OS:Lab02-制作grub启动软盘

Stu:金泽文 No:PB15111604

**实验目的**：

通过制作grub启动软盘，理解linux启动引导的机制。

**实验内容**：

1. 准备模拟器qemu
2. 制作带grub的磁盘映像
3. 利用qemu和gdb调试linux

**具体过程及相关原理、命令解释**：

1. 准备模拟器qemu

·安装qemu

通过sudo apt-get install qemu安装qemu

通过sudo ln -s /usr/bin/qemu-system-i386 /usr/bin/qemu得到qemu链接以方便访问

·下载linux并编译

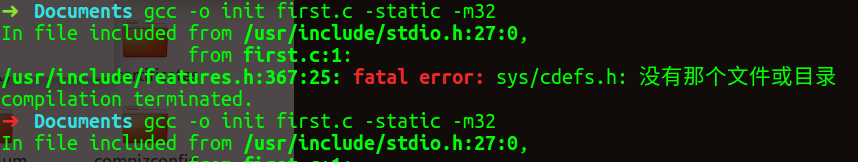
在google中得到3.10.98的下载链接，

由于我下载的格式为xz，tar –xvf出错，google之后使用xz正确解压缩

·制作根目录系统

按照教程，用mknod命令在./dev/目录下为设备cosole和ram分配主设备号与此设备号。（由于linux将设备抽象为文件，故设备号通过修改文件符号标志设备的类型）。

写一个init程序并用-static –m32编译。

 期间，遇到插曲：

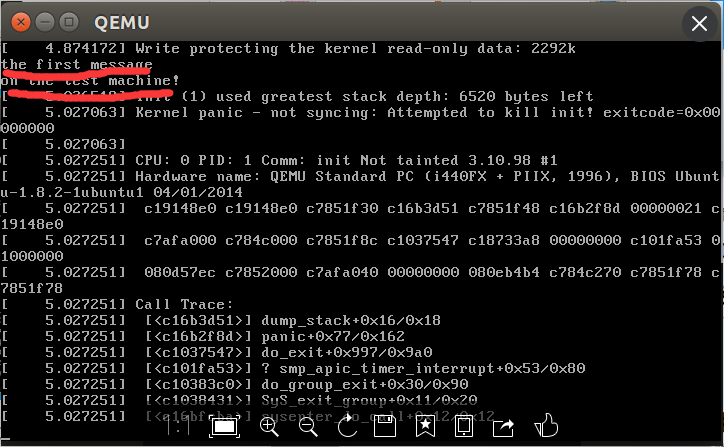
经过google，发现少32位lib，通过sudo apt-get install libc6-dev-i386解决。

通过cpio将文件信息传入到rootfs.img镜像中。所用的格式newc是新兴（SVR4）跨平台格式，支持大于65536i节点的文件系统，正好适用于我们的rootfs。

·运行内核

由于前面助教提醒，这一步基本没有走弯路。

只不过一开始没有理解 PATH\_to\_bzImage是什么意思并且将大写的i看成了小写的L，浪费了十多分钟时间。

这一步得到：

这里解释一下qemu –kernel bzImage –initrd rootfs.img –append”root=/dev/ram rdinit=init”各个指令的含义。

Qemu当然是运行内核的模拟器。

bzImage是压缩的内核映像，bz表示“big zImage”，

initrd则表示Linux的初始Ram磁盘，是在系统引导的过程中挂载的临时根文件系统，所有后面跟的是rootfs.img，

跟着是root = /dev/ram，表示root设备是/dev/ram,

rdinit用来指定ramdisk\_execute\_command，以通知系统执行init\_post(),在我们的例子中，就会首先执行可执行文件init（类似于真是linux系统中的init父进程）。

1. 制作带grub的磁盘映像

·获取grub。

Grub--多系统引导管理器。

·建立软盘映像

dd if=/dev/zero of=a.img bs=512 count=2880

由man dd得到dd的相关信息：convert and copy a file.

If表示read from file 而不是stdin，

of表示write to file 而不是stdout。

bs表示一次读的字节数，为512字节

count表示读的次数。

而/dev/zero负责提供0

可见这个命令只是为了开辟a.img这样大小的空间

·添加grub启动功能

sudo losetup /dev/loop3 a.img

sudo dd if=./grub-0.97-i386-pc/boot/grub/stage1 of=/dev/loop3 bs=512 count=1

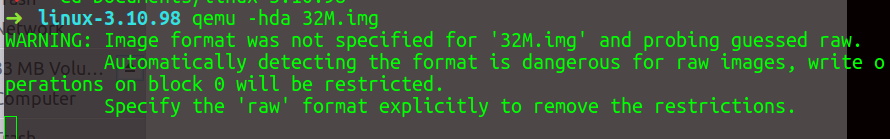
sudo dd if=./grub-0.97-i386-pc/boot/grub/stage2 of=/dev/loop3 bs=512 seek=1

