayMonkey
膜拜大佬，早找到你的文章就好了,Android开发一枚，一直...
fy
@wowo：是的。我只是就这里的“禁止抢占”非必要，问问wo...

文章分类

Linux内核分析(9)
统一设备模型(13)
电源管理子系统(42)
中断子系统(14)
进程管理(13)
内核同步机制(17)
GPIO子系统(5)
时间子系统(14)
通信类协议(7)
内存管理(20)
图形子系统(1)
文件系统(3)
TTY子系统(5)
u-boot分析(3)
Linux应用技巧(13)
软件开发(6)
基础技术(13)
蓝牙(16)
ARMv8A Arch(13)
显示(3)
基础学科(9)
技术漫谈(12)
项目专区(0)
X Project(28)

随机文章

2）虚拟终端（VT）

> 控制台终端的输出设备（显示器）一般只有一个，同一时刻由一个应用程序独占；
>
> 但在多任务的操作环境中，有时需要在将终端切换给另一个应用程序之前，保留当前应用在终端上的输出，以方便后面查看；
>
> 因此Unix/Linux系统在控制台终端的基础上，又虚拟出来6个终端----称作虚拟终端，不同的应用程序可以在这些虚拟终端上独立的输出，在需要的时候，可以通过键盘的组合键（CTRL+ALT+ F1~F6）将某一个虚拟终端调出来在屏幕上显示。

3）串口终端（TTY）

这是正牌的TTY设备：

> 输入设备和输出设备集成在一个独立的硬件上（称作TTY设备），这个硬件和计算机通过串口连接；
>
> 输入设备（键盘）的输入动作，将会转换为串口上的RX数据包（以计算机为视角），发送给计算机；
>
> 计算机的输出会以TX数据包的形式发送给TTY设备，TTY设备转换后在输出设备（屏幕）上显示。

4）软件终端

这是我们现在最常用的终端：

> 既然人机交互的数据流可以封装后经过串口传输，那么终端设备的形式就不再受限了，只要可以接收用户的输入并打包通过串口发送给计算机，以及接收计算机从串口发来的输出并显示出来，任何设备都可以变成终端设备，例如另一台计算机；
>
> 当另一台计算机被当作终端设备时，通常不会把它的所有资源都用来和对端进行人机交互，常用的方法是，在这个计算机上利用软件，模拟出来一个“终端设备”。该软件就像一个中间商：从键盘接收用户输入，然后控制串口发送给对端；从串口接收对端的输出，然后在软件界面上显示出来；
>
> 平时大家经常使用的PuTTY、SecureCRT、Windows超级终端、等等，都是“软件终端”。

5）USB、网络等终端

> 既然串口可以作为人机交互数据的传说媒介，其它通信接口一样可以，例如USB、Ethernet、等等，其原理和串口终端完全一样，这里不再过多说明。

6）图形终端

> 前面所介绍的那些终端，人机交互的输出界面都是字符界面，随着计算机技术的发展，GUI界面慢慢出现并成为主流，这些通过GUI交互的形式，也可以称作图形终端。不过这已经超出了TTY framework系列文章的讨论范围了，因为TTY的势力范围只涵盖字符界面。

2.2 输入和输出

首先声明，这里的输入和输出，都是针对计算机设备而言。

另外，计算机是一个硬件设备，它本身没有输入和输出的自主意识，因此，输入和输出，都是指运行在计算机中的软件，所谓的人机交互，其实是人和一个个的软件交互。

以Linux系统为例：

> kernel会通过控制台终端输出日志信息（printk）；

内存初始化代码分析（二）：内存布局
Linux kernel scatterlist API介绍
Linux PWM framework(1)_简介和API描述
Linux kernel的中断子系统之（六）：ARM中断处理过程
Linux时间子系统之（六）：POSIX timer

