构建根文件系统

uboot 启动内核，kernel 启动应用程序，应用程序在文件系统中。因此要构建文件系统。

printf()标准输出；scanf() ；标准输入；err() 标准错误

mount -a 的意思是将/etc/fstab的所有内容重新加载。开机时 fstab 已经挂载后

另外，你可以运行mount，会列出所有已经挂载的项。

可以运行fdisk -l，查看所有磁盘分区，对比mount的结果，就可以知道挂载到哪里了

内核如何启动第一个应用程序：

/linuxrc //exec init

/sbin/init

parse\_inittab();//inittab用来指示在何时执行哪个action,对应某个程序。

/etc/init.d/rcS //在inittab脚本里面指示先执行rcS,而rcS 中mount -a 的意思是将/etc/fstab的所有内容重新加载。

main.c 开始 //内核源文件

kernel\_init

init\_post

sys\_open((const char \_\_user \*) "/dev/console", O\_RDWR, 0)// 打开控制台，标准输入使用控制台，我们程序使用串口1

(void) sys\_dup(0); //标准输出也使用控制台

(void) sys\_dup(0); //标准错误也使用控制台

if (execute\_command) //在命令行设置 execute\_command init=/linuxrc

run\_init\_process(execute\_command);

实际上是执行linuxrc,而linux 最终执行exec /sbin/init.

/sbin/init 实际上是busybox的Init命令。实际执行程序init\_main();

busybox是一系列命令的集合：如ls,cd.执行ls 实际是执行busybox里面的ls程序。

# ls -l /bin/ls

lrwxrwxrwx 1 1000 1000 7 Jan 6 2010 /bin/ls -> busybox

# busybox ls

bin lib mnt sbin usr

dev linuxrc proc sys

etc lost+found root tmp

init程序：

1、读取配置文件

2、解析配置文件

3、执行应用程序

busybox:init.c

init\_main

parse\_inittab();

file = fopen(INITTAB, "r");// INITTAB就是/ect/inittab

new\_init\_action(a->action, command, id);

run\_actions(SYSINIT);//执行 sysinit类的process

waitfor(a, 0); //执行应用程序，等待它执行完成，执行完成后删除

run(a);

wpid = waitpid(runpid, &status, 0);

delete\_init\_action(a);

run\_actions(WAIT);

waitfor(a, 0);

run(a);

wpid = waitpid(runpid, &status, 0);

delete\_init\_action(a);

run\_actions(ONCE);//不会等待执行完毕

run(a);

delete\_init\_action(a);

while (1) {

run\_actions(RESPAWN);

if (a->pid == 0) {

a->pid = run(a);

run\_actions(ASKFIRST);

if (a->pid == 0) {

a->pid = run(a);

if (a->action & ASKFIRST) {

"\nPlease press Enter to activate this console. ";

if (a->pid == wpid) {

a->pid = 0;

}

inittab配置文件格式

<id>:<runlevels>:<action>:<process>

id: 用作终端 /dev/id stdin stdout stderr printf scanf err从哪里输入，输出到哪里

runlevels： 可以忽略

action:何时执行

process：被执行的应用程序脚本

例子：command 即是process

new\_init\_action(a->action, command, id);

new\_init\_action(ASKFIRST, bb\_default\_login\_shell, VC\_2);

const char bb\_default\_login\_shell[] ALIGN1 = LIBBB\_DEFAULT\_LOGIN\_SHELL;

#define ASKFIRST 0x004

#define LIBBB\_DEFAULT\_LOGIN\_SHELL "-/bin/sh"

# define VC\_2 "/dev/tty2"

id=VC\_2

process=/bin/sh

action=ASKFIRST

new\_init\_action 内部操作：

1、构建init\_action结构

2、将Init\_action加入到init\_action\_list链表中。

new\_init\_action(CTRLALTDEL, "reboot", "");

new\_init\_action(SHUTDOWN, "umount -a -r", "");

new\_init\_action(RESTART, "init", "");

new\_init\_action(ASKFIRST, bb\_default\_login\_shell, "");

new\_init\_action(ASKFIRST, bb\_default\_login\_shell, VC\_2);

new\_init\_action(ASKFIRST, bb\_default\_login\_shell, VC\_3);

new\_init\_action(ASKFIRST, bb\_default\_login\_shell, VC\_4);

new\_init\_action(SYSINIT, INIT\_SCRIPT, "");

::ctrlaltdel:reboot //同时按下 ctrl+alt+del键重启系统

::shutdown:unmount -a -r //卸载所有文件系统

::restart:init

::askfirst:-bin/sh

tty2::askfirst:-/bin/sh

tty3::askfirst:-/bin/sh

::sysinit:/etc/init.d/rcS

## 根文件系统里面都需要什么：

1、/dev/console /dev/null

2、/ect/inittab

配置文件里指定的应用程序

3、c库

4、init 即busybox

配置编译busybox

vim Makefile

/CROSS 搜索CROSS

ARCH ?= arm

CROSS\_COMPILE ?=arm-linux-

make menuconfig

make

make CONFIG\_PREFIX=/work/nfsroot/firstfs/ install//不用加sudo

1、打开Makefile,搜索/CROSS.设置CROSS\_COMPILE ?=arm-linux-

2、make menuconfig

1>设置TAB不全

busybox setting---->

busybox library tuning ---->

[\*] Tab completion

2>是否使用静态连接，不使用。默认也是不使用

build option----->

[] build busybox as a static binary

使用静态连接时会出现问题。使用动态连接busybox,构造文件系统时需要在/lib目录下放置glic库文件

3>Archival utilities 默认已经选上所有

选择tar 命令

general setting--->

archival utilities ---->

Enable tar;enable archive, -j ;-x;-z...

