K9-Ramdisk_4M 文件系统

1. linux 文件系统的基础知识

linux 文件系统至少应包括以下几个内容：
1.基本的文件系统结构,包含一些必需的目录如/bin /etc /dev /proc /lib /usr /tmp 等。
2.基本程序运行所需的库函数如 Glibc/uC_libc
3.基本的系统配置文件，如 rc,inittab 等脚本文件
4.必要的设备支持文件 如 /dev/hd* /dev/tty*
5.基本的应用程序 如 sh,ls,cp
嵌入式 linux 启动过程
1.在第一阶段完成硬件检测、初始化和内核的引导。
2.在第二阶段就是 init 的初始化过程。init 程序通常在/sbin 或/bin 下，从/etc/inittab 获取所有的信息。
所谓的 4M 大小的文件系统，其在 Flash 里面存放的文件一般为压缩文件，并没有 4M；只有当此压缩经拷贝到 SDRAM 并解压之后，才是真实的 4M 大小。

2. 开发 Linux 文件系统的二种方法

对于 Linux 文件系统的开发，有两种方法，一种就是自己从头开始建立根文件系统，另外一种是下载或者获取一个已经生成的文件系统，然后在此基础上添加和修改，最后形成自己的文件系统。

2．1 以一个建好的文件系统为基础来创建

K9 提供简单可用的 4M 文件系统 k9fs4m.gz，用户可以直接使用，或者在这个文件系统的基础上进行自己的文件系统开发。

步骤如下：

- 文件夹设置
  创建文件夹/mnt/tmp_k9fs
  将 k9fs4m.gz 存放在/usr/local/arm/k9fs 下面
- 解开压缩
  gunzip k9fs4m.gz
- 影象文件挂装
  mount -o loop k9fs4m /mnt/tmp_k9fs
- 对/mnt/tmp_k9fs 目录进行操作，增减文件
  bash$ cd /mnt/tmp_k9fs
  bash$ do_what_you_want (create directories, files ...)
  #如在 usr 目录下添加 k9hello 的可执行文件
- 到影象文件目录下
  bash$ cd /usr/local/arm/k9fs
- 卸装文件系统
  bash$ umount /mnt/tmp_k9fs
- 压缩文件系统，生成最终的文件系统影象

bash$ gzip -v9 k9fs4m

最后，检查 k9fs4m.gz 大小，尽量使 k9fs4m.gz 小于 2816(KB)。

至此，重新生成了用户自己的 4M 文件系统，文件名仍为 k9fs4m.gz

2．2 自己建立根文件系统

一般创造根文件系统可以有下面步骤：

- 创建一定大小的根文件系统

  mke2fs -vm0 /dev/ram 4096　　＃4096 表示是创建 4M 的文件系统

- 根文件系统挂装

  mount -t ext2 /dev/ram /mnt　　#ext2 表示创建的是 ext2 文件系统

- 文件系统的操作

  cd /mnt

  cp /bin, /sbin, /etc, /dev ... files in mnt

  cd ../

- 去除挂装

  umount /mnt

- 文件系统生成

  dd if=/dev/ram bs=1k count=4096 of=k9fs4m

- 得到文件系统影象

  gzip -v9 k9fs4m

  最后，检查 k9fs4m.gz 大小，尽量使 k9fs4m.gz 小于 2816(KB)。

  至此，生成了用户自己的 4M 文件系统，文件名为 k9fs4m.gz

mke2fs 是用于在任何设备上创建 ext2 文件系统的实用程序，它创建超级块、索引节点以及索引节点表等等。

在上面的用法中，/dev/ram 是上面构建有 4096 个块的 ext2 文件系统的设备。然后，将这个设备（/dev/ram）挂装在名为 /mnt 的临时目录上并且复制所有必需的文件。一旦复制完这些文件，就卸装这个文件系统并且设备(/dev/ram)的内容被转储到一个文件(k9fs4m)中，它就是所需的 Ramdisk（Ext2 文件系统）。

上面的顺序创建了一个 4 MB 的 Ramdisk，并用必需的文件实用程序来填充它。

附录一： 网上收集的 RAMDISK 文章--Linux 下用 BusyBox 制作 Ramdisk 全过程

发布时间：2007.04.26 06:34　　来源：赛迪网技术社区　　作者：skid

1 建立根文件系统结构

```
#mkdir rootfs
#cd rootfs
#mkdir bin dev etc lib proc sbin tmp usr var
#chmod 1777 tmp
#mkdir usr/bin usr/lib usr/sbin
#mkdir var/lib var/lock var/log var/run var/tmp
#chmod 1777 var/tmp
```

2 准备链接库

```
#cd ${OBJ_LIB}/lib (${OBJ_LIB}是交叉编译环境的目录)
#for file in libc libcrypt libdl libm \
>libpthread libresolv libutil
>do
>cp $file-*.so /home/fortis/rootfs/lib
>cp -d $file.so.[*0-9] /home/fortis/rootfs/lib
>done
#cp -d ld*.so* /home/fortis/rootfs/lib
```

3 使用 busybox 制作系统应用程序

3.1 下载 busybox（http://www.busybox.net/）并解压。

3.2 进入解压后的目录，配置 Busybox

```
$make menuconfig
Busybox Settings >
General Configuration >
[*] Support for devfs
Build Options >
[*] Build BusyBox as a static binary (no shared libs)
/* 将 busybox 编译为静态连接，少了启动时找动态库的麻烦 */
[*] Do you want to build BusyBox with a Cross Compiler?
(/usr/local/arm/3.3.2/bin/armlinux)
Cross Compiler prefix/* 指定交叉编译工具路径 */
Init Utilities >
[*] init
[*] Support reading an inittab file
/* 支持 init 读取/etc/inittab 配置文件，一定要选上 */
Shells >
Choose your default shell (ash) >
  /* (X) ash 选中 ash，这样生成的时候才会生成 bin/sh 文件
    * 看看我们前头的 linuxrc 脚本的头一句：
    * #!/bin/sh 是由 bin/sh 来解释执行的*/
[*] ash
Coreutils >
```

