

嵌入式系统设计 实验报告

学生姓名 许强

学生学号 SA18225428

实验日期 2018.10.30

实 验 报 告

一、实验名称：Linux 文件系统构建实验

二、实验学时：4 学时

三、实验内容和目的：

1、实验内容：

(1) 根文件系统开发实验

(2) Ramdisk 文件系统制作实验

2、实验目的：熟悉 Linux 文件系统目录结构，创建自己的文件系统，通过 NFS 方式测试。

四、实验原理：

文件是计算机系统的软件资源，操作系统本身和大量的用户程序、数据都是以文件形式组织和存放的，对这些资源的有效管理和充分利用是操作系统的重要任务之一。

Linux 最早的文件系统是 Minix，但是专门为 Linux 设计的文件系统——扩展文件系统第二版或 EXT2 被设计出来并添加到 Linux 中，这对 Linux 产生了重大影响。EXT2 文件系统功能强大、易扩充、性能上进行了全面优化，也是所有 Linux 发布和安装的标准文件系统类型。

每个实际文件系统从操作系统和系统服务中分离出来，它们之间通过一个接口层：虚拟文件系统或 VFS 来通讯。VFS 使得 Linux 可以支持多个不同的文件系统，每个表示一个 VFS 的通用接口。由于软件将 Linux 文件系统的所有细节进行了转换，所以 Linux 核心的其它部分及系统中运行的程序将看到统一的文件系统。Linux 的虚拟文件系统允许用户同时能透明地安装许多不同的文件系统。

在 Linux 文件系统中，作为一种特殊类型 /proc 文件系统只存在内存当中，而不占用外存空间。它以文件系统的方式为访问系统内核数据的操作提供接口。/proc 文件系统是一个伪文件系统，用户和应用程序可以通过 /proc 得到系统的信息，并可以改变内核的某些参数。

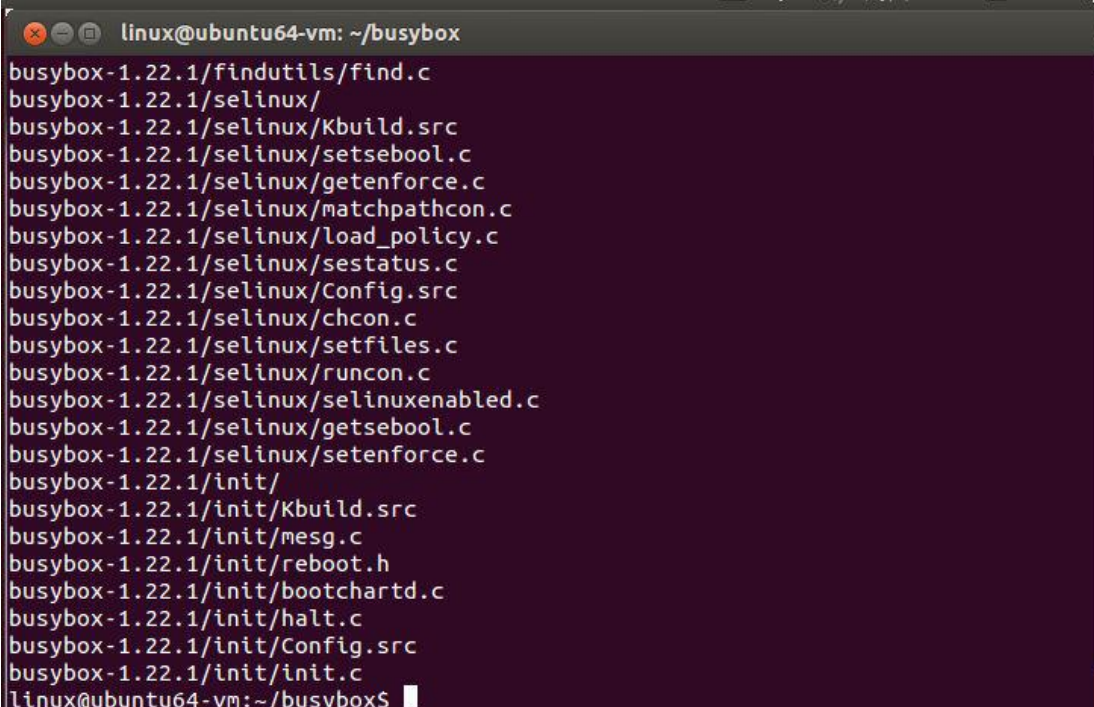
在 Linux 文件系统中，EXT2 文件系统、虚拟文件系统、/proc 文件系统是三个具有代表性的文件系统，本论文试图通过对他们的分析来研究 Linux 文件系统