4> linux module utilities 模块相关，默认已经选上所有需要的

5>linux system utilities 默认都已经选上

支持mdev,可以很方便的构造/dev目录，并且可以支持热插拔设备，另外选中mount、umount

并且让mount命令支持NFS网络文件系统

[\*] mdev │ │

[\*] Support /etc/mdev.conf

[\*] Support command execution at device addition/removal

[\*] mount

[\*] Support mounting NFS file systems

[\*] umount │ │

[\*] umount -a option

6> networking utilities

[\*] ifconfig

[\*] Enable status reporting output (+7k) │ │

[ ] Enable slip-specific options "keepalive" and "outfill" │ │

[ ] Enable options "mem\_start", "io\_addr", and "irq" │ │

[\*] Enable option "hw" (ether only) │ │

[\*] Set the broadcast automatically

创建根文件系统：

切换到的busybox 安装目录里面有bin,linuxrc,sbin,usr文件

1、创建设备节点

cd /work/nfs/firstfs

mkdir dev

cd dev/

sudo mknod console c 5 1

sudo mknod null c 1 3

cd ..

mkdir etc

vim etc/inittab

2、拷贝glibc库

-a是静态库，so是动态库

1、cd /work/tools/gcc-3.4.5-glibc-2.3.6/arm-linux/lib/

2、 mkdir /work/nfsroot/firstfs/lib

3、cp \*.so\* /work/nfsroot/firstfs/lib -d //-d 如果是链接拷贝链接文件

3、制作yaffs2 Image

制作工具，system目录的yaffs\_source\_util\_larger\_small\_page\_nand.tar.bz2

拷贝到 虚拟机/work/tools

tar xjf yaffs\_source\_util\_larger\_small\_page\_nand.tar.bz2

cd Development\_util\_ok //Development\_util\_ok就是解压后的文件

cd yaffs2/utils/

make //生成mkyaffs2image

sudo cp mkyaffs2image /usr/local/bin

sudo chmod +x /usr/local/bin/mkyaffs2image

cd /work/nfsroot/

mkyaffs2image firstfs firstfs.yaffs2 //firstfs是文件系统所在目录

完善文件系统：

1、firstfs目录创建proc目录

mkdir proc

2、vim etc/inittab

添加 ::sysinit:/etc/init.d/rcS

cd etc/

mkdir init.d

cd init.d

vim rcS

//把proc这个虚拟文件系统挂载到/proc目录，mount [-t vfstype] [-o options] device dir

//none标识无真实设备

// mount -t proc none /proc 使用mount -a 代替

mount -a //根据/etc/fstab 内容挂接根文件系统

chmod +x etc/init.d/rcS //添加执行权限

使用mdev构建文件系统

fstab 里面添加

# device mount-point type options dump fsck order

proc /proc proc defaults 0 0

tmpfs /tmp tmpfs defaults 0 0

sysfs /sys sysfs defaults 0 0

tmpfs /dev tmpfs defaults 0 0

mount -a

rcS文件添加

mkdir /dev/pts

mount -t devpts devpts /dev/pts

echo /sbin/mdev > /proc/sys/kernel/hotplug

mdev -s // 创建设备节点null和console. Dev目录指示创建那些设备节点

ifconfig eth0 192.168.0.112

mount -t nfs -o nolock,vers=2 192.168.0.111:/home/chao/work /mnt

Inittab 文件

# /etc/inittab

::sysinit:/etc/init.d/rcS

s3c2410\_serial0::askfirst:-/bin/sh

::ctrlaltdel:/sbin/reboot

::shutdown:/bin/umount -a -r

创建其他文件夹

mkdir mnt tmp sys root

制作映像文件：

mkyaffs2image firstfs firstmdevfs.yaffs2

uboot可以设置启动文件类型，一般使用yaffs2，不用设置

roofstype=jffs2/yaffs2

通过网络挂载根文件系统

1、#sudo apt-get install nfs-kernel-server

ubuntu12.04中的已经是最新版本了，无需安装

2、sudo /etc/init.d/nfs-kernel-server restart

3、sudo mount -t nfs localhost:/work/nfsroot/firstfs /mnt

4、sudo umount /mnt //虚拟机中的挂载

5、 umount /mnt //开发板中之前挂载的

6、 mount -t nfs -o nolock,vers=2 192.168.0.111:/work/nfsroot/firstfs /mn

7、setenv bootargs 'noinitrd console=ttySAC0 root=/dev/nfs nfsroot=192.168.0.111:/work/nfsroot/firstfs

ip=192.168.0.112:192.168.0.111:192.168.0.1:255.255.255.0::eth0:off'

8、save

9、重新启动