

《操作系统原理与设计》实验报告

姓 名：邹易澄
学 号：PB14011009
学 院：计算机科学与技术学院
实验编号：03
实验名称：制作启动硬盘并启动一个操作系统映像
日 期：2016年5月2号

实验目的：

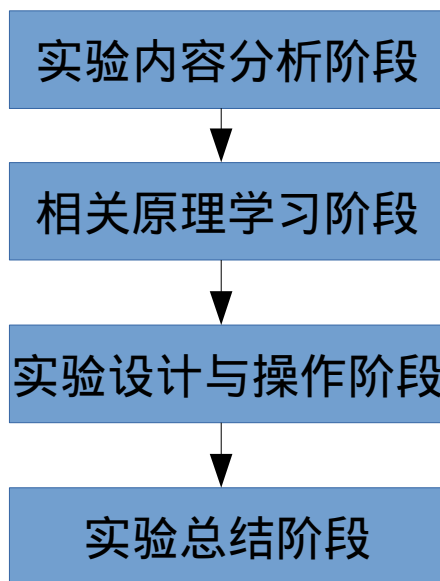
1. 理解 linux 下磁盘与文件系统管理的原理。理解 linux 系统中分区与挂载的概念。
2. 理解操作系统启动的原理，熟悉操作系统启动的过程。
3. 学会制作简单的操作系统启动硬盘，并启动一个简单的操作系统映像。

实验内容：

1. 下载 grub-0.97.tag.gz 并编译，或者直接用编译好的 grub-0.97-i386-pc。
2. 制作 grub 启动软盘，进而制作启动硬盘。
3. 下载一个 RTEMS 操作系统的映像: hello.exe，然后编写 menu.lst，利用 grub 启动之。

实验过程概述：

本次实验分为实验内容分析阶段、相关原理学习阶段、实验设计与操作阶段以及实验总结阶段。
流程图如下：



实验内容分析阶段主要对此实验的流程进行分析，说明实验所在的环境。相关原理学习阶段主要对本次实验操作中所涉及的原理、要用到的各种命令进行说明。实验设计与操作阶段主要描述详细的制作过程。实验总结阶段主要对此次实验遇到的问题和实验收获进行总结。

实验内容分析阶段：

此次实验的流程为：1. 准备 grub-0.97 安装目录和文件；2. 制作 grub 启动软盘；3. 准备磁盘映像；4. 在磁盘映像中添加 grub 启动功能；5. 用制作好的启动硬盘启动 RTEMS 操作系统映像。

此实验在 ubuntu-15.10 操作系统上进行，计算机所用的指令集为 x86_64，属于 CISC 指令集。gcc 版本为 gcc-5.2.1。

相关原理学习阶段：

1.qemu 模拟器：

QEMU 是一套以 GPL 许可证分发源码的模拟处理器，在 GNU/Linux 平台上使用广泛。默认支持多种架构。可以模拟 IA-32(x86)个人电脑，AMD 64 个人电脑，MIPS-R4000，升阳的 SPARKsun3 与 PowerPC 架构。

在此计算机的平台上，QEMU 要选择模拟 x86 的架构，相应的模拟工具为 qemu-system-i386 或 qemu-system-x86_64。

通过 man qemu-system-i386 命令，可以获取 qemu 工具的功能，以下是本实验所要用到的一些命令行参数。

```
Block device options:

-fda file
-fdb file
    Use file as floppy disk 0/1 image.

-hda file
-hdb file
-hdc file
-hdd file
    Use file as hard disk 0, 1, 2 or 3 image.
```

-fda+[file]可以将 file 作为软盘 0 的镜像，-hda+[file]可以将 file 作为硬盘 0 的镜像。

```
-boot [order=drives][,once=drives][,menu=on|off][,splash=sp_name][,splash-time=sp_time][,reboot-timeout=rb_timeout][,strict=on|off]
Specify boot order drives as a string of drive letters. Valid drive letters depend on the target achitecture. The x86 PC uses: a,
b (floppy 1 and 2), c (first hard disk), d (first CD-ROM), n-p (Etherboot from network adapter 1-4), hard disk boot is the
default. To apply a particular boot order only on the first startup, specify it via once.

Interactive boot menus/prompts can be enabled via menu=on as far as firmware/BIOS supports them. The default is non-interactive
boot.

A splash picture could be passed to bios, enabling user to show it as logo, when option splash=sp_name is given and menu=on, If
firmware/BIOS supports them. Currently Seabios for X86 system support it. limitation: The splash file could be a jpeg file or a
BMP file in 24 BPP format(true color). The resolution should be supported by the SVGA mode, so the recommended is 320x240,
640x480, 800x640.

A timeout could be passed to bios, guest will pause for rb_timeout ms when boot failed, then reboot. If rb_timeout is '-1', guest
will not reboot, qemu passes '-1' to bios by default. Currently Seabios for X86 system support it.

Do strict boot via strict=on as far as firmware/BIOS supports it. This only effects when boot priority is changed by bootindex
options. The default is non-strict boot.

# try to boot from network first, then from hard disk
qemu-system-i386 -boot order=nc
# boot from CD-ROM first, switch back to default order after reboot
qemu-system-i386 -boot once=d
# boot with a splash picture for 5 seconds.
qemu-system-i386 -boot menu=on,splash=/root/boot.bmp,splash-time=5000

Note: The legacy format '-boot drives' is still supported but its use is discouraged as it may be removed from future versions.
```