机制。并且在分析这三种文件系统的基础上对 Linux 文件系统操作进行了解、研究

五、实验步骤：

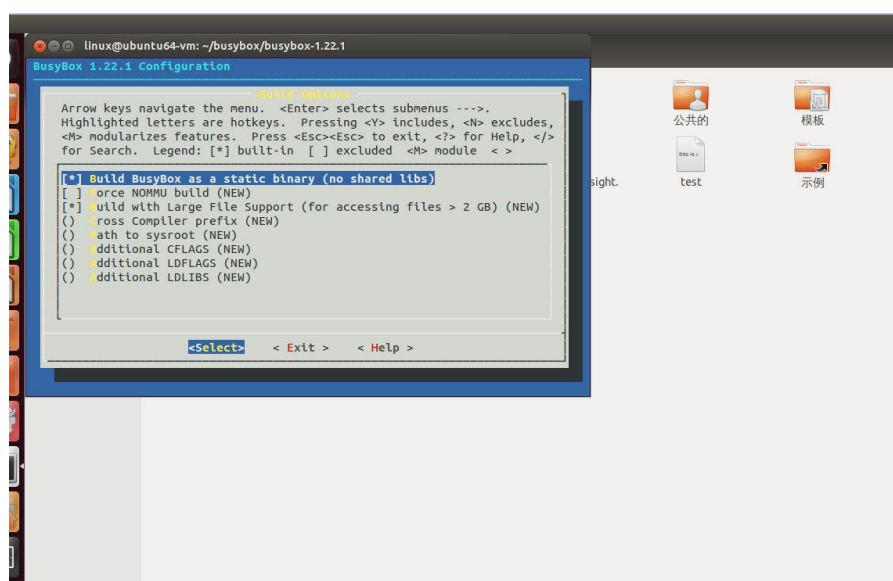
1、根文件系统开发实验

(1) 将 busybox-1.22.1.tar.bz2 文件拷贝到 Ubuntu 虚拟机上，解压文件系统，如下图所示：



```
linux@ubuntu64-vm: ~/busybox
busybox-1.22.1/findutils/find.c
busybox-1.22.1/selinux/
busybox-1.22.1/selinux/Kbuild.src
busybox-1.22.1/selinux/setsebool.c
busybox-1.22.1/selinux/getenforce.c
busybox-1.22.1/selinux/matchpathcon.c
busybox-1.22.1/selinux/load_policy.c
busybox-1.22.1/selinux/sestatus.c
busybox-1.22.1/selinux/Config.src
busybox-1.22.1/selinux/chcon.c
busybox-1.22.1/selinux/setfiles.c
busybox-1.22.1/selinux/runcon.c
busybox-1.22.1/selinux/selinuxenabled.c
busybox-1.22.1/selinux/getsebool.c
busybox-1.22.1/selinux/setenforce.c
busybox-1.22.1/init/
busybox-1.22.1/init/Kbuild.src
busybox-1.22.1/init/mesg.c
busybox-1.22.1/init/reboot.h
busybox-1.22.1/init/bootchartd.c
busybox-1.22.1/init/halt.c
busybox-1.22.1/init/Config.src
busybox-1.22.1/init/init.c
linux@ubuntu64-vm:~/busybox$
```