文章存档

2017年11月(1)
2017年10月(1)
2017年9月(5)
2017年8月(4)
2017年7月(4)
2017年6月(3)
2017年5月(3)
2017年4月(1)
2017年3月(8)
2017年2月(6)
2017年1月(5)
2016年12月(6)
2016年11月(11)
2016年10月(9)
2016年9月(6)
2016年8月(9)
2016年7月(5)
2016年6月(8)
2016年5月(8)
2016年4月(7)
2016年3月(5)
2016年2月(5)
2016年1月(6)
2015年12月(6)
2015年11月(9)
2015年10月(9)
2015年9月(4)
2015年8月(3)
2015年7月(7)
2015年6月(3)
2015年5月(6)
2015年4月(9)
2015年3月(9)
2015年2月(6)
2015年1月(6)
2014年12月(17)
2014年11月(8)
2014年10月(9)
2014年9月(7)
2014年8月(12)
2014年7月(6)
2014年6月(6)
2014年5月(9)
2014年4月(9)
2014年3月(7)
2014年2月(3)
2014年1月(4)

> kernel启动之后，init进程会打开控制台终端，以便和我们交互（接收一些指令并应答、输出日志信息、等等）；
>
> 随后，getty应用可以打开任意的终端设备，和我们交互，以便我们可以登入系统；
>
> 登入系统之后，getty将终端的控制权交给shell（bash、sh等等），我们与之交互，进行着愉快的人机对话，巴拉巴拉……
>
> 当然，在人机对话的过程中，我们可以命令shell启动其它的应用，例如地图应用，该应用可能会打开另一个TTY设备（例如GPS UART），并与之交互（由此可见，上面提到的人机交互其实是狭隘的概念，也可以机机交互~~）；
>
> 等等，等等。

3. 软件视角

讲完概念，我们还是要回到软件上来，厘清Linux TTY framework中的那些奇怪概念。

注1：建议读者结合“Linux TTY framework框架[1]”阅读本节内容。

3.1 console driver

Linux系统中可以存在多个种类各异的终端设备，工程师会使用一个个的TTY driver（struct tty_driver）驱动它们。如果某些我们需要让某些终端作为控制台终端，可以基于TTY driver，创建对应的console driver，并注册给kernel。

关于console driver，kernel有如下的策略：

1）可以同时注册多个console driver，并有选择的使用。

2）可以在kernel启动的时候，通过命令行（或者后来的device tree），告诉kernel使用哪个或者哪些控制台终端。例如：

只有这种方式？

> console=”/dev/ttyS0”, console=”/dev/ttyUSB0”

3）对kernel的日志输出来说，可以在所有被选中的控制台终端上输出。

4）后续可用作人机交互的控制台终端只能有一个（后指定的那个）。

3.2 控制台终端（/dev/console）

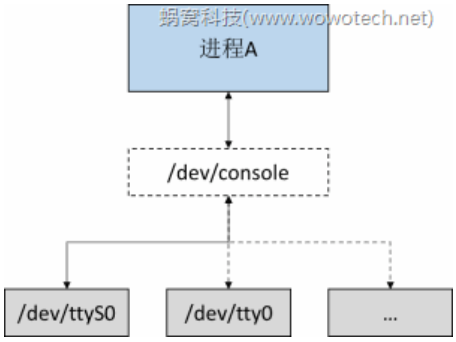
前面提到过，控制台终端只能有一个（最后指定的那个），那么“/dev/console”又是怎么回事呢？

了解linux kernel启动过程的同学都知道，kernel启动的后期，会在kernel_init线程（最后会退化为init进程）中打开控制台终端。但是由上面3.1小节的介绍可知，控制台终端的类型、名称是五花八门的，怎么让kernel的核心代码无视这些差异呢？这就是“/dev/console”的存在意义：

由“Linux TTY framework(2)_软件架构[2]”中的介绍可知，/dev/console的设备号固定为(5, 1)，当init线程打开该设备的时候，TTY core会问system console core：喂，哪一个终端适合做控制台终端啊？

也是就说，在init线程打开设备前，就有tty设备被注册为console driver并在命令行指定了具体的tty设备名称，如：console=”/dev/ttyS0”,console=”/dev/ttyUSB0”