这是-boot 参数的介绍，它可以设置启动设备，a 和 b 分别代表从 1 号软盘和 2 号软盘启动，c 代表从第一个硬盘启动，d 代表从 CD-ROM 中启动，n-p 代表从网络启动。

2. losetup 命令

loop 设备介绍：

在类 UNIX 系统里，loop 设备是一种伪设备(pseudo-device)，或者也可以说是仿真设备。它能使我们像块设备一样访问一个文件。

在使用之前，一个 loop 设备必须要和一个文件进行连接。这种结合方式给用户提供了一个替代块特殊文件的接口。因此，如果这个文件包含有一个完整的文件系统，那么这个文件就可以像一个磁盘设备一样被 mount 起来。

上面说的文件格式，经常是 CD 或 DVD 的 ISO 光盘镜像文件或者是软盘(硬盘)的 img 镜像文件。通过这种 loop mount (回环 mount)的方式，镜像文件就可以被 mount 到当前文件系统的一个目录下。

通过 man losetup 命令，查看 losetup 命令的用法。以下是本实验要用到的一些参数。

```
-d, --detach loopdev...  
Detach the file or device associated with the specified loop device(s).
```

```
-o, --offset offset  
The data start is moved offset bytes into the specified file or device.
```

```
/dev/loop[0..N]  
loop block devices
```

Set up a loop device:

```
losetup [-o offset] [--sizelimit size]  
[-Pr] [--show] -f|loopdev file
```

其中，-d 是卸载设备，-o 可以设置数据偏移量，在 linux 系统中有 loop0 ~ loop7 共 8 个 loop 设备，可利用这些设备将文件虚拟成块设备，将其视为硬盘驱动器，光驱或软驱，并挂入当作目录来使用。

3. fdisk 命令

fdisk 是一个在 Linux 下进行磁盘管理的工具，方便易用。进入 fdisk 选择界面后，可以输入 m 来获取帮助。

```
Help:

DOS (MBR)
a  toggle a bootable flag
b  edit nested BSD disklabel
c  toggle the dos compatibility flag

Generic
d  delete a partition
l  list known partition types
n  add a new partition
p  print the partition table
t  change a partition type
v  verify the partition table

Misc
m  print this menu
u  change display/entry units
x  extra functionality (experts only)

Script
I  load disk layout from sfdisk script file
O  dump disk layout to sfdisk script file

Save & Exit
w  write table to disk and exit
q  quit without saving changes

Create a new label
g  create a new empty GPT partition table
G  create a new empty SGI (IRIX) partition table
o  create a new empty DOS partition table
s  create a new empty Sun partition table
```

在此实验中，需要用到的命令有 n、a 和 w，n 表示新增一个 partition 分区，a 表示设置引导扇区，w 表示保存修改并退出。

4. 磁盘的格式化命令

常见的磁盘格式化命令是 mkfs，在 linux 系统中有一些快速格式化命令。

```
rowitzou@rowitzou-Lenovo-G50-70:~/Lab$ mkfs
mkfs          mkfs.cramfs  mkfs.ext3     mkfs.ext4dev  mkfs.minix    mkfs.ntfs
mkfs.bfs      mkfs.ext2    mkfs.ext4     mkfs.fat      mkfs.msdos    mkfs.vfat
```

例如，如果要将/dev/hda1 格式化为 ext4 文件系统格式，可以用命令 mkfs.ext4 /dev/hda1 直接将其格式化。

5. 分区的 mount 与 umount 命令

通过 man mount 与 man umount 可以得知以上两个命令的使用方法。这两个命令是用来挂载和卸载设备的。linux 系统中，挂载点是目录，而这个目录是进入磁盘分区槽(即文件系统)的入口。