[*] cp
[*] cat
[*] ls
[*] mkdir
[*] echo (basic SuSv3 version taking no options)
[*] env
[*] mv
[*] pwd
[*] rm
[*] touch
Editors >
[*] vi
Linux System Utilities >
[*] mount
[*] umount
[*] Support loopback mounts
[*] Support for the old /etc/mtab file
Networking Utilities >
[*] inetd
/* * 支持 inetd 超级服务器 */

3.3 编译并安装 Busybox
$make TARGET_ARCH=arm CROSS=armlinux\
PREFIX=/home/arm/dev_home/rootfs/my_rootfs/ all install
PREFIX 指明安装路径：就是我们根文件系统所在路径。

4 准备所需的设备文件
可以直接拷贝宿主机上的，或者自建几个就是。
#cd rootfs/dev
#mknod -m 600 console c 5 1

5 创建 linuxrc 文件
内容如下:
$ vim rootfs/linuxrc
#!/bin/sh
echo "Hello linux ,gggggg"
exec /sbin/init
然后修改权限：chmod 775 linuxrc

6 制作 initrd 映象文件
#mkdir initrd
#dd if=/dev/zero of=initrd.img bs=1k count=8192
#/sbin/mke2fs -F -v -m0 initrd.img
#mount -o loop initrd.img initrd

```
#cp -av rootfs/* initrd
#umount
#gzip -9 initrd.img
```

附录二：群友编写的 Ramdisk 制作文章－－作者 dandan

1、编译busybox

（1）解压源码

cd /work

tar -zxf

busybox1.4.1.

tar.gz

（2）修改源码(解决静态编译busybox 时的BUG)

/work/busybox1.4.1/

applets/applets.c 作以下修改

```
/* Apparently uclibc defines __GLIBC__ (compat trick?). Oh well. */
/*#if ENABLE_STATIC && defined(__GLIBC__) && !defined(__UCLIBC__)
#warning Static linking against glibc produces buggy executables
#warning (glibc does not cope well with ld -gcsections).
#warning See sources.redhat.com/bugzilla/show_bug.cgi?id=3400
#warning Note that glibc is unsuitable for static linking anyway.
#warning If you still want to do it, remove -Wl,
-gcsections
#warning from toplevel
Makefile and remove this warning.
#endif*/
```

（3）修改Makefile

```
ARCH ?= $(SUBARCH)
CROSS_COMPILE ?=
```

改为

```
ARCH ?= arm
CROSS_COMPILE ?=armlinux（
```

4）配置busybox

cd /work/busybox1.4.1

make defconfig

make menuconfig

（5）配置选项（没有提到的选项使用默认设置）

Busybox Settings>

Build Options>

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

Installation Options>

[*] Don't use /usr

Linux Module Utilities>（动态加载容易出错，故暂不用）

[ ] insmod

[ ] rmmod

[ ] lsmod

[ ] modprobe

Miscellaneous Uyilities>（默认有这两项，编译通不过，故暂不用，原因待查）

[ ] readahead
[ ] taskset
Shells>
Choose your default Shell>（特别重要，不然进入LINUX 后无法找到SHELL）
(X) ash
[*]所有选项
（6）编译busybox
cd /work/busybox1.4.1
make
make install
2、定制文件系统
（1）创建目录结构
cd /work/fs
mkdir dev etc lib mnt proc usr
（2）创建设备节点
cd /work/fs/dev
mknod console c 5 1
mknod full c 1 7
mknod kmem c 1 2
mknod mem c 1 1
mknod null c 1 3
mknod port c 1 4
mknod random c 1 8
mknod urandom c 1 9
mknod zero c 1 5
for i in `seq 0 7`; do （特别注意，` 不是单引号,而是TAB健上边那个，下同）
mknod loop$i b 7 $i
done
for i in `seq 0 9`; do
mknod ram$i b 1 $i
done
ln -s ram1 ram
mknod tty c 5 0
for i in `seq 0 9`; do
mknod tty$i c 4 $i
done
for i in `seq 0 9`; do
mknod vcs$i b 7 $i
done
ln -s vcs0 vcs
for i in `seq 0 9`; do
mknod vcsa$i b 7 $i
done
ln -s vcsa0 vcsa

（3）添加应用程序

```
# cd /work
# cp  -a ./busybox1.4.1/_install/* ./fs
```

（4）配置系统文件

```
# cd /work
# tar  -zxf rootfs.tar.gz
# cp  -a ./rootfs/etc/* ../fs/etc
```

（5）添加运行时库（主要是网络命令和DNS解析使用）

```
# cd /work
# cp -a
./rootfs/lib/* ./fs/lib
```

3、制作RAMDISK （编译成UBOOT格式导致加载错误，故不用）

```
# cd /work
# mkdir /mnt/initrd
# dd if=/dev/zero of=K9.img bs=1k count=4096
# mkfs.ext2  -F K9.img
# mount  -o loop K9.img /mnt/initrd
# cp  -a ./fs/* /mnt/initrd
# mount /mnt/initrd
# gzip  -best  -c K9.img > K9.img.gz
```

4、发布RAMDISK到TFTP

```
# cp /work/K9.img.gz /tftpboot
```