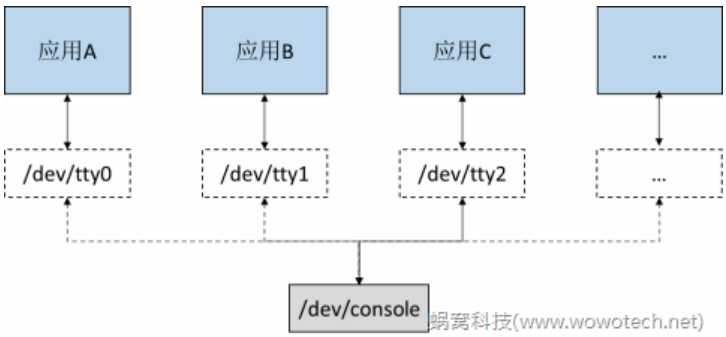
因此，最终打开的是那个具体的、可以当作控制台终端的设备，而“/dev/console”，仅仅是一个占位坑，如下图所示：



3.3 虚拟终端（/dev/ttyN）

正如其名，虚拟终端是虚拟出来的一个终端，不对应具体的设备（屏幕和键盘）。应用程序可以打开某一个虚拟终端，以便和人进行交互。

对应用程序而言，这个终端和具体的物理终端，没有任何区别（应用程序也无法区分）。而对整个系统来说，由于物理资源（键盘和屏幕）只有一套，因此同一时刻只能和某一个虚拟终端对接。从另一个角度看，各个虚拟终端轮流使用物理资源和人进行交互，如下所示：



3.4 伪终端(Pseudo Terminal，pty)

前面提到过，既然串口（serial）可用作终端和计算机之间的数据传输通道，那么其它诸如Ethernet的通信介质，也可以实现类似的功能。但这里面有一个问题（以网络为例）：

> Linux系统中的网络驱动，并不是以TTY的形式提供API（众所周知，网络使用socket接口）。因此，系统中的应用程序，无法直接打开网络设备，进而和对应的终端设备通信。怎么办？

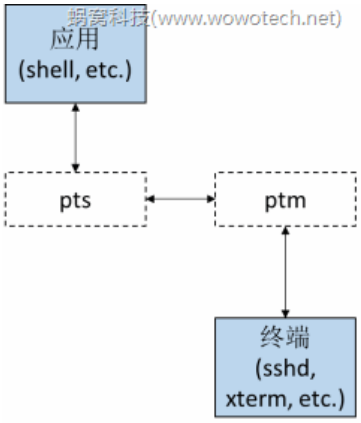
解决方案就是伪终端（英文为Pseudo Terminal，简称pty）。字面上理解，伪终端根本不是终端（虚拟终端好歹还是），是为了让应用程序可以从网络等非串口类的接口上，和终端设备交互而伪造出来的一种东西。它的原理如下：

> pty由pts(pseudo-terminal slave)和ptm(pseudo-terminal master)两部分组成。
>
> pts伪造出一个标准的TTY设备，应用程序可以直接访问。应用程序向pts写入的数据，会直接反映到ptm上，同样，应用程序从pts读数据，则相当于直接从ptm读取。
>
> 而pym，则根据具体情况，具体实现。例如：要通过网络接口和终端设备交互，则pym需要打开对应的socket，将pts写来的数据，从socket送出，将从socket读取的数据，送回给pts。

有了伪终端，Unix/Linux系统中的终端设备就可以脱离具体的物理限制，可以是任何形态，例如在GUI环境中，使用Terminal软件（如xterm）模拟出来的终端，其原理为：

> 用户启动xterm等Terminal软件的时候，该软件会生成一对pts和ptm，同时执行shell应用；
>
> Terminal软件会将pts交给shell应用，shell应用傻傻分不清，还以为它是一个标准的TTY设备呢，欢快的运行了；
>
> shell从pts读入数据，相当于从ptm读入；shell向pts输出数据，相当于向ptm写入；这是由TTY framework自动完成的，不需要Terminal软件关心；
>
> 在另一端，Terminal软件将会从ptm读取数据（实际上是shell通过pts写入的数据），并显示在自己的GUI窗口上；于此同时，Terminal软件会从自己的GUI窗口接收用户输入，并通过pts转给shell，太完美了。