以下是 mount 命令的常用选项与参数：

选项与参数：

-a ：依照配置文件 `/etc/fstab` 的数据将所有未挂载的磁盘都挂载上来

-l ：单纯的输入 mount 会显示目前挂载的信息。加上 -l 可增列 Label 名称！

-t ：与 `mkfs` 的选项非常类似的，可以加上文件系统种类来指定欲挂载的类型。

常见的 Linux 支持类型有：ext2, ext3, vfat, reiserfs, iso9660(光盘格式),
nfs, cifs, smbfs(此三种为网络文件系统类型)

-n ：在默认的情况下，系统会将实际挂载的情况实时写入 `/etc/mtab` 中，以利
其他程序

的运作。但在某些情况下(例如单人维护模式)为了避免问题，会刻意不写入。
此时就得要使用这个 -n 的选项了。

-L ：系统除了利用装置文件名(例如 `/dev/hdc6`)之外，还可以利用文件系统的
标头名称

(Label)来进行挂载。最好为你的文件系统取一个独一无二的名称吧！

-o ：后面可以接一些挂载时额外加上的参数！比方说账号、密码、读写权限等：

ro, rw: 挂载文件系统成为只读(ro) 或可擦写(rw)

async, sync: 此文件系统是否使用同步写入(sync) 或异步(async) 的
内存机制，请参考 [文件系统运作方式](#)。预设为 async。

auto, noauto: 允许此 partition 被以 mount -a 自动挂载(auto)

dev, nodev: 是否允许此 partition 上，可建立装置档案？ dev 为可允许

suid, nosuid: 是否允许此 partition 含有 suid/sgid 的文件格式？

exec, noexec: 是否允许此 partition 上拥有可执行 binary 档案？

user, nouser: 是否允许此 partition 让任何使用者执行 mount？一般来说，
mount 仅有 root 可以进行，但下达 user 参数，则可以让
一般 user 也能够对此 partition 进行 mount。

defaults: 默认值为：rw, suid, dev, exec, auto, nouser, and async

remount: 重新挂载，这在系统出错，或重新更新参数时，很有用！

6. 操作系统启动的原理

第一阶段：BIOS

上个世纪 70 年代初，“只读内存”（read-only memory，缩写为 ROM）发明，开机程序被刷入 ROM 芯片，计算机通电后，第一件事就是读取它。

这块芯片里的程序叫做“基本输出输入系统”（Basic Input/Output System），简称为 BIOS。

首先，BIOS 要进行硬件自检，硬件自检完成后，BIOS 把控制权转交给下一阶段的启动程序。

第二阶段：主引导记录

BIOS 按照“启动顺序”，把控制权转交给排在第一位的储存设备。即根据用户指定的引导顺序从软盘、硬盘或是可移动设备中读取启动设备的 MBR，并放入指定的位置（0x7c000）内存中。

这时，计算机读取该设备的第一个扇区，也就是读取最前面的 512 个字节。如果这 512 个字节的最后两个字节是 0x55 和 0xAA，表明这个设备可以用于启动；如果不是，表明设备不能用于启动，控制权于是被转交给“启动顺序”中的下一个设备。这最前面的 512 个字节，就叫做“主引导记录”（Master boot record，缩写为 MBR）。MBR 负责将控制权交给指定的系统分区。

第三阶段：硬盘启动

这时，计算机的控制权就要转交给硬盘的某个分区了，这里又分成三种情况。包括卷引导记录、扩展分区与逻辑分区，以及启动管理器。在本实验中，grub 引导属于第三种情况。

对于 grub 而言，在 MBR 中的 446 字节的引导程序属于 GRUB 的开始执行程序，通过这段程序，进一步执行 stage1.5 或是 stage2 的执行程序。

其中 stage1.5 或是 stage2 便属于阶段 2 引导的过程了，stage2 过程也是作为 GRUB kernel 的核心代码出现。Stage1.5 过程的功能很单一，主要就是为了引导 stage2 过程服务。stage2 过程中，主要会把系统切换到保护模式，设置好 C 运行时环境，找到 config 文件（事实上就是 menu.lst 文件），如果没有找到就执行一个 shell，等待用户的执行。然后的工作就变成了输入命令->解析命令->执行命令的循环中。当然该阶段引导的最终状态就是执行 boot 命令，将内核和 initrd 镜像加载进入内存中，进而将控制权转交给内核。

