根文件系统制作

一、busybox移植

1、概述

在前文的分析中知道在内核启动和挂载了根文件系统后通过运行/sbin/init来使进程1从内核态到用户态的转化，并且该进程然后会执行相应的初始化动作和生成其他进程。大致行为就是做一些初始化(比如设置信号处理函数)，然后解析inittab文件，根据inittab文件的内容做相应的事情，inittab文件的格式和含义在LINUX内核配置编译一文中讲解，这里不再重复，接下来就重点关注如何利用busybox制作部分根文件系统内容，已经如何制作inittab文件和其他剩下的根文件系统部分，最后得到完整的根文件系统。

2、busybox配置

在配置和编译之前先修改busybox的makefile文件，使得busybox对应好相应的架构。修改如下：

```
CROSS_COMPILE ?= arm-linux-gnueabihf-
...
ARCH ?= arm
```

修改完后通过make menuconfig进入配置界面。然后大部分使用默认配置，做部分修改如下：

(1) 编译链接选项

设置busybox为静态链接，这样busybox运行不需要额外的动态库。

```
Busybox Settings --->
    Busybox Options --->
        [*]Build BusyBox as a static binary
        ...
```

(2) 网络选项

需要去除inetd功能，否则编译会出错，可能的原因是busybox版本和交叉编译工具版本不匹配的问题。

```
Networking Utilities --->
    []inetd (NEW)
```

修改完后保存退出。

3、busybox编译和安装

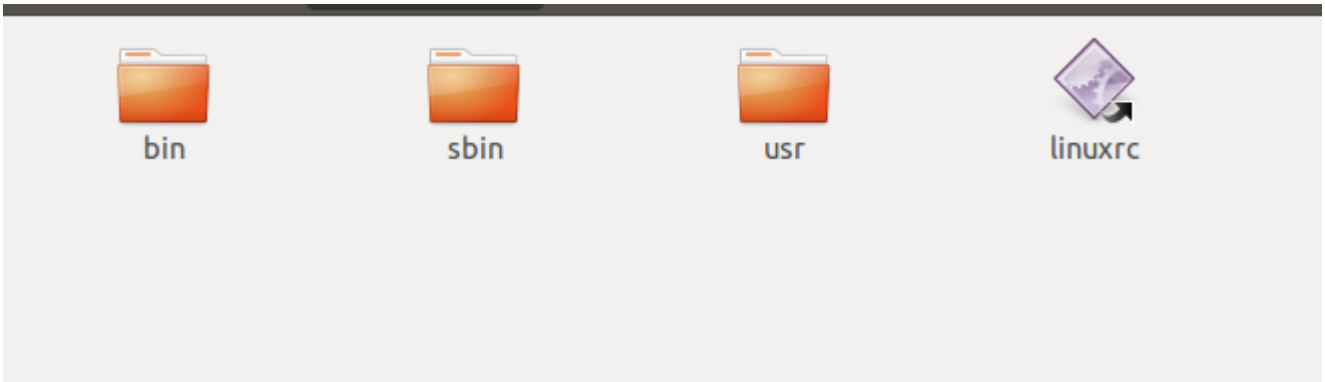
(1) 编译

输入make -j4执行编译。最后成功如下：

```
  CC      util-linux/readprofile.o
  CC      util-linux/rev.o
  CC      util-linux/rtcwake.o
  CC      util-linux/script.o
  CC      util-linux/scriptreplay.o
  CC      util-linux/setarch.o
  CC      util-linux/swaponoff.o
  CC      util-linux/switch_root.o
  CC      util-linux/umount.o
  AR      util-linux/lib.a
  LINK    busybox_unstripped
Trying libraries: crypt m
 Library crypt is not needed, excluding it
 Library m is needed, can't exclude it (yet)
Final link with: m
```

(2) 安装

编译完后可以选择将编译后得到的东西安装到指定的文件夹，我选择安装到/home/hn/TEST/busybox-rootfs目录，命令为make CONFIG_PREFIX=/home/hn/TEST/busybox-rootfs install。最后在busybox-roofs目录得到如下的内容:



