使用BusyBox制作linux根文件系统（CramFS+mdev）

操作系统：Ubuntu9.04

内核版本：linux-2.6.24.7

开发板：博创arm2410s

交叉编译工具：arm-linux-gcc-4.1.1

BusyBox：busybox-1.11.1

CramFS：CramFS-1.1

注意：由于要制作的根文件系统使用了mdev(BusyBox简化版的udev），因此，需要内 核支持sysfs、procfs和ramfs（tmpfs）文件系统以及hotplug（uevent）事件机制。

浅蓝色为命令，橘红色为 代码和脚步

1、准备根文件系统

使用shell脚本create_rootfs.sh，建立根文件系统的目录框架

lingd@ubuntu:~/arm2410s$ vi create_rootfs.sh

create_rootfs.sh 内容如下：

```
#!/bin/sh
echo "------Create rootfs directons....."
mkdir rootfs
cd rootfs
echo "--------Create root,dev....."
mkdir root dev etc bin sbin mnt sys proc lib home tmp var usr
mkdir usr/sbin usr/bin usr/lib usr/modules
mkdir mnt/usb mnt/nfs mnt/etc mnt/etc/init.d
mkdir lib/modules
chmod 1777 tmp
cd ..
```

echo "-------make direction done---------"

为 create_rootfs.sh添加执行权限

lingd@ubuntu:~/arm2410s$ chmod a+x create_rootfs.sh

执行create_rootfs.sh，建立根文件系统的目录框架

lingd@ubuntu:~/arm2410s$ ./create_rootfs.sh

lingd@ubuntu:~/arm2410s$ cd rootfs

lingd@ubuntu:~/arm2410s/rootfs$ ls

bin dev etc home lib mnt proc root sbin sys tmp usr var

2、创建设备 文件

mdev 是通过 init 进程来启动的，在使用 mdev 构造 /dev 目录之前，init 至少要用到设备文件/dev/console、/dev/null ，所以需要事先建立这两个设备文件:

lingd@ubuntu:~/arm2410s/rootfs$ cd dev

lingd@ubuntu:~/arm2410s/rootfs/dev$ sudo mknod -m 660 console c 204 64

[sudo] password for lingd:

lingd@ubuntu:~/arm2410s/rootfs/dev$ sudo mknod -m 660 null c 1 3

lingd@ubuntu:~/arm2410s/rootfs/dev$ ls -l

total 0

crw-rw---- 1 root root 5, 1 2010-04-02 15:49 console

crw-rw---- 1 root root 1, 3 2010-04-02 15:50 null

注意：本来console的主次设备号应为5和1.但是因为init在执行完/etc/init.d/rcS脚本后，就 会在一个控制台上，按照inittab的指示开一个shell（或者是开getty+login，这样用户就会看到提示输入用户名的提示符）。在mdev -s未执行之前，/dev目录下只有我们创建的/dev/null和/dev/console，也就是说，没有控制台可供init用来按照 inittab 的指示开一个shell。而在s3c24xx 系列芯片的串口驱动里面用了s3c2410_serial做为设备名（在内核源码的“drivers/serial/s3c2410.c”文件的949

行），因此，启动时可用s3c2410_serial0、s3c2410_serial1或s3c2410_serial2作为init用来按照 inittab的指示开一个shell的控制台，这里我用了串口0，即s3c2410_serial0（主次设备号为204和64）作为控制台。这里直接 将console当s3c2410_serial0，所以console的主次设备号才会为204和64。

3、准备一些配置文件和系统启动时所需的文件

3.1、mdev配置文件mdev.conf

mdev 会在/etc目录下找mdev的配置文件: mdev.conf. 如果该文件不存在，那么在执行mdev –s这个命令时，会提示找不到mdev.conf。我们不需要mdev规则，所以只是touch生成一个空文件就OK了。当然也可以根据mdev的规则来 编写mdev.conf。我把所有配置文件都是在/mnt/etc下，而不是/etc，后面解释这么做的原因。

```
lingd@ubuntu:~/arm2410s/rootfs/dev$ cd ../mnt/etc
lingd@ubuntu:~/arm2410s/rootfs/mnt/etc$ touch mdev.conf
lingd@ubuntu:~/arm2410s/rootfs/mnt/etc$ ls
init.d  mdev.conf
```