(2) 配置 busybox 源码，效果图如下所示：



(3) 编译源码

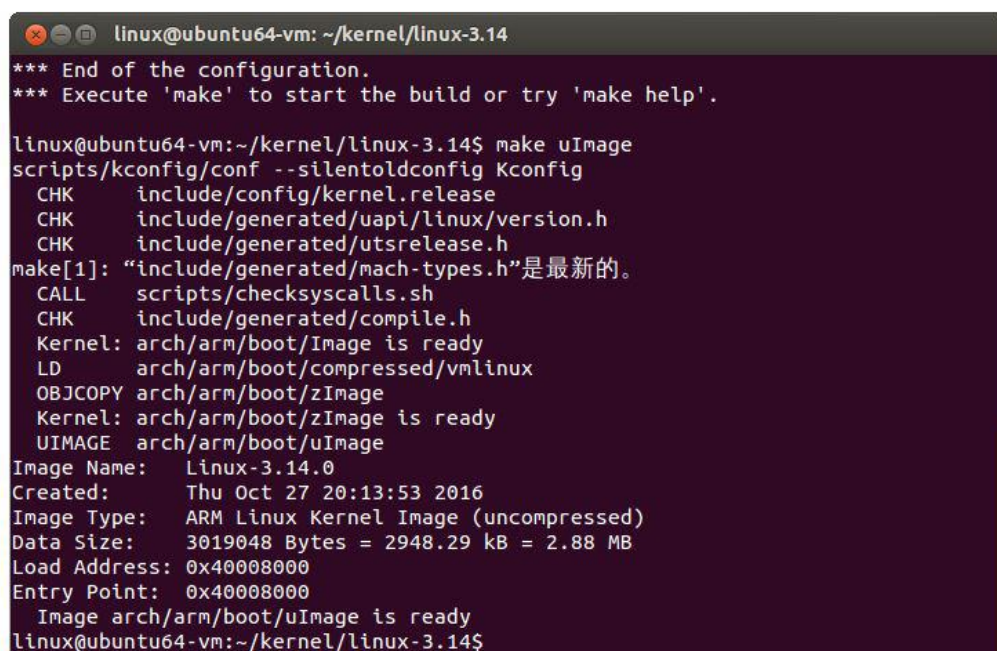
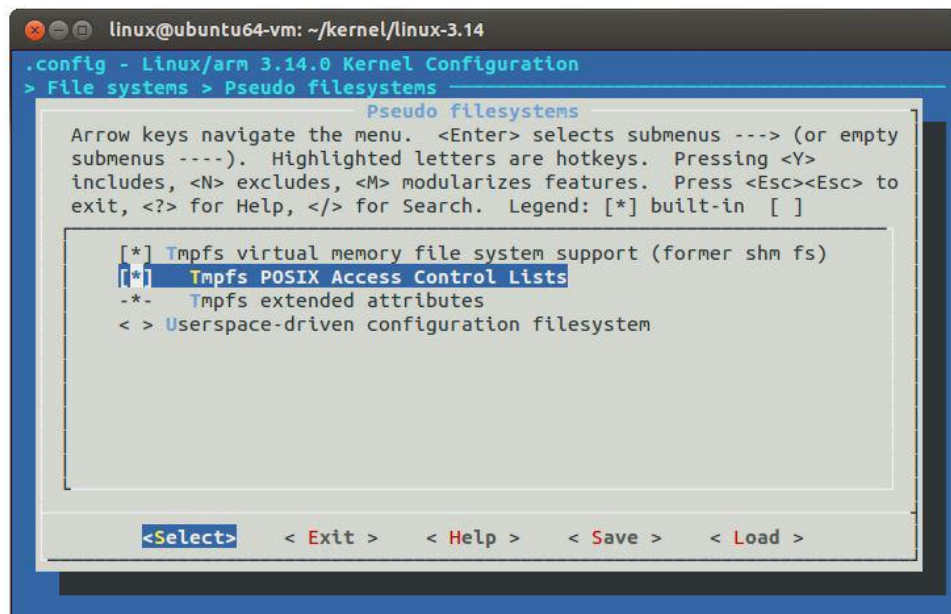
```
linux@ubuntu64-vm: ~/busybox/busybox-1.22.1
CC      util-linux/volume_id/linux_raid.o
CC      util-linux/volume_id/linux_swap.o
CC      util-linux/volume_id/luks.o
CC      util-linux/volume_id/nilfs.o
CC      util-linux/volume_id/ntfs.o
CC      util-linux/volume_id/ocfs2.o
CC      util-linux/volume_id/reiserfs.o
CC      util-linux/volume_id/romfs.o
CC      util-linux/volume_id/sysv.o
CC      util-linux/volume_id/udf.o
CC      util-linux/volume_id/util.o
CC      util-linux/volume_id/volume_id.o
CC      util-linux/volume_id/xfs.o
AR      util-linux/volume_id/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
DOC     busybox.pod
DOC     BusyBox.txt
DOC     busybox.1
DOC     BusyBox.html
linux@ubuntu64-vm:~/busybox/busybox-1.22.1$
```