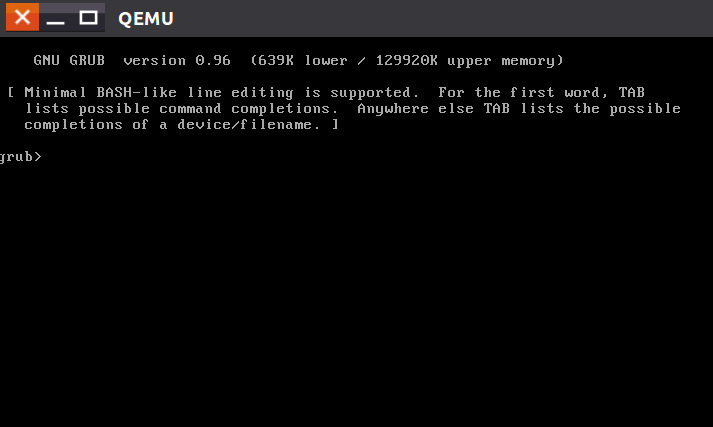
sudo losetup -d /dev/loop

在类unix系统里， loop设备是一种伪设备，能使我们像块设备一样访问文件。

一个loop设备必须和一个文件进行连接，比如这里, /dev/loop3与a.img连接，这个a.img映像文件就像一个磁盘设备一样被挂载（叫loop mount），这里借助dd和losetup的方式，将./grub-0.97-i386-pc/boot/grub/中对应文件信息按块输出到我们的a.img上.

运行qemu –had 32M.img后，





正常。

解释：qemu –had 32M.img 是将32M.img作为软盘0的镜像

·准备磁盘映像

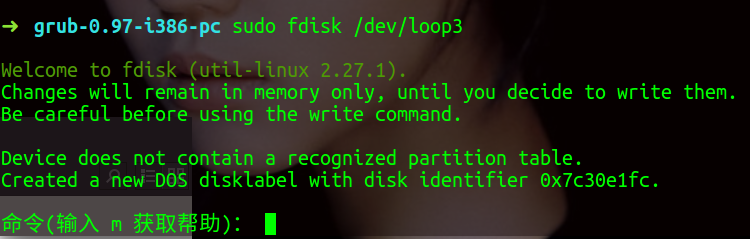
dd if=/dev/zero of=32M.img bs=4096 count=8192

sudo losetup /dev/loop3 32M.img

在磁盘映像上建立一个活动分区

sudo fdisk /dev/loop3 ，

fdisk作为磁盘管理工具，可以用m查看help，经过n，一系列默认之后就建立了一个partition分区，这一步其实是在32M.img分为一个区，并将其设置为引导扇区MBR。



sudo losetup -d /dev/loop3 卸载映像

·将活动分区格式化成ext2fs，并mount到rootfs目录上

sudo losetup -o $((2048\*512)) /dev/loop3 32M.img

其中，$((2048\*512))是分区的起始位置,通过-o选项设置数据偏移量，因为前面$((2048\*512))是MBR内容。其中，2048是通过file 32M.img得到的startsector信息。

sudo mke2fs /dev/loop3

这里mke2fs即为格式化命令。

sudo mount /dev/loop3 rootfs

挂载活动分区到rootfs上。

将前面制作的bzImage和rootfs.img拷贝到rootfs中。

·添加grub功能

准备目录，拷贝文件

sudo mkdir rootfs/boot

sudo mkdir rootfs/boot/grub

sudo cp ./grub-0.97-i386-pc/boot/grub/\*rootfs/boot/grub

编写menu.lst

default 0  
timeout 30  
title linux on 32M.img  
root (hd0,0)  
kernel (hd0,0)/bzImage root=/dev/ram init=/bin/ash