3.2、linuxrc

linuxrc 位于根文件系统的顶层目录/，这里为rootfs。/Linuxrc 执行init 进程初始化文件。主要工作是把已安装根文件系统中的/etc 安装为ramfs，并拷贝/mnt/etc/目录下所有文件到/etc，这里存放系统启动后的许多特殊文件；接着Linuxrc 重新构建文件分配表inittab；之后执行系统初始化进程/sbin/init。

```
lingd@ubuntu:~/arm2410s/rootfs/mnt/etc$ cd ../..
lingd@ubuntu:~/arm2410s/rootfs$ vi linuxrc
```

linuxrc 内容如下：

```
#!/bin/sh
echo "Processing /linuxrc"

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"
/bin/mount -f -t cramfs -o remount,ro /dev/bon/3 /
/bin/mount -f -t ramfs ramfs /etc

echo "start init"
exec /sbin/init
```

```
lingd@ubuntu:~/arm2410s/rootfs$ ls
bin  dev  etc  home  lib  linuxrc  mnt  proc  root  sbin  sys  tmp  usr  var
```

Linuxrc 脚本分析

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

这句话的作用加载一个ramfs作为/etc目录。这样 /etc就是一个可写目录。

从这个脚本可知，你的根文件系统是一个cramfs（只读文件系统），而/etc作为系统运行配置文件的存放地点，可 能会写一些运行状态在这里，linuxrc第一件事情就是将一个ramfs mount到/etc只读目录中，使得/etc/目录可写，指定参数 -n的目的是告诉mount不要写/etc/mtab, 这个文件存放当前系统已挂载（mount）的文件系统清单。因为现在/etc/目录还是只读，所以这次mount不要写这个文件，否则会失败。而且 mount上后，原/etc会覆盖掉（原/etc下的文件都不见了，umount后会重新出现），所以我们把配置文件都保存在/mnt /etc，mount上ramfs到/etc后，再把配置文件拷贝到/etc。而不是直接将配置文件保存在/etc/下

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

/etc成为可写目录后，将所有/mnt/etc中的配置文件拷贝到/etc/中，这说明ramfs可能是一个空的ramfs，没有配置文件，或者配置文件比较老。同时也说明这个系统是一个只读系统，每次系统运行中写入的配置不会保留。

将以前 mount的那些信息重新写到/etc/mtab中，命令就是下面这些。

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

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

这些命令只是将这 些mount信息写到/etc/mtab中，不会实际去mount这些block device，说明你的根文件系统依然是以前的那个/dev/bon/3

exec /sbin/init

执行根文件系统中的init执行 程序，使其成为1号进程。shell正式运行。

3.3、rcS

rcS 文件位于/etc/init.d,是busybox版init第一个运行的脚步（常见的init还有Sys V init 版，其第一个执行的脚步是/etc/rc.d/rc.sysinit)。/mnt/etc/init.d/rcS 完成各个文件系统的 mount，再执行/mnt/etc/rc.local；通过rcS 可以调用 ifconfig 程序配置网络。rcS 执行完了以后，init 就会在一个 console 上，按照 inittab 的指示开一个 shell，或者是开 getty + login，这样用户就会看到提示输入用户名的提示符。/etc/init.d/rcS 文件内容如下：

lingd@ubuntu:~/arm2410s/rootfs$ cd mnt/etc/init.d

lingd@ubuntu:~/arm2410s/rootfs/mnt/etc/init.d$ vi rcS

/etc/init.d/rcS内容如下：

#!/bin/sh

echo "Processing /etc/init.d/rcS"

echo "mount -a"

mount -a #mount上fstab文件中所有文件系统

exec /etc/rc.local

3.4、/etc/rc.local

/etc/rc.local是被init.d/rcS 文件调用执行的特殊文件，与Linux 系统硬件平台相关，如安装核心模块、进行网络配置、运行应用程序、启动图形界面等。内容如下：

```
lingd@ubuntu:~/arm2410s/rootfs/mnt/etc/init.d$ cd ..
lingd@ubuntu:~/arm2410s/rootfs/mnt/etc$ vi rc.local
/etc/rc.local内容如下：
#!/bin/sh
echo "Processing /etc/rc.local"

echo "get hostname"
/bin/hostname -F /etc/hostname

echo "Starting mdev"
echo /sbin/mdev > /proc/sys/kernel/hotplug
mdev -s

echo "ifconfig eth0 192.168.1.21"
ifconfig eth0 192.168.1.21

echo "************************************************"
echo "*                                              *"
echo "*        lingd  rootfs for linux 2.6.24.7      *"
echo "*                                              *"
echo "*         arm-linux-gcc version 4.1.1          *"
echo "*                                              *"
echo "*               2010-03-30                     *"
echo "*                                              *"
echo "************************************************"


lingd@ubuntu:~/arm2410s/rootfs/mnt/etc$ ls
init.d  rc.local
```