第四阶段：操作系统

控制权转交给操作系统后，操作系统的内核被载入内存。

在本次实验中，RTEMS 操作系统映像只需提供一个内核，并不需要 initrd 镜像文件。因此，到此为止，全部启动过程完成。

实验设计与操作阶段：

1. 准备 grub-0.97 安装目录和文件。

在本实验中，我采用已经编译好的 grub-0.97-i386-pc，里面包含了 grub 引导必要的启动文件。或者，可以尝试自己编译 grub-0.97。

（1）下载 grub-0.97.tar.gz。

（2）解压缩。

```
rowitzou@rowitzou-Lenovo-G50-70:~/lab$ tar xvf grub-0.97.tar.gz
```

（3）解压缩后进入 grub-0.97 目录，尝试编译。



grub-0.97



grub-0.97-i386-pc



grub-0.97.tar.gz

根据 install 安装说明文件，输入 ./configure 命令，出现错误：error : C compiler cannot create executables。通过查阅 config.log 文件，发现以下错误。

```

configure:2421: checking for C compiler default output file name
configure:2424: gcc -m32  conftest.c >&5
/usr/bin/ld: cannot find crt1.o: No such file or directory
/usr/bin/ld: cannot find crti.o: No such file or directory
/usr/bin/ld: skipping incompatible /usr/lib/gcc/x86_64-linux-gnu/5/libgcc.a when searching for -lgcc
/usr/bin/ld: cannot find -lgcc
/usr/bin/ld: skipping incompatible /usr/lib/gcc/x86_64-linux-gnu/5/libgcc_s.so when searching for -lgcc_s
/usr/bin/ld: cannot find -lgcc_s
/usr/bin/ld: cannot find -lc
/usr/bin/ld: skipping incompatible /usr/lib/gcc/x86_64-linux-gnu/5/libgcc.a when searching for -lgcc
/usr/bin/ld: cannot find -lgcc
/usr/bin/ld: skipping incompatible /usr/lib/gcc/x86_64-linux-gnu/5/libgcc_s.so when searching for -lgcc_s
/usr/bin/ld: cannot find -lgcc_s
/usr/bin/ld: cannot find crtn.o: No such file or directory
collect2: error: ld returned 1 exit status
configure:2427: $? = 1
configure: failed program was:
| /* confdefs.h.  */
|
| #define PACKAGE_NAME "GRUB"
| #define PACKAGE_TARNAME "grub"
| #define PACKAGE_VERSION "0.97"
| #define PACKAGE_STRING "GRUB 0.97"
| #define PACKAGE_BUGREPORT "bug-grub@gnu.org"
| #define PACKAGE "grub"
| #define VERSION "0.97"
| /* end confdefs.h.  */
|
| int
| main ()
| {
|
| ;
| return 0;
| }
configure:2466: error: C compiler cannot create executables

```

通过在 Stack Overflow 网站上询问原因，得知是少安装了必要的编译工具。之后，我安装了 gcc-multilib 工具。

```

rowitzou@rowitzou-Lenovo-G50-70:~/lab/grub-0.97$ sudo apt-get install gcc-multilib

```


再次尝试编译，又出现以下错误：

```
rowitzou@rowitzou-Lenovo-G50-70:~/lab/grub-0.97$ ./configure
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking whether to enable maintainer-specific portions of Makefiles... no
checking for gcc... gcc
checking for gcc... (cached) gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ANSI C... none needed
checking for style of include used by make... GNU
checking dependency style of gcc... gcc3
checking dependency style of gcc... (cached) gcc3
checking for ranlib... ranlib
checking whether optimization for size works... yes
checking whether gcc has -fno-stack-protector... yes
checking whether -Wundef works... yes
checking whether -falign-loops works... yes
checking for objcopy... objcopy
checking if C symbols get an underscore after compilation... no
checking whether objcopy works for absolute addresses... no
configure: error: GRUB requires a working absolute objcopy; upgrade your binutil
s
```

通过在网上查阅相关资料，得知是因为 grub-0.97 版本过低，可以尝试使用 gcc 的较低版本进行编译，或者通过改动 configure 脚本里面的一条语句来使之适应最新的 gcc 编译器。

原来的 configure 脚本有下面这一条语句：

```
if { ac_try='${OBJCOPY-objcopy} -O binary conftest.exec conftest' |
```

将其改为如下语句：

```
if { ac_try='${OBJCOPY-objcopy} -R .note.gnu.build-id -O binary conftest.exec conftest'
```

再次输入./configure 命令，这次就顺利完成。最后，在终端下输入 make 命令，完成编译。

2. 制作 grub 启动软盘。

(1) 首先，安装 qemu 模拟器。

```
rowitzou@rowitzou-Lenovo-G50-70:~/lab$ sudo apt-get install qemu
```

(2) 其次建立大小为 1.5M 的 boot.img 文件。

```
rowitzou@rowitzou-Lenovo-G50-70:~/lab$ dd if=/dev/zero of=boot.img bs=512 count=
2880
记录了2880+0 的读入
记录了2880+0 的写出
1474560字节(1.5 MB)已复制，0.00434558 秒，339 MB/秒
```