(4) 创建需要的目录文件

```
linux@ubuntu64-vm: ~/busybox/busybox-1.22.1/_install
-----
linux@ubuntu64-vm:~/busybox/busybox-1.22.1$ ls
applets          console-tools   libbb           printutils
applets_sh       coreutils       libpwdgrp       procp
arch             debianutils     LICENSE         README
archival         docs            loginutils      runit
AUTHORS          e2fsprogs       mailutils       scripts
busybox          editors         Makefile.custom selinux
busybox.links    examples        Makefile.flags  shell
busybox_unstripped findutils       Makefile.help   sysklogd
busybox_unstripped.map include         miscutils       testsuite
busybox_unstripped.out init            modutils        TODO
Config.in        _install       networking      TODO_unicode
configs          INSTALL        networking      util-linux
linux@ubuntu64-vm:~/busybox/busybox-1.22.1$ cd _install/
linux@ubuntu64-vm:~/busybox/busybox-1.22.1/_install$ ls
bin  linuxrc /sbin  usr
linux@ubuntu64-vm:~/busybox/busybox-1.22.1/_install$ mkdir dev etc mnt proc vat
linux@ubuntu64-vm:~/busybox/busybox-1.22.1/_install$ mkdir var tmp sys root
linux@ubuntu64-vm:~/busybox/busybox-1.22.1/_install$ rm -rf vat
linux@ubuntu64-vm:~/busybox/busybox-1.22.1/_install$ ls
bin  dev  etc  linuxrc  mnt  proc  root /sbin  sys  tmp  usr  var
linux@ubuntu64-vm:~/busybox/busybox-1.22.1/_install$
```

(5) 删除静态库和共享库文件中的符号表，效果图如下所示：

- (6) 添加系统启动文件 inittab
- (7) 配置内核和编译内核



- (8) 添加 rcS 文件，并修改文件的权限

```
linux@ubuntu64-vm: ~/busybox/busybox-1.22.1/_install/etc/init.d
UIMAGE arch/arm/boot/uImage
Image Name: Linux-3.14.0
Created: Thu Oct 27 20:13:53 2016
Image Type: ARM Linux Kernel Image (uncompressed)
Data Size: 3019048 Bytes = 2948.29 kB = 2.88 MB
Load Address: 0x40008000
Entry Point: 0x40008000
Image arch/arm/boot/uImage is ready
linux@ubuntu64-vm:~/kernel/linux-3.14$ cp arch/arm/boot/uImage /tftpboot/
linux@ubuntu64-vm:~/kernel/linux-3.14$ cd /home/linux/busybox/busybox-1.22.1/_install/
linux@ubuntu64-vm:~/busybox/busybox-1.22.1/_install$ cd etc
linux@ubuntu64-vm:~/busybox/busybox-1.22.1/_install/etc$ ls
fstab inittab
linux@ubuntu64-vm:~/busybox/busybox-1.22.1/_install/etc$ mkdir init.d
linux@ubuntu64-vm:~/busybox/busybox-1.22.1/_install/etc$ cd init.d/
linux@ubuntu64-vm:~/busybox/busybox-1.22.1/_install/etc/init.d$ ls
linux@ubuntu64-vm:~/busybox/busybox-1.22.1/_install/etc/init.d$ vim rcs
linux@ubuntu64-vm:~/busybox/busybox-1.22.1/_install/etc/init.d$ cat rcs
#!/bin/sh
/bin/mount -a
echo /sbin/mdev > /proc/sys/kernel/hotplug
/sbin/mdev -s
linux@ubuntu64-vm:~/busybox/busybox-1.22.1/_install/etc/init.d$
```

(9) 添加 profile 文件

```
linux@ubuntu64-vm: /
#!/bin/sh
/bin/mount -a
echo /sbin/mdev > /proc/sys/kernel/hotplug
/sbin/mdev -s
linux@ubuntu64-vm:~/busybox/busybox-1.22.1/_install/etc/init.d$ chomd +x rcs
No command 'chomd' found, did you mean:
  Command 'chmod' from package 'coreutils' (main)
chmod: command not found
linux@ubuntu64-vm:~/busybox/busybox-1.22.1/_install/etc/init.d$ chmod +x rcs
linux@ubuntu64-vm:~/busybox/busybox-1.22.1/_install/etc/init.d$ cd ../
linux@ubuntu64-vm:~/busybox/busybox-1.22.1/_install/etc$ ls
fstab init.d inittab
linux@ubuntu64-vm:~/busybox/busybox-1.22.1/_install/etc$ vim profile
linux@ubuntu64-vm:~/busybox/busybox-1.22.1/_install/etc$ cat profile
#!/bin/sh
export HOSTNAME=farsight
export USER=root
export HOME=root
export PS1="[$USER@$HOSTNAME\W]\#"
PATH=/bin:/sbin:/usr/bin:/usr/sbin
LD_LIBRARY_PATH=/lib:/usr/lib:$LD_LIBRARY_PATH
export PATH LD_LIBRARY_PATH
linux@ubuntu64-vm:~/busybox/busybox-1.22.1/_install/etc$ cd /
linux@ubuntu64-vm:/$
```