在rc.local使用了"/bin/hostname -F /etc/hostname"来设置主机名(设置主机名主要是为了后面设置命令提示符PS1)。这条命令需要了一个主机名配置文件/etc/hostname，其内容如下：

arm2410s

3.5、/etc/profile

rc.local 首先执行该文件配置应用程序需要的环境变量等。

lingd@ubuntu:~/arm2410s/rootfs/mnt/etc$ vi profile

/etc/profile内容如下：

```
#/etc/profile
echo "Processing /etc/profile"
echo "set user path"
PATH=/bin:/sbin:/usr/bin:/usr/sbin

echo "set search library path"
LD_LIBRARY_PATH=/lib:/usr/lib

echo "set PS1"
HOSTNAME=`/bin/hostname`
PS1='\u@\h:\w\$ ' # 设置命令提示符为ubuntu风格
export PATH LD_LIBRARY_PATH HOSTNAME PS1
```

```
lingd@ubuntu:~/arm2410s/rootfs/mnt/etc$ ls
init.d  profile  rc.local
```

改变这四个文件的权限

```
lingd@ubuntu:~/arm2410s/rootfs/mnt/etc$ cd ../..
lingd@ubuntu:~/arm2410s/rootfs$ chmod 775 linuxrc mnt/etc/init.d/rcS
```

mnt/etc/rc.local mnt/etc/profile

3.6、/etc/inittab

内核引导完成后，内核会启动初始化进程init（用户级进程）来进行 系统的各项配置。init是系统第一个进程，它是系统上运行的所有其他进程的父进程，他会观察其子进程，并在需要时启动、停止、重启它们。init主要是 用来完成系统的各项配置。init从/etc/inittab获取所有信息。想了解BusyBox init及其inittab基本原理的，可以看这篇文章http://blog.chinaunix.net/u3/109117/showart_2208026.html。

lingd@ubuntu:~/arm2410s/rootfs$ cd mnt/etc

lingd@ubuntu:~/arm2410s/rootfs/mnt/etc$ vi inittab

/etc /inittab内容如下：

```
#/etc/inittab
::sysinit:/etc/init.d/rcS
console::askfirst:-/bin/sh
::ctrlaltdel:/sbin/reboot
::shutdown:/bin/umount -a -r
```

"console::askfirst:-/bin/sh"中 的-表示的是让busybox开启一个登录(login)shell,login shell在执行前都会读取配置文件/etc/profile和.profile。由BusyBox源码的shell/ash.c文件可知 这一点：

```
int ash_main(int argc ATTRIBUTE_UNUSED, char **argv)
{
    ……
    if (argv[0] && argv[0][0] == '-')
        isloginsh = 1;
    if (isloginsh) {
        state = 1;
        read_profile("/etc/profile");
state1:
```

```
        state = 2;
        read_profile(".profile");
    ……
}
```

因为我们在/etc/profile对PATH、LD_LIBRARY_PATH、 HOSTNAME、PS1等环境变量进行了修改,所以BusyBox开启的必须是一个login shell（这样可以保证/profile的内容对开发板上所有shell都是有效的）；否则/etc/profile定义的内容将不会执行。

做 个小实验(以这次做好的根文件系统为基础)：

Please press Enter to activate this console.　　#启动开发板，引导linux内核并进行各系统配置后，执行到这里。按下回车键，显示以下内 容：

starting pid 797, tty '/dev/console': '-/bin/sh '　#在控制台/dev/console上开启一个login shell

Processing /etc/profile　　#执行/etc/profile配置文 件

set user path

set search library path

set PS1

root@arm2410s:/#

root@arm2410s:/# exit　#退出当前 shell

Please press Enter to activate this console.　　#再按回车开启一个新的login shell

starting pid 799, tty '/dev/console': '-/bin/sh '

Processing /etc/profile　　#再次执行/etc/profile配置文 件

set user path

set search library path

set PS1

lingd@ubuntu:~/arm2410s/rootfs/mnt/etc$ ls

init.d inittab profile rc.local

3.7、/etc/fstab

文件/etc/fstab存放的是系统中的文件系统信息。当正确的设置了该 文件，则可以通过"mount /directoryname"命令来加载一个文件系统，或mount -a来加载该文件中所有文件系统。每种文件系统都对应一个独立的行，每行中的字段都有空格或tab键分开。同时fsck、mount、umount的等命 令都利用该程序。

lingd@ubuntu:~/arm2410s/rootfs/mnt/etc$ vi fstab

fstab文件内容如下：

#/etc/fstab: static file system information.