(3) 先将 boot.img 设置成虚拟软盘映像，然后在软盘映像中添加 grub 引导功能。其中 stage1 大小为 512Byte，作为 MBR，stage2 作为 grub 最核心的管理部分。

```
rowitzou@rowitzou-Lenovo-G50-70:~/lab$ sudo losetup /dev/loop0 boot.img
rowitzou@rowitzou-Lenovo-G50-70:~/lab$ sudo dd if=./grub-0.97-i386-pc/boot/grub/
stage1 of=/dev/loop0 bs=512 count=1
记录了1+0 的读入
记录了1+0 的写出
512字节(512 B)已复制，0.0126571 秒，40.5 kB/秒
rowitzou@rowitzou-Lenovo-G50-70:~/lab$ sudo dd if=./grub-0.97-i386-pc/boot/grub/
stage2 of=/dev/loop0 bs=512 seek=1
记录了197+1 的读入
记录了197+1 的写出
101138字节(101 kB)已复制，0.0219997 秒，4.6 MB/秒
rowitzou@rowitzou-Lenovo-G50-70:~/lab$ sudo losetup -d /dev/loop0
```

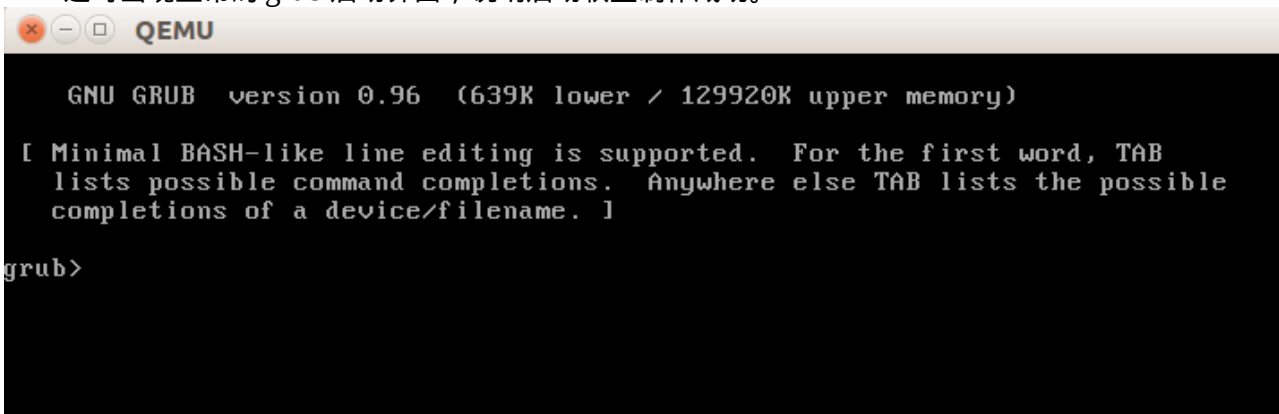
(4) 测试是否能进入 grub 引导界面。先查看 qemu 的各个命令，选择适合于本机的指令系统的命令。

```
rowitzou@rowitzou-Lenovo-G50-70:~/lab$ qemu-
qemu-aarch64          qemu-ppc64          qemu-system-mipsel
qemu-alpha            qemu-ppc64abi32     qemu-system-moxie
qemu-arm              qemu-ppc64le        qemu-system-or32
qemu-armeb            qemu-s390x           qemu-system-ppc
qemu-cris              qemu-sh4             qemu-system-ppc64
qemu-i386              qemu-sh4eb           qemu-system-ppc64le
qemu-img              qemu-sparc           qemu-system-ppcemb
qemu-io                qemu-sparc32plus     qemu-system-s390x
qemu-m68k              qemu-sparc64         qemu-system-sh4
qemu-make-debian-root qemu-system-aarch64  qemu-system-sh4eb
qemu-microblaze        qemu-system-alpha    qemu-system-sparc
qemu-microblazeel      qemu-system-arm      qemu-system-sparc64
qemu-mips              qemu-system-cris     qemu-system-tricore
qemu-mips64            qemu-system-i386     qemu-system-unicore32
qemu-mips64el          qemu-system-lm32     qemu-system-x86_64
qemu-mipsel            qemu-system-m68k     qemu-system-xtensa
qemu-mipsn32           qemu-system-microblaze qemu-system-xtensaeb
qemu-mipsn32el         qemu-system-microblazeel qemu-unicore32
qemu-nbd               qemu-system-mips     qemu-x86_64
qemu-or32              qemu-system-mips64
qemu-ppc               qemu-system-mips64el
```