这里面有一个叫busybox的可执行程序，在bin目录中，剩下的全都是符号链接，通过直接或间接连接到busybox，包括sbin/init。这些是通过busybox得到的主要的根文件系统内容，剩下部分需要通过人工来生成。

二、构建其他子目录

1、构建/etc目录

(1) 创建/etc/inittab文件

init进程做了相应初始化后根据/etc/inittab文件来创建其他子进程，比如调用配置脚本配置IP，挂载其他文件系统，启动shell等。

```
# my inittab

::sysinit:/etc/init.d/rcS
::sysinit:/usr/bin/static_sysinit
::sysinit:/usr/bin/dynamic_sysinit
::askfirst:-/bin/sh
::shutdown:/usr/bin/static_shutdown
::shutdown:/usr/bin/dynamic_shutdown
::shutdown:/bin/umount -a -r
```

```
VFS: Mounted root (ext3 filesystem) on device 179:2.
devtmpfs: mounted
Freeing init memory: 148K
this is feth0: device MAC address 0a:12:e2:95:f3:6c
first rcS
static sysinit
dynamic sysinit

Please press Enter to activate this console.  ++OTG Interrupt: A-Device Timeout Change++

/ # reboot
/ # static shutdown
dynamic shutdown
umount: devtmpfs busy - remounted read-only
The system is going down NOW!
Sent SIGTERM to all processes
SeDisabling non-boot CPUs ...
Restarting system.
```

上述现象验证了该脚本，首先执行/etc/init.d/rcS，/usr/bin/static_sysinit，/usr/bin/dynamic_sysinit，然后打印please press Enter to activate this console，在按下enter键后进入控制台模式，然后通过reboot重启系统，此时三条shutdown相关的条目执行。

(2) 创建/etc/init.d/rcS文件

inittab第一个条目就是执行rcS，如果没有rcS，busybox会打印报警信息。

```
#!/bin/sh

echo "this is first rcS"
ifconfig eth0 172.16.89.185
mount -a

# mdev相关
echo /sbin/mdev > /proc/sys/kernel/hotplug
mdev -s
```

该执行脚本首先通过ifconfig配置网卡ip地址，然后通过mount -a挂载/etc/fstab文件中指定的文件系统，最后两条指令是mdev相关的，在下文有讲解。

(3) 创建/etc/fstab文件

fstab文件用来控制mount的行为。具体用法参考相关书籍或网上资料。

```
#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
```

2、构建/dev目录

(1) 静态创建设备文件

```
sudo mknod console c 5 1
sudo mknod null c 1 3
sudo mknod ttyS0 c 4 64
sudo mknod mmcblk0 b 179 0
sudo mknod mmcblk0p1 b 179 1
sudo mknod mmcblk0p2 b 179 2
sudo mknod mmcblk0p3 b 179 3
```

(2) 使用mdev动态创建设备文件

mdev是udev简化版，它通过内核发现设备提供的信息来动态创建/dev目录中相应的设备文件以及其他热插拔动作。要使用mdev需要内核sysfs的支持。在/etc/init.d/rcS文件中增加的最后两条指令就是对mdev的使用，其中echo /sbin/mdev > /proc/sys/kernel/hotplug含义是当有热插拔事件发生时就调用/sbin/mdev，mdev -s 含义是扫描/sys目录下的相关的文件来生成/dev目录下的设备文件。

3、构建/lib目录

将工程原来的yocto制作的根文件系统的/lib目录中的内容全部拷贝到/lib目录下，然后应用程序交叉编译不加-static 选项也能运行了，如前文中的dynamic_sysinit和dynamic_shutdown都是不加-static的动态链接程序，最后正常运行。

4、其他目录

其他目录可以不需要创建内容，只创建空目录。

```
mkdir proc mnt tmp sys root
```

执行完上述所有步骤后就得到了如下所示的完整的文件夹形式的根文件系统。