#<File system> <mount pt> <type> <options> <dump> <pass>

proc /proc proc defaults 0 0

sysfs /sys sysfs defaults 0 0

mdev /dev ramfs defaults 0 0

none /tmp ramfs defaults 0 0

注意： 已单独mount了的文件系统，就不要出现在/etc/fstab文件中，以免使用mount -a时把先前已经mount上的文件系统被覆盖了。

3.8、/etc/passwd

/etc/passwd文件存放着所有用户的信息，包括账号和密码。内容如下：

#username:password:User ID:Group ID:comment:home directory:shell

root:x:0:0:root:/root:/bin/sh

4、编译busybox

BusyBox下载地址：http://www.busybox.net/

lingd@ubuntu:~/arm2410s$ tar xjvf busy-1.11.1.tar.bz2

lingd@ubuntu:~/arm2410s$ cd busybox-1.11.1

lingd@ubuntu:~/arm2410s/busybox-1.11.1$ vi Makefile

首先修改 Makefile ,将以下两项改为

CROSS_COMPILE = arm-linux-

ARCH = arm

配置busybox，修改以下选项（其他选项默认就可 以了，或者根据需要再裁减一下）：

lingd@ubuntu:~/arm2410s/busybox-1.11.1$ make menuconfig

Busybox Settings --->

Build Options --->

[*] Build BusyBox as a static binary (no shared libs)

[ ] Build with Large File Support (for accessing files > 2 GB)

Installation Options --->

[*] Don't use /usr #这项必选选上，否则BusyBox默认安装路径是/usr，原/usr下的东西可能会被覆盖了

Applets links (as soft-links) --->

(./_install) BusyBox installation prefix

Busybox Library Tuning --->

[*] Command line editing

(1024) Maximum length of input

[*] vi-style line editing commands

(15) History size

[*] History saving

[*] Tab completion

[*] Username completion

[*] Fancy shell prompts

这两项必须选上，不然BusyBox将无法识别环境变量PS1里的参数。

Linux Module Utilities --->

[*] insmod

[*] Enable load map (-m) option

[*] Symbols in load map

[*] rmmod

[*] lsmod

[*] Pretty output for 2.6.x Linux kernels

[*] modprobe

[ ] Multiple options parsing

[ ] Fancy alias parsing

[ ] Blacklist support

--- Options common to multiple modutils

[ ] Support tainted module checking with new kernels

[ ] Support version 2.2.x to 2.4.x Linux kernels

[*] Support version 2.6.x Linux kernels

(/lib/modules) Default directory containing modules

(modules.dep) Default name of modules.dep

保存退出 Busybox

lingd@ubuntu:~/arm2410s/busybox-1.11.1$ make

lingd@ubuntu:~/arm2410s/busybox-1.11.1$ make install

编译BusyBox时，可能会出现以下错误：

4.1、inotifyd出错

交叉编译busybox - 1.11.1时，出现以下错误：

CC miscutils/inotifyd.o

miscutils/inotifyd.c:31:27: linux/inotify.h: No such file or directory

miscutils/inotifyd.c: In function `inotifyd_main':

miscutils/inotifyd.c:61: error: `IN_ALL_EVENTS' undeclared (first use in this function)

miscutils/inotifyd.c:61: error: (Each undeclared identifier is reported only once
miscutils/inotifyd.c:61: error: for each function it appears in.)
miscutils/inotifyd.c:129: error: dereferencing pointer to incomplete type
miscutils/inotifyd.c:139: error: dereferencing pointer to incomplete type
miscutils/inotifyd.c:140: error: dereferencing pointer to incomplete type
miscutils/inotifyd.c:140: error: dereferencing pointer to incomplete type
miscutils/inotifyd.c:143: error: invalid application of `sizeof' to incomplete type `inotify_event'
miscutils/inotifyd.c:143: error: dereferencing pointer to incomplete type
make[1]: *** [miscutils/inotifyd.o] Error 1
make: *** [miscutils] Error 2