(10) 删除原先的/source/rootfs 目录下的文件，将我们新建的根文件系统拷贝到/source/rootfs 目录下，以 NFS 文件系统的挂载方式启动开发板，效果图如下：


```
linux@ubuntu64-vm: /source/rootfs
export HOME=root
export PS1="[$USER@$HOSTNAME\W]\#"
PATH=/bin:/sbin:/usr/bin:/usr/sbin
LD_LIBRARY_PATH=/lib:/usr/lib:$LD_LIBRARY_PATH
export PATH LD_LIBRARY_PATH
linux@ubuntu64-vm:~/busybox/busybox-1.22.1/_install/etc$ cd /
linux@ubuntu64-vm:/$ ls
bin      etc      lib      media    root     source   tmp
boot     fontconfig lib32     mnt      run      srv      usr
cdrom    home     lib64    opt      sbin     sys      var
dev      initrd.img lost+found proc      selinux  vmlinux  vmlinux
linux@ubuntu64-vm:/$ sudo rm -rf /source/rootfs
linux@ubuntu64-vm:/$ cd /source/
linux@ubuntu64-vm:/source$ ls
rootfs.tar.xz
linux@ubuntu64-vm:/source$ sudo mkdir rootfs
linux@ubuntu64-vm:/source$ ls
rootfs rootfs.tar.xz
linux@ubuntu64-vm:/source$ sudo cp /home/linux/busybox/busybox-1.22.1/_install/*
/source/rootfs -a
linux@ubuntu64-vm:/source$ cd rootfs/
linux@ubuntu64-vm:/source/rootfs$ ls
bin dev etc lib linuxrc mnt proc root sbin sys tmp usr var
linux@ubuntu64-vm:/source/rootfs$
```

```
COM3 - PuTTY
[ 1.775000] mmcblk1: mmc0:e624 SS08G 7.40 GiB
[ 1.790000] mmcblk1: p1
[ 1.890000] usb 1-3: new high-speed USB device number 2 using exynos-ehci
[ 2.010000] IP-Config: Guessing netmask 255.255.255.0
[ 2.010000] IP-Config: Complete:
[ 2.015000] device=eth0, hwaddr=00:0a:2d:a6:55:a2, ipaddr=192.168.100.19
1, mask=255.255.255.0, gw=255.255.255.255
[ 2.025000] dm9000 5000000.ethernet eth0: link up, 100Mbps, full-duplex, lpa
0xCDE1
[ 2.030000] host=192.168.100.191, domain=, nis-domain=(none)
[ 2.030000] bootserver=255.255.255.255, rootserver=192.168.100.192, root
path=
[ 2.030000] clk: Not disabling unused clocks
[ 2.050000] hub 1-3:1.0: USB hub found
[ 2.055000] hub 1-3:1.0: 3 ports detected
[ 2.210000] VFS: Mounted root (nfs filesystem) on device 0:10.
[ 2.215000] devtmpfs: mounted
[ 2.215000] Freeing unused kernel memory: 228K (c052d000 - c0566000)

Please press Enter to activate this console.
[root@farsight]#ls
bin      etc      linuxrc  proc      sbin      tmp      var
dev      lib      mnt      root      sys      usr
[root@farsight]#
```