initrd (hd0,0)/rootfs.img

default 0 代表grub的默认启动项为0号系统

timeout 30 若用户在30s中没有操作，则启动默认项

title 指定启动的操作系统菜单的名称

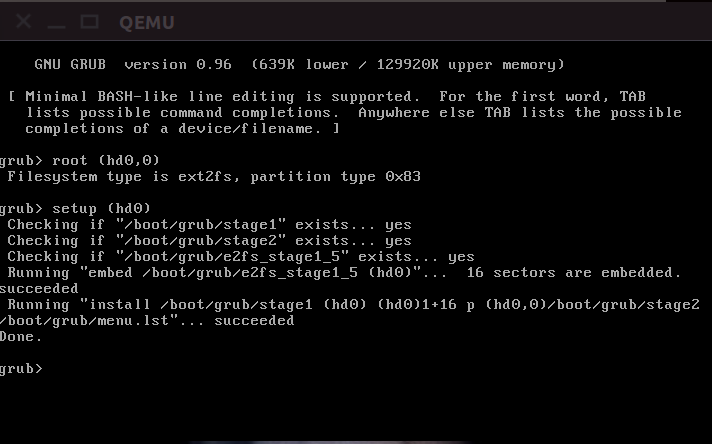
root指定启动分区，此处为（hd0,0）,表示第一块硬盘的第一个分区。

Kernel指定启动的内核的绝对路径和名称，root，init为参数

Initrd指定根系统目录

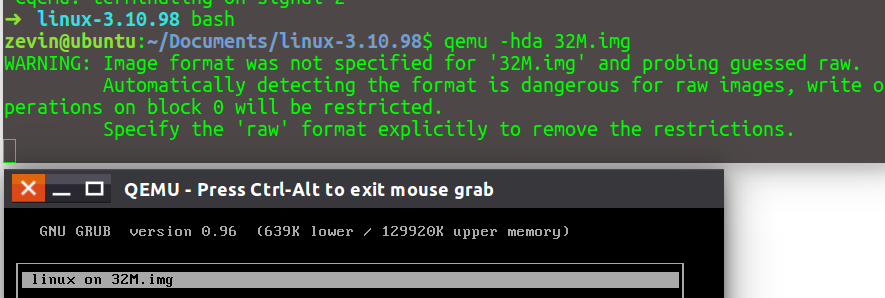
·利用grub启动软盘，在硬盘映像上添加grub功能

qemu -boot a -fda a.img -hda 32M.img

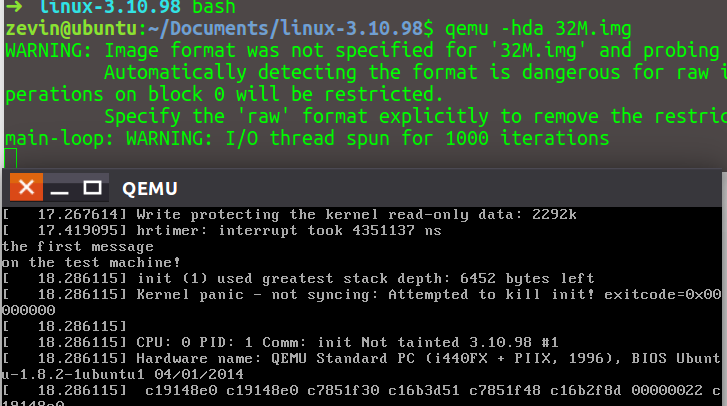


Root(hd0,0) 将根分区设置为磁盘第一分区

Setup(hd0) 自动安装grub

·测试从磁盘启动进入grub界面

成功！



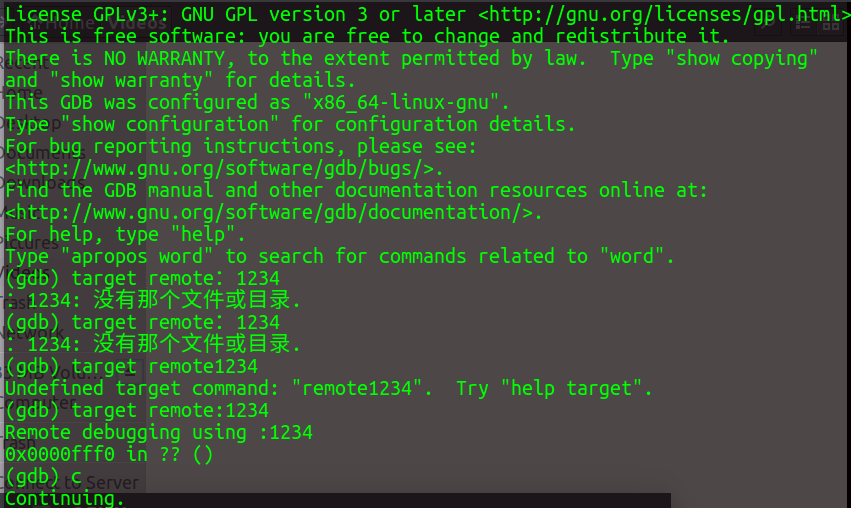
1. 利用qemu和gdb调试linux

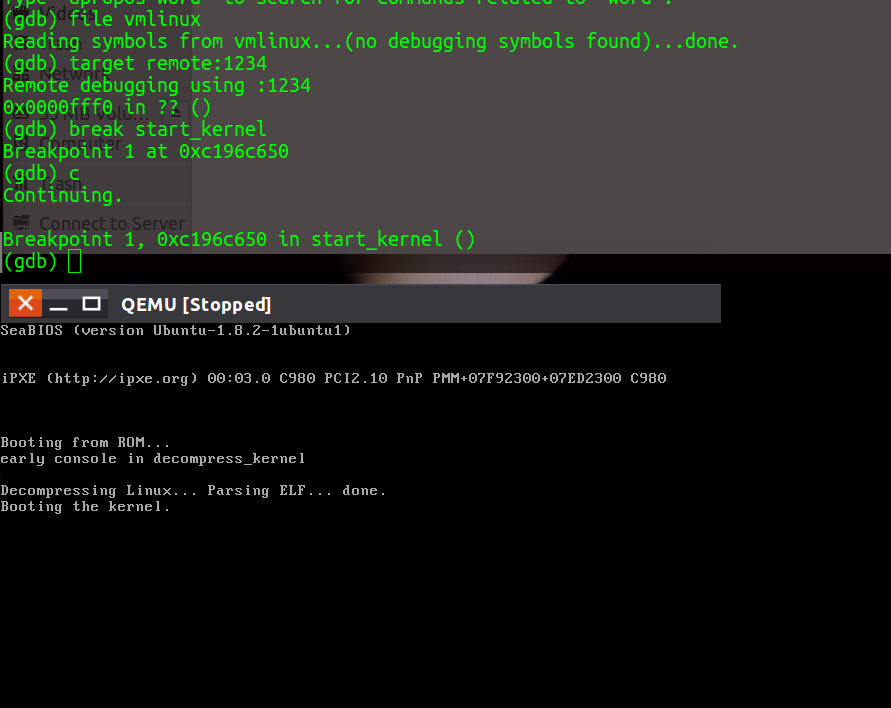
qemu -kernel arch/x86/boot/bzImage -s –S

-s 设置remote端口1234