网上说这是busybox的一个bug，解决方法：去掉对inotifyed的支持，具体步骤如下：

make menuconfig
Miscellaneous Utilities --->
[ ]inotifyd

还有另一个bug是 taskset，也要将它去掉，不然编译时又会出错。

Miscellaneous Utilities --->
[ ]taskset

4.2、启动时，BusyBox提示"/bin/sh:can't access tty;job console turn off"

解决方法：

shell --->
[ ]Job Console
[*]Cttyhack

4.3、未定义 ARPHRD_INFINIBAND

错误信息如下：

networking/interface.c:818: error: 'ARPHRD_INFINIBAND' undeclared here (not in

a function)

make[1]: *** [networking/interface.o] Error 1

make: *** [networking] Error 2

通过查看内核源代码目录中的“include/linux/ifarp.h”文件可知“ARPHRD_INFINIBAND”的值为“32”。然后修改“networking/interface.c”文件，在其中添加：

#define ARPHRD_INFINIBAND 32 /* InfiniBand */

默认在_install目录中生成基本的文件系统, 包括以下目录或文件bin、sbin、linuxrc该目录下包含了基本的 shell 命令.将编译好的BusyBox的_Install目录下的bin和sbin用tar命令打包复制到~/rootfs目录，解压后删除打包文件。

lingd@ubuntu:~/arm2410s/busybox-1.11.1$ cd _install

lingd@ubuntu:~/arm2410s/busybox-1.11.1/_install$ ls

bin linuxrc sbin

lingd@ubuntu:~/arm2410s/busybox-1.11.1/_install$ tar zcvf rootfs.tar.gz bin sbin

lingd@ubuntu:~/arm2410s/busybox-1.11.1/_install$ ls

bin linuxrc rootfs.tar.gz sbin

lingd@ubuntu:~/arm2410s/busybox-1.11.1/_install$ mv rootfs.tar.gz ../../rootfs

lingd@ubuntu:~/arm2410s/busybox-1.11.1/_install$ ls

bin linuxrc sbin

lingd@ubuntu:~/arm2410s/busybox-1.11.1/_install$ cd ../../rootfs

lingd@ubuntu:~/arm2410s/rootfs$ ls

bin etc lib mnt root sbin tmp var

dev home linuxrc proc rootfs.tar.gz sys usr

lingd@ubuntu:~/arm2410s/rootfs$ tar zvxf rootfs.tar.gz

lingd@ubuntu:~/arm2410s/rootfs$ rm rootfs.tar.gz

5、 lib库文件复制到rootfs/lib目录下（根据需要复制）

lingd@ubuntu:~/arm2410s/rootfs$ cd /opt/crosstool/gcc-4.1.1-glibc-2.3.2/arm-linux/arm-linux/lib

lingd@ubuntu:/opt/crosstool/gcc-4.1.1-glibc-2.3.2/arm-linux/arm-linux/lib$ cp -dR ld* ~/arm2410s/rootfs/lib

lingd@ubuntu:/opt/crosstool/gcc-4.1.1-glibc-2.3.2/arm-linux/arm-linux/lib$ cp -dR libc*.so* ~/arm2410s/rootfs/lib

lingd@ubuntu:/opt/crosstool/gcc-4.1.1-glibc-2.3.2/arm-linux/arm-linux/lib$ cp -dR libcrypt* ~/arm2410s/rootfs/lib

lingd@ubuntu:/opt/crosstool/gcc-4.1.1-glibc-2.3.2/arm-linux/arm-linux/lib$ cp -dR libgcc_s.so* ~/arm2410s/rootfs/lib

lingd@ubuntu:/opt/crosstool/gcc-4.1.1-glibc-2.3.2/arm-linux/arm-linux/lib$ cp -dR libm* ~/arm2410s/rootfs/lib

lingd@ubuntu:/opt/crosstool/gcc-4.1.1-glibc-2.3.2/arm-linux/arm-linux/lib$ cp -dR libpthread.so* ~/arm2410s/rootfs/lib

