一：TTY----- 终端设备统称；

**终端是一种字符型设备，通常使用tty简称各种类型的终端。**tty设备包括串口、终端、伪终端三大类

**linux的终端类型：**

**/dev/ttySn,串行口终端**

**/dev/pty,伪终端**

**/dev/tty,当前进程的控制终端，可以是介绍的其它任何一种终端**

**/dev/ttyn,tty1~tty6是虚拟终端，tty0当前虚拟终端的别名。**

**/dev/console，控制台终端（显示器）**

**二：/dev/console是系统控制台，是与操作系统交互的设备，系统所产生的信息会发送到该设备上**

**控制台：供内核使用的终端为控制台。控制台在Linux启动时，通过命令  
console=…指定，如果没有指定控制台，系统把第一个注册的终端(tty)作为控制台。 如printk打印到哪里哪里就是控制台，如lcd屏幕，终端软件的那个等**

**1. 控制台是一个虚拟的终端，它必须映射到真正的终端上。**

**2. 控制台可以简单的理解为printk输出的地方。**

**3. 控制台是个只输出的设备，功能很简单，只能在内核中访问。**

1. **终端（tty）为主机提供了人机接口，每个人都通过终端使用主机的资源。终端有字符终端和图形终端两种。一台主机可以连很多终端。**
2. **控制台（console）是一种特殊的人机接口, 是人控制主机的第一人机接口。而主机对于控制台的信任度高于其他终端。**

**某些情况下console和tty0是一致的，都是一种人机交互接口。**

**如果一个终端设备要实现console功能，必须向内核注册一个struct console结构，一般的串口驱动中都会有。如果设备要实现tty功能，必须要内核的tty子系统注册一个struct tty\_driver结构，注册函数在drivers/tty/tty\_io.c中。一个设备可以同时实现console和tty\_driver，一般串口都这么做。**

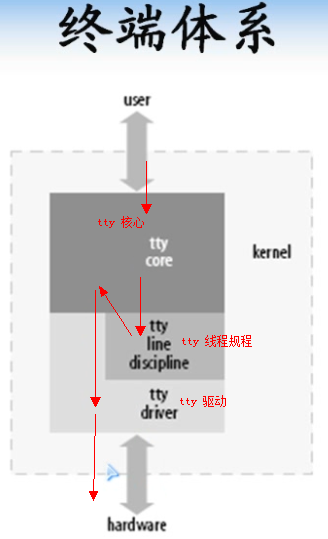
**串口终端（/dev/ttySn）**

 串行端口终端（Serial Port Terminal）是使用计算机串行端口连接的终端设备。计算机把每个串行端口都看作是一个字符设备。有段时间串行端口设备通常被称为终端设备，那时它的最大用途就是用来连接终端，所以这些串行端口所对应的设备名称是/dev/tts/0（或/dev/ttyS0）、/dev/tts/1（或/dev /ttyS1）等，设备号分别是（4,0）、（4,1）等（对应于win系统下的COM1、COM2等）。若要向一个端口发送数据，可以在命令行上把标准输出重定向到这些特殊文件名上即可。

       例如，在命令行提示符下键入：echo tekkaman> /dev/ttyS1会把“tekkaman”发送到连接在ttyS1（COM2）端口的设备上。

 serial core是构建在tty core之上的。注册一个串口驱动即在tty core层注册一个tty驱动

串口驱动中用到的两个最重要的数据机构 struct uart\_driver 表示一个serial驱动，struct uart\_port 表示一个串口端口。



TTY体系分为：TTY核心，TTY线路规程，TTY驱动3部分

**第一层：tty\_core**

所有tty类型的驱动的顶层构架，向应用曾提供了统一的接口，应用层的read/write等调用首先会到达这里。此层由内核实现，代码主要分布在drivers/char目录下的n\_tty.c,tty\_io.c等文件中

**第二层：线路规程**

不同的tty类型的设备，具有不同的线路规程。这一层也由内核实现，主要代码在drivers/char.tty\_ldisc.c文件中从tty\_read/tty\_write函数可以看出，他们最后调用到了线路规程的read/write函数

**第三层：**

具体的tty类型的驱动，由我们实现比如，以下是摘自serial\_core.c的一段代码，描述的是串口驱动：

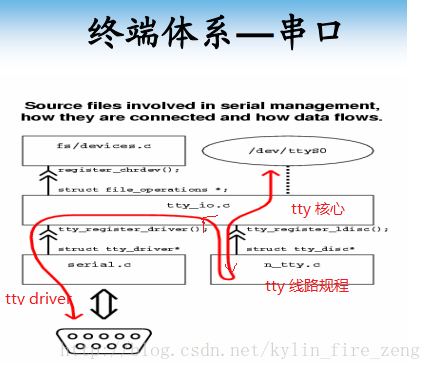
写数据流程：

TTY核心从用户获取要发送给TTY设备的数据，然后把数据传递给TTY线路规程, 它对数据进行处理后，负责把数据传递到TTY驱动程序，TTY驱动程序负责格式化数据，并通过硬件发送出去

读数据流程：

从硬件收到的数据向上通过TTY驱动, 进入TTY线路规程, 再进入TTY核心, 最后被用户获取

TTY驱动可以直接和TTY核心通讯, 但是通常TTY线路规程会修改在两者之间传送的数据。TTY驱动不能直接和线路规程通信，甚至不知道它的存在，线路规程的工作是格式化从用户或者硬件收到的数据. 这种格式化常常实现为一个协议, 如PPP或Bluetooth



uart\_driver描述串口驱动，它包含串口设备的驱动名、设备名、设备号等信息。

struct uart\_driver

{

  struct module \*owner;

  const char \*driver\_name;  //驱动名

  const char \*dev\_name;   //设备名

  int major;   //主设备号

  int minor;   //起始次设备号

  int nr;       //设备数

  struct console \*cons;

  struct uart\_state \*state;

  struct tty\_driver \*tty\_driver;

}

Linux为串口驱动注册提供了如下接口：

int uart\_register\_driver(struct uart\_driver \*drv)

端口描述：

uart\_port用于描述一个UART端口（一个串口）的地址、FIFO大小、端口类型等信息

一个uart\_port实例对应一个串口设备

struct uart\_port

{

  spinlock\_t lock; /\* 端口锁\*

  unsigned int iobase; /\* IO端口基地址\*/

  unsigned char \_\_iomem \*membase; /\* IO内存基地址\*/

  unsigned int irq; /\* 中断号\*/

  unsigned char fifosize; /\* 传输fifo大小\*/

  const structuart\_ops \*ops;  
  …………………………………………………………  
}

终端串口驱动大体流程：