对于此实验，我选择 qemu-system-i386 工具，下面进行 grub 启动测试。在终端输入以下命令：

```
rowitzou@rowitzou-Lenovo-G50-70:~/lab$ qemu-system-i386 -fda boot.img
```

这时出现正常的 grub 启动界面，说明启动软盘制作成功。



```
GNU GRUB version 0.96 (639K lower / 129920K upper memory)

[ Minimal BASH-like line editing is supported. For the first word, TAB
lists possible command completions. Anywhere else TAB lists the possible
completions of a device/filename. ]

grub>
```


3. 准备磁盘映像。

(1) 首先，初始化一个 32M 大小的文件。

```
rowitzou@rowitzou-Lenovo-G50-70:~/lab$ dd if=/dev/zero of=RTEMS.img bs=4096 count=8192
记录了8192+0 的读入
记录了8192+0 的写出
33554432字节(34 MB)已复制, 0.0251457 秒, 1.3 GB/秒
```

(2) 对 RTEMS.img 进行分区，只将其分为 1 个区，并设置为引导扇区。

```
rowitzou@rowitzou-Lenovo-G50-70:~/lab$ fdisk RTEMS.img

Welcome to fdisk (util-linux 2.26.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x10eb8b6e.

命令(输入 m 获取帮助) : n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
分区号 (1-4, default 1): 1
First sector (2048-65535, default 2048): 2048
Last sector, +sectors or +size[K,M,G,T,P] (2048-65535, default 65535): 65535

Created a new partition 1 of type 'Linux' and of size 31 MiB.

命令(输入 m 获取帮助) : a
Selected partition 1
The bootable flag on partition 1 is enabled now.

命令(输入 m 获取帮助) : w
The partition table has been altered.
Syncing disks.
```

(3) 格式化分区。

把前面的 2048 个扇区 (0~2047) 作为引导扇区使用，格式化分区从第 2048 个扇区开始。其中， $1048576=2048*512$ ，将 RTEMS.img 设置成虚拟硬盘映像。

```
rowitzou@rowitzou-Lenovo-G50-70:~/lab$ sudo losetup -o 1048576 /dev/loop0 RTEMS.img
```

将其格式化为 ext3 文件系统，RTEMS 操作系统可以识别 ext3。

```
rowitzou@rowitzou-Lenovo-G50-70:~/lab$ sudo mkfs.ext3 /dev/loop0
mke2fs 1.42.12 (29-Aug-2014)
Discarding device blocks: 完成
Creating filesystem with 31744 1k blocks and 7936 inodes
Filesystem UUID: e1e74820-88e6-4fbb-bc8d-8c2ab1292563
Superblock backups stored on blocks:
    8193, 24577

Allocating group tables: 完成
正在写入inode表: 完成
Creating journal (1024 blocks): 完成
Writing superblocks and filesystem accounting information: 完成
```

(4) 创建一个新的目录./boot_RTEMS，并挂载活动分区。

```
rowitzou@rowitzou-Lenovo-G50-70:~/lab$ mkdir boot_RTEMS
```

```
rowitzou@rowitzou-Lenovo-G50-70:~/lab$ sudo mount /dev/loop0 boot_RTEMS/
```

(5) 将 hello.exe 操作系统映像复制到 boot_RTEMS 目录下。

```
rowitzou@rowitzou-Lenovo-G50-70:~/lab$ sudo cp hello.exe ./boot_RTEMS
```

4. 在已经制作好的磁盘映像中添加 grub 功能。

(1) 在 boot_RTEMS 目录中创建 boot/grub 目录来存放 grub。

```
rowitzou@rowitzou-Lenovo-G50-70:~/lab$ sudo mkdir ./boot_RTEMS/boot
rowitzou@rowitzou-Lenovo-G50-70:~/lab$ sudo mkdir ./boot_RTEMS/boot/grub
rowitzou@rowitzou-Lenovo-G50-70:~/lab$ sudo cp ./grub-0.97-i386-pc/boot/grub/* boot_RTEMS/boot/grub
```

(2) 在 boot_RTEMS/boot/grub 下编写 menu.lst，具有如下内容：

```
default 0
timeout 30
title hello
root (hd0,0)
kernel (hd0,0)/hello.exe
```

default：后跟一个数字，指 grub 的默认启动项。此处设置为 0，表示默认启动的第 0 号的操作系统。

timeout：指定一个超时值，单位为秒，若用户在 grub 等待的超时时间范围内没有任何操作，则启动默认项，此处等待 30 秒后自动启动默认项。

title：指定启动的操作系统菜单项的名称，即在 grub 列表里的名称，此处设置为 hello。

root：指定启动分区，此处设置为(hd0,0)，表示第一块硬盘的第一个分区。

kernel：指定启动的内核的绝对路径和名称，后边跟参数。此处内核路径即为/hello.exe。

(3) 利用已经制作好的 grub 启动软盘，在磁盘映像上添加 grub 功能。

在终端输入如下命令，它表示用 grub 启动软盘启动磁盘映像，进而添加 grub 功能：

```
rowitzou@rowitzou-Lenovo-G50-70:~/lab$ qemu-system-i386 -boot a -fda boot.img -hda RTEMS.img
```