6、生成 CramFS文件系统镜像文件CramFS.img

首先从http://sourceforge.net/projects/cramfs/下载 cramfs-1.1.tar.gz

lingd@ubuntu:/opt/crosstool/gcc-4.1.1-glibc-2.3.2/arm-linux/arm-linux/lib$ cd ~/arm2410s

lingd@ubuntu:~/arm2410s$ tar zxvf cramfs-1.1.tar.gz

lingd@ubuntu:~/arm2410s$ cd cramfs-1.1

lingd@ubuntu:~/arm2410s/cramfs-1.1$ ls

COPYING cramfsck.c GNUmakefile linux mkcramfs.c NOTES README

lingd@ubuntu:~/arm2410s/cramfs-1.1$ make

gcc -W -Wall -O2 -g -I. mkcramfs.c -lz -o mkcramfs

gcc -W -Wall -O2 -g -I. cramfsck.c -lz -o cramfsck

在 cramfs-1.1目录下会出现两个bin文件mkcramfs和cramfsck

lingd@ubuntu:~/arm2410s/cramfs-1.1$ ls

COPYING cramfsck.c linux mkcramfs.c README

cramfsck GNUmakefile mkcramfs NOTES

把他们copy到/usr/bin:

lingd@ubuntu:~/arm2410s/cramfs-1.1$ sudo cp mkcramfs cramfsck /usr/bin

创建根文件系统的cramfs镜像

lingd@ubuntu:~/arm2410s/cramfs-1.1$ mkcramfs ../rootfs CramFS01.img

Directory data: 6020 bytes

Everything: 908 kilobytes

Super block: 76 bytes

CRC: 526c6bb5

warning: gids truncated to 8 bits (this may be a security concern)

lingd@ubuntu:~/arm2410s/cramfs-1.1$ ls

COPYING cramfsck GNUmakefile mkcramfs NOTES

CramFS01.img cramfsck.c linux mkcramfs.c README

测试一下刚生成的CramFS01.img：

lingd@ubuntu:~/arm2410s/cramfs-1.1$ sudo mount -o loop -t cramfs

~/CramFS01.img /mnt/usb

lingd@ubuntu:~/arm2410s/cramfs-1.1$ ls /mnt/usb

查 看是否有bin、etc等目录

编译cramfs时出现如下错误：

mkcramfs.c:37:18: zlib.h: No such file or directory

mkcramfs.c: In function `write_superblock':

mkcramfs.c:392: warning: implicit declaration of function `crc32'

mkcramfs.c:392: error: `Z_NULL' undeclared (first use in this function)

mkcramfs.c:392: error: (Each undeclared identifier is reported only once

mkcramfs.c:392: error: for each function it appears in.)

mkcramfs.c: In function `do_compress':

mkcramfs.c:598: warning: implicit declaration of function `compress2'

mkcramfs.c:598: error: `Z_BEST_COMPRESSION' undeclared (first use in this function)

mkcramfs.c:599: error: `Z_OK' undeclared (first use in this function)

mkcramfs.c:600: warning: implicit declaration of function `zError'

mkcramfs.c: In function `main':

mkcramfs.c:824: error: `Z_NULL' undeclared (first use in this function)

make: *** [mkcramfs] Error 1

原因没安装 zlib-devel包

在ubuntu软件源里zlib和zlib-devel叫做zlib1g zlib1g.dev

$ sudo apt-get install zlib1g

$ sudo apt-get install zlib1g.dev

参考文章

ARM嵌入式Linux系统开发技术详解 杨水清、张剑、施云飞等编著

使 用mdev制作cramfs根文件系统

http://hi.baidu.com/kkernel/blog/item/cf2d6845b917e836879473b3.html

飞 凌OK-2440-III文件系统移植-------busybox+mdev

http://bbs.21ic.com/icview-155436-1-1.html

s3c2410 平台linux2.6.22内核的文件系统移植总结

http://hi.baidu.com/hzau_wall_e/blog/item/76258afb58b6b5819e5146cc.html/cmtid/d4e418ee679a101cfdfa3cd4

linuxrc 详解

http://blog.chinaunix.net/u1/38576/showart_444508.html

详解 Linuxrc、rcS、rc.local、Profile

http://hi.baidu.com/wangy0919/blog/item/1e0bc18fd27f86f8513d92fb.html

使 用Busybox制作CRAMFS文件系统成功

http://linux.chinaunix.net/techdoc/system/2008/08/20/1026770.shtml

关于根文件系统中命令行提示符的显示

http://www.eefocus.com/ayayayaya/blog/10-01/183654_4d808.html