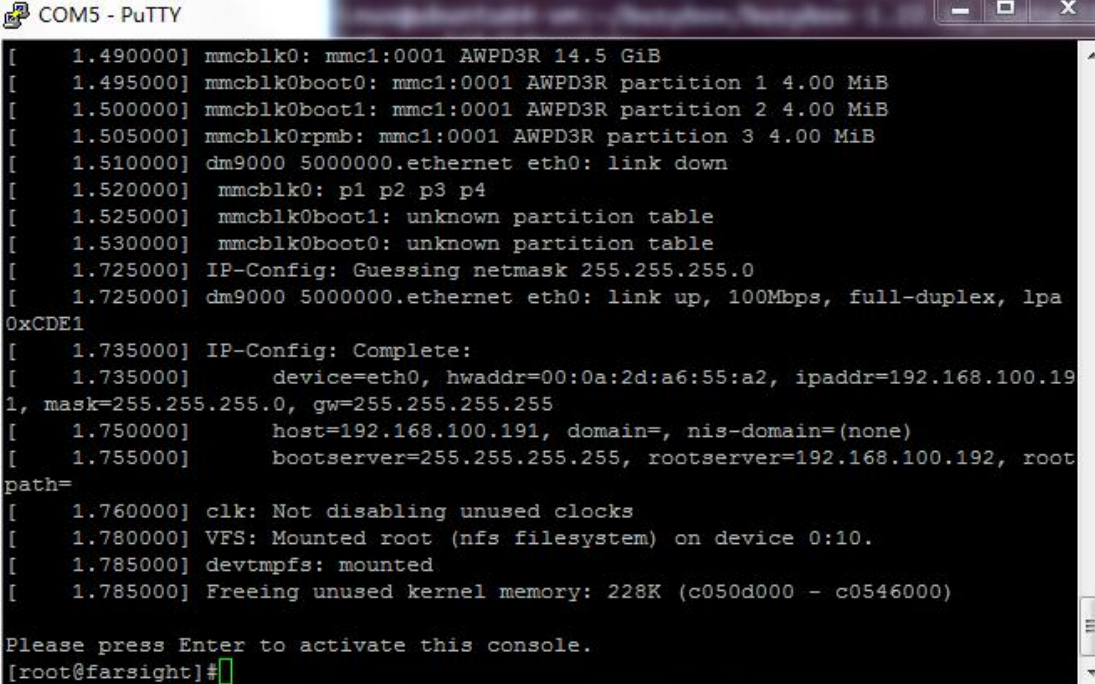
删除原先的/source/rootfs:

\$ sudo rm -rf /source/rootfs

将我们新建的根文件系统拷贝到/source/rootfs 目录下

```
$sudo mkdir /source/rootfs
```

```
$ sudo cp _install/* /source/rootfs -a
```



```
COM5 - PuTTY
[ 1.490000] mmcblk0: mmc1:0001 AWPDP3R 14.5 GiB
[ 1.495000] mmcblk0boot0: mmc1:0001 AWPDP3R partition 1 4.00 MiB
[ 1.500000] mmcblk0boot1: mmc1:0001 AWPDP3R partition 2 4.00 MiB
[ 1.505000] mmcblk0rpmb: mmc1:0001 AWPDP3R partition 3 4.00 MiB
[ 1.510000] dm9000 5000000.ethernet eth0: link down
[ 1.520000] mmcblk0: p1 p2 p3 p4
[ 1.525000] mmcblk0boot1: unknown partition table
[ 1.530000] mmcblk0boot0: unknown partition table
[ 1.725000] IP-Config: Guessing netmask 255.255.255.0
[ 1.725000] dm9000 5000000.ethernet eth0: link up, 100Mbps, full-duplex, lpa
0xCDE1
[ 1.735000] IP-Config: Complete:
[ 1.735000] device=eth0, hwaddr=00:0a:2d:a6:55:a2, ipaddr=192.168.100.19
1, mask=255.255.255.0, gw=255.255.255.255
[ 1.750000] host=192.168.100.191, domain=, nis-domain=(none)
[ 1.755000] bootserver=255.255.255.255, rootserver=192.168.100.192, root
path=
[ 1.760000] clk: Not disabling unused clocks
[ 1.780000] VFS: Mounted root (nfs filesystem) on device 0:10.
[ 1.785000] devtmpfs: mounted
[ 1.785000] Freeing unused kernel memory: 228K (c050d000 - c0546000)

Please press Enter to activate this console.
[root@farsight]#
```

重新启动开发板，能够正常挂载，功能正常。

2.Ramdisk 文件系统制作实验

1、制作一个大小为 8M 的镜像文件

```
$ cd ~
```

```
$ dd if=/dev/zero of=ramdisk bs=1k count=8192 (ramdisk 为 8M)
```



```

linux@ubuntu64-vm:~/busybox/busybox-1.22.1/_install/lib$ cd ~
linux@ubuntu64-vm:~$ dd if=/dev/zero of=ramdisk bs=1k count=8192
记录了8192+0 的读入
记录了8192+0 的写出
8388608字节(8.4 MB)已复制, 0.0542834 秒, 155 MB/秒
linux@ubuntu64-vm:~$ mkfs.ext2 -F ramdisk
mke2fs 1.42 (29-Nov-2011)
Discarding device blocks: 完成
文件系统标签=
OS type: Linux
块大小=1024 (log=0)
分块大小=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
2048 inodes, 8192 blocks
409 blocks (4.99%) reserved for the super user
第一个数据块=1
Maximum filesystem blocks=8388608
1 block group
8192 blocks per group, 8192 fragments per group
2048 inodes per group

Allocating group tables: 完成
正在写入inode表: 完成
Writing superblocks and filesystem accounting information: 完成

