# tty串口驱动程序设计

## tty驱动程序构架

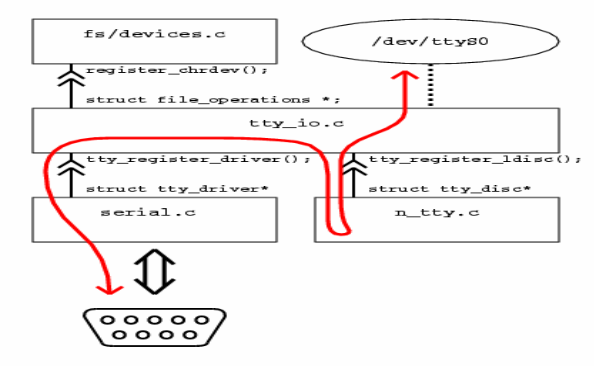
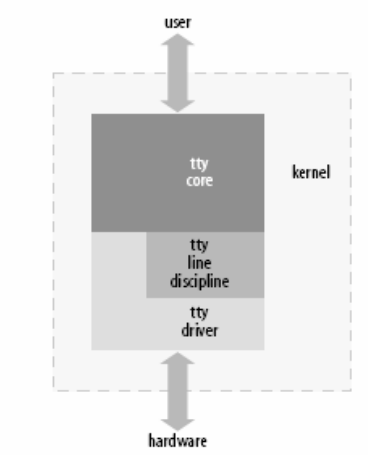
在Linux系统中，终端是一类字符型设备，它包括多种类型，通常使用tty来简称各种类型的终端设备。

1. 串口中断（/dev/ttyS\*）
2. 控制台终端（/dev/console）
3. 虚拟终端(/dev/tty\*)

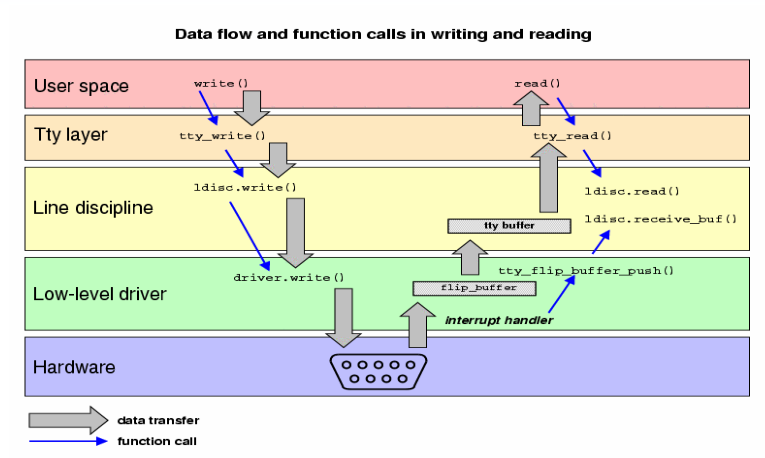
**控制台**

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。

**终端体系**



在Linux中，tty体系分为：TTY核心，tty线路规程，tty驱动3部分。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，他对数据进行处理后，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。



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线程规划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。

## 驱动设计

读操作

tty驱动从硬件收到数据后，负责把数据传递到tty核心，tty核心将从tty驱动收到的数据缓存到一个tty\_flip\_buffer类型的结构中。该结构包含两个数据数组。从TTY设备接设备接收到的数据被存储于第一个数组，当这个数组满,等待数据的用户将被通知。当用户从这个数组读数据时数据时,任何从TTY驱动驱动新来的数据将被存储在第2个数组。当第二个数组存满后，数据再次提交给用户,并且驱动又开始填充第1个数组，以此交替。

驱动描述

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;

};

驱动注册

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

端口描述

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 struct uart\_ops \*ops;

…… …… …… …… …… …… …… …… …… …… ……

}

操作串口

struct uart\_ops

{

unsigned int(\*tx\_empty)(struct uart\_port\*);

void(\*set\_mctrl)(struct uart\_port \*, unsigned int mctrl);

unsigned int(\*get\_mctrl)(struct uart\_port\*);

void(\*stop\_tx)(struct uart\_port\*); //停止发送

void(\*start\_tx)(struct uart\_port\*); //开始发送

void(\*send\_xchar)(struct uart\_port \*, char ch); //发送xchar

void(\*stop\_rx)(struct uart\_port\*); //停止接收

。。。。。。 。。。。。。 。。。。。。

}

添加端口

int uart\_add\_one\_port(struct uart\_driver \*drv, struct uart\_port \*port)

操作流程

1. 定义一个usrt\_druver的变量，并初始化；
2. 使用uart\_register\_driver来注册这个驱动；
3. 初始化uart\_port和ops
4. 调用uart\_add\_one\_port()添加初始化好的uart\_port

## 串口的打开与发送

串口打开

1. 使能串口接收功能：rx\_enabled
2. 为数据接收注册中断处理程序
3. 使能串口发送功能tx\_enabled
4. 为数据发送注册中断处理程序

串口发送数据

1. 判断x\_char是否为0，如果不为0，则发送x\_char。
2. 如果发送缓冲区或驱动为停止发送，取消发送。
3. 循环发送，循环条件：发送缓冲区不为空
4. 如果发送缓冲区的剩余数据量<256，则唤醒之前阻塞的发送进程。
5. 如果发送缓冲为空，则关闭发送使能。