上面提到了有关伪终端的两个例子，其示意图图下：



3.5 控制终端（control terminal，/dev/tty）

讲到控制终端（control terminal），就不得不提Unix/Linux的Job control[3]功能。有关job control，感兴趣的读者可以参考其它文章（如[3]），这里简单总结如下：

> 1）job control是Unix/Linux shell（记住，shell是一个应用程序）中的概念，是shell用来管理、控制jobs的一种方法。
>
> 2）job是进程组（process group[4]）在shell中的体现。换句话说，shell借用进程组实现了job。
>
> 3）进程组（或者job，在本文中可以等同，后面不在区分）是多个进程的组合。

基于上面简单的知识（不一定准确，但不影响我们对控制终端的理解），我们引入控制终端的概念：

> 1）通常情况下，Linux启动后，终端（以后都用TTY指代）的控制权会交给shell（一种应用程序）。所谓的控制权，就是指shell程序可以通过TTY读取终端的输入，以及通过TTY向终端输出。
>
> 2）通过shell，可以启动其它的应用程序，相应地，应用程序在需要的时候也会获得TTY的控制权。
>
> 3）同一时刻，只能有一个应用可以占有TTY，即只有一个应用可以通过TTY输入、输出。
>
> 4）那个占有TTY、可以进行输入输出的应用，称作前台应用。相应的，不能进行输入输出的应用，称作后台应用。因此，shell中只有一个前台应用，可以有多个后台应用。
>
> 5）然后，问题就来了：如果某个后台应用，就是想输入输出，怎么办？有一个办法，就是通过控制终端（control terminal）。
>
> 6）控制终端在Linux中的名称固定为/dev/tty, 设备号为(5, 0)，作用和/dev/console类似，进程可以通过TTY core提供的ioctl，选择控制终端所对应的实际的终端设备。
>
> 7）暂且抛开前台应用不谈（因为人家有TTY设备），对于那些后台应用，如果想输入输出，可以读取或者写入控制终端。此时，一般情况下，TTY core会向后台应用发送SIGTTIN(读取控制终端时) 或者SIGTTOU（写入控制终端时）信号，这会终止该后台应用。
>
> 8）不过，shell会重设收到 SIGTTOU信号时的行为，于是，后台应用写入的内容，可以通过控制终端显示出来。

也就是说，控制台终端出了通过kernel命令行和设备树的方式指定具体的物理终端外，还可以通过ioctl指定吗？

控制终端和前台应用的tty设备可以是同一个吗？

4. 参考文档

[1] Linux TTY framework框架

[2] Linux TTY framework(2)_软件架构

[3] https://en.wikipedia.org/wiki/Job_control_(Unix)

[4] http://www.wowotech.net/process_management/process_identification.html

标签: Linux tty console vt 虚拟终端 pty 伪终端 控制终端

« DRAM 原理 5 ：DRAM Devices Organization | 内存初始化（上）»

评论：

江南书生
2016-10-20 14:16

你好，感觉文章写的真不错。不过，看到"终端"的时候还有一些疑问，虚拟终端（6个）、跟ubuntu上面的ctrl+alt+T调出来的终端、通过远程登录软件登录的终端，这三者有什么关系呢？

回复

wowo
2016-10-20 16:54

@江南书生：多谢夸奖。
虚拟终端（6个），ubuntu上面的ctrl+alt+T调出来的终端，它们两个是一样的；
通过远程登录软件登录的终端，一般是伪终端（可参考 3.4 伪终端）。

回复

江南书生
2016-10-20 16:56

@wowo：差不多懂了，谢谢

回复

electrlife
2016-10-17 16:51

@wowo:
1， 对应用程序写的不多，从来没有关注过所谓的控制terminal，如果需要放在后台运行，也简单的使用下面类似的shell command:
./application &

对于后台的应用或者说是daemon设计时需要特别的注意吗（在control terminal方面）？

2， 对于/dev/tty，我的理解是每个进程读取这个设备应该会得到不同的值，即这个设备文件简单理解为banked类型。不知是否正确？应用程序的stdin,stdout, stderr一般情况下应该就是该应用程序的control terminal吗？