进入 grub 界面后，输入 root (hd0,0)和 setup (hd0)。表示将根分区设置为磁盘的第一个分区。以及将引导文件 stage1 等写入磁盘，其实就是自动安装 grub。

```
QEMU

GNU GRUB  version 0.96  (639K lower / 129920K upper memory)

[ Minimal BASH-like line editing is supported.  For the first word, TAB
  lists possible command completions.  Anywhere else TAB lists the possible
  completions of a device/filename. ]

grub> root (hd0,0)
Filesystem type is ext2fs, partition type 0x83

grub> setup (hd0)
Checking if "/boot/grub/stage1" exists... yes
Checking if "/boot/grub/stage2" exists... yes
Checking if "/boot/grub/e2fs_stage1_5" exists... yes
Running "embed /boot/grub/e2fs_stage1_5 (hd0)"...  16 sectors are embedded.
succeeded
Running "install /boot/grub/stage1 (hd0) (hd0)1+16 p (hd0,0)/boot/grub/stage2
/boot/grub/menu.lst"...  succeeded
Done.

grub>
```

可见，grub 引导功能添加成功。

(4) 卸载磁盘映像。

```
rowitzou@rowitzou-Lenovo-G50-70:~/lab$ sudo umount boot_RTEMS
rowitzou@rowitzou-Lenovo-G50-70:~/lab$ sudo losetup -d /dev/loop0
```

5. 测试从磁盘启动进入 grub 界面，并进入 RTEMS 系统。

在终端输入如下命令：

```
rowitzou@rowitzou-Lenovo-G50-70:~/lab$ qemu-system-i386 -hda RTEMS.img
```

成功进入 grub 引导界面：

```
QEMU

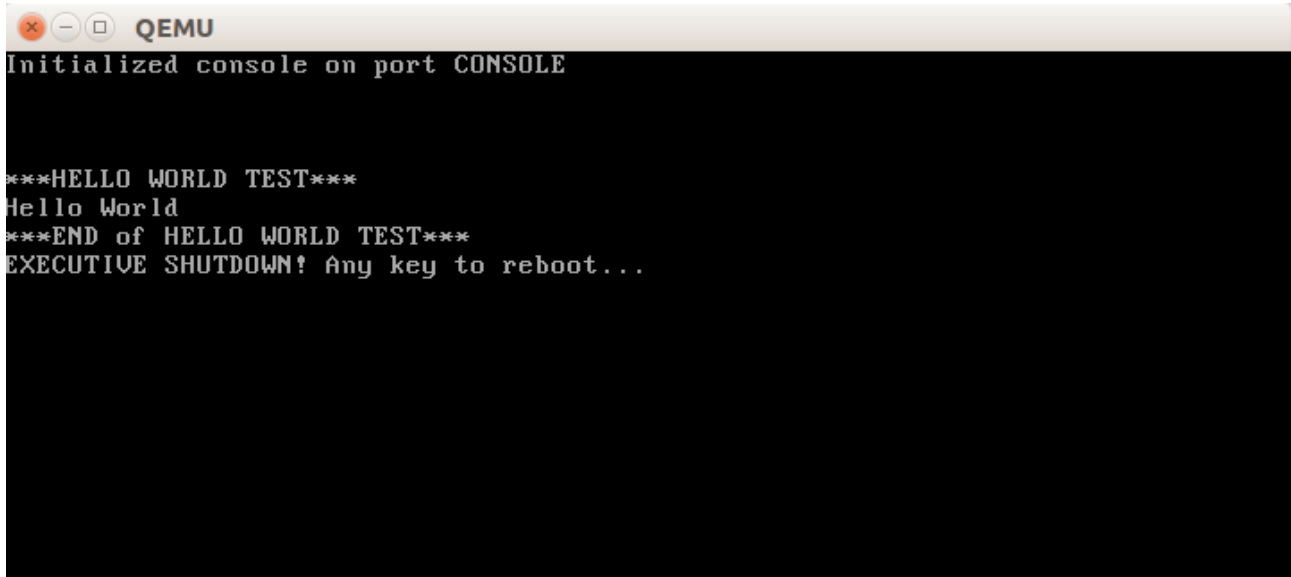
GNU GRUB  version 0.96  (639K lower / 129920K upper memory)

hello

Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the
commands before booting, or 'c' for a command-line.

The highlighted entry will be booted automatically in 12 seconds.
```