```

2、格式化这个镜像文件为 ext2

```
$ mkfs.ext2 -F ramdisk
```

3、在 mount 下面创建 initrd 目录作为挂载点

```
$ sudo mkdir /mnt/initrd
```

4、将这个磁盘镜像文件挂载到/mnt/initrd 下

注意这里的 ramdisk 不能存放在 rootfs 目录中

```
$ sudo mount -t ext2 -o loop ramdisk /mnt/initrd
```

5、将我们的文件系统复制到 initrd.img 中

将测试好的文件系统里的内容全部拷贝到 /mnt/initrd 目录下面

```
$ sudo cp /source/rootfs/* /mnt/initrd -a
```

6、卸载 initrd

```
$ sudo umount /mnt/initrd
```

7、压缩 initrd.img 为 initrd.img.gz 并拷贝到/tftpboot 下

```
$ gzip --best -c ramdisk > ramdisk.gz
```

8、格式化为 uboot 识别的格式

```
$ mkimage -n "ramdisk" -A arm -O linux -T ramdisk -C gzip -d ramdisk.gz
```

```
ramdisk.img
```

```
$ cp ramdisk.img /tftpboot
```

```

linux@ubuntu64-vm:~$ sudo mkdir /mnt/initrd
linux@ubuntu64-vm:~$ sudo mount -t ext2 -o loop ramdisk /mnt/initrd
linux@ubuntu64-vm:~$ sudo cp /source/rootfs/* /mnt/initrd -a
linux@ubuntu64-vm:~$ sudo umount /mnt/initrd
linux@ubuntu64-vm:~$ gzip --best -c ramdisk > ramdisk.gz
linux@ubuntu64-vm:~$ mkimage -n "ramdisk" -A arm -O linux -T ramdisk -C gzip -
ramdisk.gz ramdisk.img
Image Name:      ramdisk
Created:         Thu Oct 27 20:22:45 2016
Image Type:      ARM Linux RAMDisk Image (gzip compressed)
Data Size:       2523671 Bytes = 2464.52 kB = 2.41 MB
Load Address:    0x00000000
Entry Point:     0x00000000
linux@ubuntu64-vm:~$ cp ramdisk.img /tftpboot
linux@ubuntu64-vm:~$ cd ~/kernel/linux-3.14
linux@ubuntu64-vm:~/kernel/linux-3.14$ make menuconfig
scripts/kconfig/mconf Kconfig

*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.

linux@ubuntu64-vm:~/kernel/linux-3.14$ make uImage
CHK      include/config/kernel.release
CHK      include/generated/uapi/linux/version.h
CHK      include/generated/utsrelease.h
make[1]: "include/generated/mach-types.h"是最新的。
CALL     scripts/checksyscalls.sh
CHK      include/generated/compile.h
Kernel: arch/arm/boot/Image is ready
Kernel: arch/arm/boot/zImage is ready
Image arch/arm/boot/uImage is ready

```

9、配置内核支持 RAMDISK

制作完 ramdisk.img 后，需要配置内核支持 RAMDISK 作为启动文件系统，修改内核配置重新编译内核，复制到/tftpboot

10、在 U-BOOT 命令行重新设置启动参数：

```
# setenv bootcmd tftp 41000000 uImage\;tftp 42000000 exynos4412-fs4412.dtb\;tftp
43000000
```

```
ramdisk.img\;bootm 41000000 43000000 42000000
```

```
# saveenv
```