3， 在传统的unix中，子进程的control terminal应该就来自其parents的control terminal，应该确实是一个实实在在的terminal。但现在的application的control terminal似乎一般使用pts做为其control terminal，这样理解是事正确？

回复

wowo
2016-10-17 21:36

@electrlife：你的问题都很好啊，可惜我不敢回答，关于控制终端、shell、job control等概念，我也是知之甚少。以后写这一块的时候，一定用你这三个问题作为引子。

回复

electrlife
2016-10-18 15:12

@wowo：@wowo: 谢谢！期待着你的大作！

回复

linuxer
2016-10-18 19:12

@electrlife：要理解这些问题需要理解session、进程组、controlling terminal的概念，不是三言两语可以讲清楚的，哈哈

回复

electrlife
2016-10-19 11:39

@linuxer：其实对于这几个概念，困扰了我很久，不知Linuxer什么时候有时间来讲讲！这几个概念似乎和tty也有很大的联系！

回复

linuxer
2016-10-19 15:24

@electrlife：如果你愿意的话，可以去讨论区提交问题，我们可以在那里讨论一下，我相信wowo同学后续会描述清楚这些内容的。

回复

electrlife
2016-10-17 16:30

@wowo: 再次拜读了下结合网上的信息，觉得这篇文章应该是在无GUI的情况下去描述的，或者说是老的传统的unix语境下去理解。因为，我们总是会和现在使用的系统上的terminal做比较，觉得不好理解。而我们现在桌面的的terminal应该是一个仿真的terminal不以属于这里描述的任何一个。传统的Unix terminal应该就是一种char设备，没有图形桌面， 在这种情况下VT或许是有意义的，现在的桌面情况下，个人觉得VT似乎失去了它本身的意义，不再需要了。

>> 8) 不过，shell会重设收到 SIGTTOU信号时的行为，于是，后台应用写入的内容，可以通过控制终端显示出来。对于这个，请教下，shell如何重设应用程序的SIGTTOUT信号

回复

wowo
2016-10-17 16:40

@electrlife：terminal和我们常用的putty等远程登录，所用的技术都一样，就是伪终端（pty）。
对一个技术，没有“是否有意义”的概念，服务器是linux天下，服务器很少有GUI，VT依旧会存在。
至于shell和SIGTTOU信号，我还没有开始去看，我也不是很清楚它的实现，你可以看看代码、找找资料，然后和我们分享一下。

回复

狂奔的蜗牛
2016-10-17 16:11

Hi，如果有时间的话可不可以介绍一下系统崩溃死掉后，为什么通过网口登录的终端看不到内核panic信息，而串口是可以的呢？是因为串口是作为标准输入输出设备导致的么

回复

wowo
2016-10-17 16:21

@狂奔的蜗牛：因为网口登录所用的pts终端，只是作为普通的终端，而不是控制台终端（命令行console=xxx指示的才是）。printk只会输出到控制台终端。

回复

狂奔的蜗牛
2016-10-17 17:28

@wowo：TKS,嗯呢，应该是这样设计，printk只管数据到控制台中断，如果还判断有多少个登录的终端，每个都发送，那就不好玩了

回复

electrlife
2016-10-17 10:45

Hi wowo， 看完之后还是有点晕:-)，按我的理解，我们平时接触最多的其实应该是GUI终端，比如ubuntu下的terminal应用，在此应用中我们可以打开多个tab，也即多个shell。我个人觉得从这种我们最常用的场景深入下去，应该更容易理解。

回复

wowo
2016-10-17 13:49

@electrlife：谢谢提醒，我会再修改把这部分补充上。
因为这一系列文章是从x project的任务4引出的，所以我最初的关注点是串口终端。
而且基于linux驱动工程师的本能（我很少使用GUI），我总觉得串口终端才是TTY的正统，所以就忽略了广大的伪终端场景。
非常感谢:-)

回复

wowo
2016-10-17 22:02

@electrlife：“3.4 伪终端(Pseudo Terminal，pty)”中增加了对Terminal软件的粗略介绍，请参考。后面的文章会跟着这个思路展开。

回复

发表评论：

昵称

邮件地址 (选填)

个人主页 (选填)

发表评论