在等待 30 秒或者按回车键后进入系统：

A screenshot of a QEMU virtual machine console window. The window has a title bar with the QEMU logo and standard window controls. The console output is as follows:

```
Initialized console on port CONSOLE

***HELLO WORLD TEST***
Hello World
***END of HELLO WORLD TEST***
EXECUTIVE SHUTDOWN! Any key to reboot...
```

至此，启动硬盘制作完毕并成功启动了 RTEMS 操作系统映像。

实验总结阶段：

首先，我要归纳总结一下实验过程中遇到的各种错误。

1. 在尝试编译 grub-0.97 的时候，出现过编译失败的情况，这已经在实验过程中指出，并给出解决方法。在此不再赘述。不过，编译虽然通过了，但 grub 中的 stage1 和 stage2 文件非常大，达到了 100 多 M。而实验提供的已经编译好的 grub-0.97-i386-pc 中的对应文件非常小。就理论上讲，stage1 的大小只有主引导扇区，即 512Byte 的容量。对于这一问题，我上网查过资料，也问过助教，但都没有得到很好地解决。初步分析，可能是因为 grub-0.97 版本过低，编译时出了某些问题。我还打开对比过两个 stage1 文件的区别，发现自己编译的 stage1 文件的前 512Byte 与标准 stage1 内容相同，只是后面无故多了许多内容。鉴于目前尚未学习编译原理，且本实验目的不在于编译，便暂时不再深究这个问题。待今后课程学习完善后，再进一步尝试解决这一问题。在实验中，我使用了已经编译好的 grub-0.97-i386-pc。

2. 在格式化活动分区的时候，我错误地将 `sudo mkfs.ext3 /dev/loop0` 命令输入为 `sudo mkfs.ext3 RTEMS.img`，结果导致 `/dev/loop0` 挂载不了，显示 `/dev/loop0` 是坏的分区。原因是未能成功将 `/dev/loop0` 格式化。通过排查，我检查出了这个错误并予以改正。

3. 在编写 menu.lst 时，我将 menu.lst 放在了主目录下，即连同 hello.exe 放在了一起。在添加硬盘 grub 启动功能时，出现了错误，提示找不到引导文件。后来，我根据提示，检查原因，将 menu.lst 放入了 `./boot/grub` 目录下，解决了这一问题。

最后，实验取得了成功，在这之后，我又尝试过启动 ticker.exe，整个过程大同小异。通过这次实验，我深刻理解了 RTEMS 这个简单操作系统映像启动的整个过程。不过，grub 引导工具中 stage2 的具体功能我还未能完全理解。对于一些复杂的操作系统，比如 linux，它还需要有一个 init 进程随着操作系统内核的载入而运行起来。这取决于不同操作系统的实现。今后，我要尝试阅读操作系统和启动管理器的源码，更深一步了解它们之间的联系和相互作用关系。

参考资料：

http://blog.csdn.net/ustc_dylan/article/details/6878252

<http://blog.csdn.net/langeldep/article/details/8788119>

http://blog.csdn.net/my_emdebed/article/details/1584575

http://blog.sina.com.cn/s/blog_70dd169101013g9j.html

<http://blog.itpub.net/8334342/viewspace-629393/>

<http://blog.csdn.net/baishuwei/article/details/4564451>

[http://wenku.baidu.com/link?](http://wenku.baidu.com/link?url=53otYV1pXqP4INRU0Qp8R1oV5_Kgv4oIgVD7sxUFsvjiZgihiqqM4izqj09_gZYloyEA08nP1ix9OdVuhNcOjSKJwwqsPHb2-fzH6OfV4Qu)

[url=53otYV1pXqP4INRU0Qp8R1oV5_Kgv4oIgVD7sxUFsvjiZgihiqqM4izqj09_gZYloyEA08nP1ix9OdVuhNcOjSKJwwqsPHb2-fzH6OfV4Qu](http://wenku.baidu.com/link?url=53otYV1pXqP4INRU0Qp8R1oV5_Kgv4oIgVD7sxUFsvjiZgihiqqM4izqj09_gZYloyEA08nP1ix9OdVuhNcOjSKJwwqsPHb2-fzH6OfV4Qu)

<http://www.ibm.com/developerworks/cn/linux/l-qemu/>

<http://blog.csdn.net/rosetta/article/details/8687556>

《鸟哥的 linux 私房菜(基础学习篇)》 鸟哥著

《Linux 操作系统分析》 陈香兰