**问：挂载自己制作的文件系统卡在这里：**

NET: Registered protocol family 1  
NET: Registered protocol family 17  
VFS: Mounted root (cramfs filesystem) readonly.  
Freeing init memory: 116K  
Failed to execute /linuxrc.  Attempting defaults...  
Kernel panic - not syncing: No init found.  Try passing init= optionto kernel.

bootargs为：noitinrd root=/dev/mtdblock2 rootfstype=cramfs console=ttySAC0,115200 init=/linurc mem=64M  
  
linuxrc的内容如下：

#!/bin/sh  
echo mount /etc as ramfs

/bin/mount-n -t ramfs ramfs /etc  
/bin/cp  -a  /mnt/etc/\*  /etc  
echo re-create the /etc/mtab entries

#re-create the /etc/mtab entries  
/bin/mount -f -t cramfs -o remount, ro /dev/mtdblock/3  /  
/bin/mount -f -t ramfs ramfs /etc  
exec /sbin/init

rcS如下：

#!/bin/sh  
/bin/mount  -a

**内核版本2.6.19.2。怎么解决？**

**答**：

这类问题很常见，先总体介绍一下解决思路。

能出现让人激动的的控制台，那么系统移植已经接近完成。 但是不少人在最后一步出现问题。  
要点如下：  
1. 在正确的位置烧写正确格式的文件系统映象  
2. 内核支持这种文件系统格式  
3. 文件系统的内容要完整  
  
上面说得简单，一个个介绍。  
1. 在正确的位置烧写正确的文件系统映象：  
**(a). 正确的位置**  
嵌入式开发中常通过bootloader烧写文件系统映象，假设写在flash的地址A处。  
内核启动时，显然要从地址A处读取文件系统，内核怎么知道的呢？通过命令行参数，比如“root=/dev/mtdblock2 ”。/dev/mtdblock2 又是怎么和地址A对应上的呢？内核将flash划分为几个分区，这是在代码中固定的。/dev/mtdblock2是第3个分区，它的开始地址必须是A。

内核启动时，可以看到这些分区的开始地址、结束地址，比如内核启动时会有类似下面的信息：

Creating 3 MTD partitions on NAND 64MiB 3,3V 8-bit:  
0x00000000-0x00030000 : bootloader

0x00050000-0x00250000: kernel

0x00250000-0x03ffc000: root

对于上面的内核信息，/dev/mtdblock2对应root分区，开始地址为0x00250000，使用bootloader写文件系统映象时，烧写的地址必须是0x00250000。  
  
所以，**要保证3点**：① bootloader烧到地址A，② 地址A是内核某个分区的开始地址，③ 命令行参数“root=/dev/mtdblockXXX ”是这个分区。  
  
**(b). 正确格式的文件系统映象**  
不同的bootloader支持烧写的文件系统映象格式不同、使用的烧写命令也可能不同，请注意这点。另外，马大哈们制作文件系统映象时，使用的工具也不要弄错了。最后，请保证这个文件系统映象是“真的烧写了”，**因为如果flash只是擦除而没有烧写，它也是“正确的**、可以挂接的文件系统”──有人碰到这个问题，我和他答非所问地折腾了很久。  
  
2. 内核支持这种文件系统格式  
配置内核时选上支持的文件系统格式  
  
1、2这两个问题如果不能保证，内核启动时会出现类似如下错误：

VFS: Cannot open root device mtdblock2 or unknown-block(2,0)  
Please append a correct root= boot option

如果1、2能保证，就可以挂接上文件系统，出现类似下面的字样时，革命已经成功了80％：

VFS: Mounted root (cramfs filesystem) readonly.

Freeing init memory: 116K

3. 文件系统的内容要完整

挂接文件系统后，内核就会读取、执行文件系统中的某个文件，通过它来启动应用程序。这个文件要么通过命令行参数“init=xxxx”指定，要么取默认的文件(下面说明)。

一般制作文件系统映象时，都是在一个目录（假设目录名为rootfs）下放好各种东西：bin/，sbin/，lib/，etc/fstab等文件，然后将这个目录制作为文件系统映象。

可以想象，如果这个目录中的东西不对、不全，即使制作出了文件系统映象，也只是能识别出来，挂接上去。但是启动不了──所谓启动，不就是执行文件系统中的程序嘛？

这时会有类似以下的错误：

Failed to execute /linuxrc.  Attempting defaults...

Kernel panic - not syncing: No init found.  Try passing init= optionto kernel

它说得很明显， Failed to execute /linuxrc ──执行/linuxrc失败：

它为什么要执行/linuxrc，还不是因为你在命令行中加入了“init=/linuxrc”这个参数。

它为什么会失败？原因有二：

**一、制作文件系统映象时，rootfs目录下有linuxrc文件吗？**

**二、rootfs目录的linuxrc文件是正确的吗？**

请好好确定这两点，大多数是没有linuxrc文件──linuxrc是busybox自动生成，只要配置好就可以。如果有linuxrc，还出现问题，原因就是它无法执行了(解决方法在下面)。

不用linuxrc行不行？当然行！看看内核文件init/main.c，有如下字样：

    run\_init\_process(/sbin/init);

    run\_init\_process(/etc/init);

    run\_init\_process(/bin/init);

    run\_init\_process(/bin/sh);

    panic(No init found.  Try passing init= option tokernel);

就是说，它会依次尝试执行/sbin/init、/etc/init、/bin/init、/bin/sh这些文件，都失败后才打印出错信息No init found.  Try passing init= option to kernel…

所以，出现这个错误信息，表明没有或是无法执行这些文件：需通过命令行参数“init=xxxx”来指定的xxx文件、/sbin/init、/etc/init、/bin/init、/bin/sh。

**解决方法：**

**一、请检查你的rootfs目录，看看这些文件是否存在**

**二、使用file命令看看它们是什么文件类型，是否可执行。**

使用busybox时，linuxrc是/bin/busybox文件的软链接，那就看看busybox的文件类型，可以使用下面的命令：

$ file linuxrc

linuxrc: symbolic link to `bin/busybox'

$ file bin/busybox

bin/busybox: ELF 32-bit LSB executable, ARM, version 1, for GNU/Linux 2.4.3,dynamically linked (uses shared libs), stripped

**注意了**：如果bin/busybox 是一个动态链接的文件，还要把它用到的库复制到rootfs中。这些库在交叉编译工具的相应目录下。

最后一点，文件系统中的各种配置文件、dev目录也要正确

回到这个帖子，它的内核打印信息为：

VFS: Mounted root (cramfs filesystem) readonly.

Freeing init memory: 116K

Failed to execute /linuxrc.  Attempting defaults...

Kernel panic - not syncing: No init found.  Try passing init= optionto kernel

说明文件系统挂接成功(VFS: Mounted root (cramfs filesystem)readonly.)；

还说明/linuxrc不存在或者不可执行（Failed toexecute /linuxrc.  Attempting defaults...）；

但是楼主的意思是linuxrc已经有了，内容为：

#!/bin/sh

echo mount /etc as ramfs

/bin/mount -n -t ramfs ramfs /etc

/bin/cp -a /mnt/etc/\* /etc

echo re-create the /etc/mtab entries

#re-create the /etc/mtab entries

/bin/mount -f -t cramfs -o remount,ro /dev/mtdblock/3 /

/bin/mount -f -t ramfs ramfs /etc

exec /sbin/init

它是一个脚本，它的执行依赖于/bin/sh，问题转为：/bin/sh是否存在？是否可以执行？

可以用file命令看看它的类型、是否需要动态库。