重新启动开发板查看能否正常启动，结果如下。

```
COM3 - PuTTY
191 init=/linuxrc console=ttySAC2,115200
bootcmd=tftp 41000000 uImage; tftp 42000000 exynos4412-fs4412.dtb; bootm 41000000
0 - 42000000
bootdelay=3
ethact=dm9000
ethaddr=11:22:33:44:55:66
fileaddr=42000000
filesize=8454
gatewayip=192.168.100.1
ipaddr=192.168.100.191
netmask=255.255.255.0
server=192.168.100.192
serverip=192.168.100.192
stderr=serial
stdin=serial
stdout=serial

Environment size: 505/16380 bytes
FS4412 # setenv bootcmd tftp 41000000 uImage\; tftp 42000000 exynos4412-fs4412.d
tb\;tftp 43000000 ramdisk.img\;bootm 41000000 43000000 42000000
FS4412 # saveenv
Saving Environment to MMC...
Writing to MMC(0)... done
FS4412 #
```

```
COM3 - PuTTY
[ 1.590000] mmcbldk0boot1: unknown partition table
[ 1.590000] mmcbldk0boot0: unknown partition table
[ 1.665000] dm9000 5000000.ethernet eth0: link down
[ 1.800000] mmc0: new high speed SDHC card at address e624
[ 1.805000] mmcbldk1: mmc0:e624 SS08G 7.40 GiB
[ 1.820000] mmcbldk1: p1
[ 1.890000] IP-Config: Guessing netmask 255.255.255.0
[ 1.890000] usb 1-3: new high-speed USB device number 2 using exynos-ehci
[ 1.895000] dm9000 5000000.ethernet eth0: link up, 100Mbps, full-duplex, lpa
0xCDE1
[ 1.905000] IP-Config: Complete:
[ 1.910000] device=eth0, hwaddr=00:0a:2d:a6:55:a2, ipaddr=192.168.100.19
1, mask=255.255.255.0, gw=255.255.255.255
[ 1.920000] host=192.168.100.191, domain=, nis-domain=(none)
[ 1.925000] bootserver=255.255.255.255, rootserver=192.168.100.192, root
path=
[ 1.935000] clk: Not disabling unused clocks
[ 1.940000] RAMDISK: gzip image found at block 0
[ 2.020000] hub 1-3:1.0: USB hub found
[ 2.020000] hub 1-3:1.0: 3 ports detected
[ 2.205000] VFS: Mounted root (ext2 filesystem) on device 1:0.

Please press Enter to activate this console.
[root@farsight]#
```

六、实验结果及分析：

第三次实验的成功，为这次实验打下了坚实的基础。这次实验比较顺利，我们通过不断的努力准确地得到了和说明文档上相类似的实验成果。

本次实验的主要内容是熟悉 Linux 文件系统目录结构，创建自己的文件系统，并通过 NFS 文件系统挂载的方式启动开发来测试自制的文件系统。通过这

次实验，我对 Linux 文件系统目录结构更加了解，对这些目录的功能更加熟悉。其中一些目录的功能如下所示：

/bin: bin 是 binary 的缩写。这个目录沿袭了 UNIX 系统的结构，存放着使用者最经常使用的命令。例如 cp、ls、cat，等等。

/boot:这里存放的是启动 Linux 时使用的一些核心文件。

/dev:dev 是 device（设备）的缩写。这个目录下是所有 Linux 的外部设备，其功能类似 DOS 下的.sys 和 Win 下的.vxd。在 Linux 中设备和文件是用同种方法访问的。例如：/dev/hda 代表第一个物理 IDE 硬盘。

/etc:这个目录用来存放系统管理所需要的配置文件和子目录。

/home:用户的主目录，比如说有个用户叫 wang，那他的主目录就是/home/wang 也可以用~wang 表示。

/lib:这个目录里存放着系统最基本的动态链接共享库，其作用类似于 Windows 里的.dll 文件。几乎所有的应用程序都须要用到这些共享库。

/mnt:这个目录是空的，系统提供这个目录是让用户临时挂载别的文件系统。

/proc:这个目录是一个虚拟的目录，它是系统内存的映射，我们可以通过直接访问这个目录来获取系统信息。也就是说，这个目录的内容不在硬盘上而是在内存里。

/root:系统管理员（也叫超级用户）的主目录。作为系统的拥有者，总要有些特权啊！比如单独拥有一个目录。

/sbin:s 就是 Super User 的意思，也就是说这里存放的是系统管理员使用的管理程序。

/tmp:这个目录不用说，一定是用来存放一些临时文件的地方了。

/var:这个目录中存放着那些不断在扩充着的东西，为了保持/usr 的相对稳定，那些经常被修改的目录可以放在这个目录下，实际上许多系统管理员都是这样干的。顺带说一下系统的日志文件就在/var/log 目录中。

/usr:这是最庞大的目录，我们要用到的应用程序和文件几乎都存放在这个目录下。其中包含以下子目录；

七、实验结论、问题及改进建议：

实验结论：通过这次 Linux 文件系统构建实验，我深入了解 Linux 的文件系统结构，并对自制一个文件系统的过程有了一定了解。

实验问题：这次实验比较顺利，没有遇到太大的问题，主要的问题是修改配置文件设置不对，导致实验不成功，重新修改配置文件，重新编译内涵即可完成实验。

通过这次实验我体会到耐心和细节都是决定成功的不可或缺的因素。我们需要努力学习嵌入式知识，要有耐力，打好基础，同时注重细节，严于律己，才能更好地掌握和深入嵌入式系统开发。