1. 打印printk 自制proc
2. 自制工具
3. 根据内核答应段错误分析
4. 修改内核
5. 我们在uboot打印出来如下，我们要设置bootargs=console=ttySAC0不设置的话板子启动的话不会打印东西。就像是卡在那里。

例如在将uboot的参数打印出来。uboot为了启动内核而且向内核传参。

设置：set bootargs 不写任何的参数，启动。试试效果，不会有任何内核东西打印出来

uboot 设置参数：只有设置了内核的信息才能打印出来。具体可以看韦东山开发手册

bootargs=console=ttySAC0,115200root=/dev/nfs nfsroot=192.168.0.105:/work/nfs\_root/first\_fs ip=192.168.0.11:192.168.0.105:192.168.0.1:255.255.255.0::eth0:off init=/linuxrc

1.uboot 设置参数 console= ttySAC0

2内核用printk打印 。肯定要发到具体的硬件上去。所以最终肯定要调用硬件处理函数。

假如打印到串口 或者是lcd 是由uboot参数决定的。

内核根据命令行参数来，例如找到bootargs=console=ttySAC0对应的硬件操作函数。

内核根据这个console=找：

会在printk.c中找到；

\_\_setup("console=", console\_setup);

然后内核处理参数console=时就调用console\_setup()来处理这个前缀后面内容。

在函数中调用这个：

add\_preferred\_console(name, idx, options);//添加喜欢的控制台，记录下来

串口打印：肯定有个结构体：

.name = ttysac0

.write=

Lcd打印:

.name = ttysac0

.write= 硬件的处理函数输出

register\_console(struct console \*console)：会将包含.name .write= 的结构体注册

而且注册的时候会与，\_\_setup传进来的名字比较。

例如在：

s3c2410.c D:\WeiDongShan\_Learn\内核配置裁剪以及启动流程linux-2.6.22.6\drivers\serial

register\_console(&s3c24xx\_serial\_console);//会有如下的一个结构体

static struct console s3c24xx\_serial\_console =

{ // #define S3C24XX\_SERIAL\_NAME "ttySAC"

.name = S3C24XX\_SERIAL\_NAME,

.device = uart\_console\_device,

.flags = CON\_PRINTBUFFER,

.index = -1,

.write = s3c24xx\_serial\_console\_write,

.setup = s3c24xx\_serial\_console\_setup

};

我们用printk的时候：

printk:

vprintk：

/\* Emit the output into the temporary buffer \*/

//会将输出的信息放到一个临时的buffer

printed\_len = vscnprintf(printk\_buf, sizeof(printk\_buf), fmt, args);

// Copy the output into log\_buf. If the caller didn't provide

//appropriate log level tags, we insert them here

//把临时buffer的数据稍作处理然后放到 log\_buf.

例如：printk("abc") printk(KERN\_ERR "abc")

#define KERN\_ERR **"<3>"**// KERN\_ERR 表示"<3>"

所以也就是：

printk("<3>""abc") 🡪 printk("<3>abc")//“”“”两个这样的双引号会自动成为一对

printk(“abc”) 会得到printk(“<4>abc”)//如果不加打印等级的话默认时4。具体的打印级别话可以看韦东山的那个写的书

可以用dmesg命令将log\_buf的数据打印出来重现内核的输出信息.

然后肯定会调用.write输出。

release\_console\_sem();

call\_console\_drivers(\_con\_start, \_log\_end);//可以同时打印在串口或者lcd从他是drivers就可以看出，也就是在设置uboot参数的时候：

console=ttySAC0 console=tty1等

\_call\_console\_drivers(start\_print, cur\_index, msg\_level);//从log\_buf得到一个级别。

\_\_call\_console\_drivers()//够级别的话才能打印

con->write(con, &LOG\_BUF(start), end - start);

//write函数就是在注册的时候写的

所以整个过程就是：

硬件驱动程序入口函数，注册register\_console()里面由.name .write 函数.

uboot传进来的参数与name比较。比较成功后

printk将打印的信息稍作调整处理放在缓冲区，然后将嗲用硬件处理函数将消息打印出去。当然里面要有打印级别的判断。