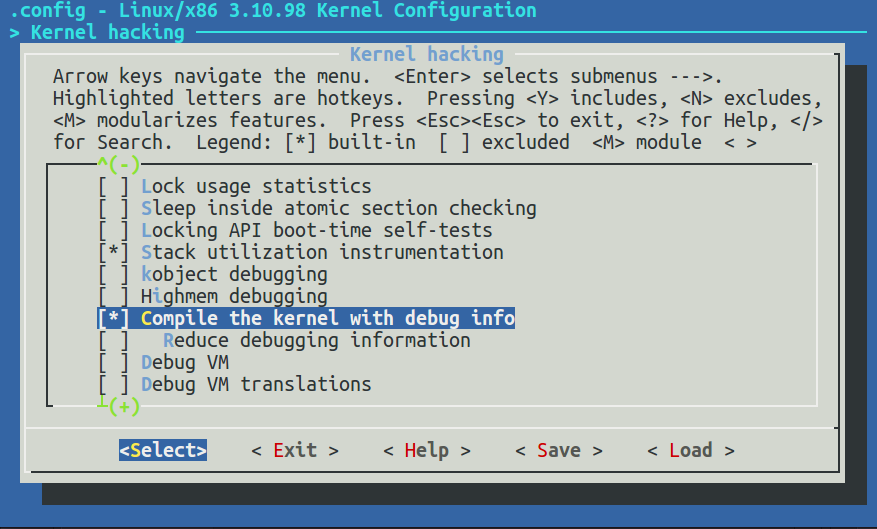
-S设置开启时中断

·打开gdb





没有携带调试信息，故重新配置linux，



再重新编译。

**实验吐槽与总结**：

首先贴上我在群里对本次实验的吐槽。

PB15111604\_金泽文(1033329461) 13:23:38

另外，我也很想吐槽一下这个实验。@xlanchen 。好几个英才班的同学羡慕我们学了很多东西，做了很多东西。但是我个人感觉，我们敲了很多东西，并且解决了一些遇到的小问题，但是实际上很多人没有学到很多东西，（这里很钦佩送小牛同学）。我记得老师当时说过（可能不那么认真的）一句话：这个实验你就是照着敲一遍也能学到东西。首先，这里我不是很清楚我们的实验二，老师到底要求或者期望我们达到怎样的一个水平，对grub引导机制这个原理的理解程度。其次，整个教程都是教我们敲命令。我比较早地完成了实验，对原理稀里糊涂，好多同学问我，他们也表示稀里糊涂。为什么呢？我想是因为没有强调原理啊。哪怕老师之前讲了一些启动的东西，但是挂载是什么，生成映像怎么实现，都没有说。这些如果本来就要求我们自己找资料自己学，那也理解，但是教程本身没有反映这样一种需求。再其次，我想知道能不能不直接给我们“所有的“命令，一是直接敲上去没理解没意思，二是很多命令教程里或刻板或易误解。倒不如让我们自己想自己推更有帮助。以上是我的看法，当然可能都是 错的 ，不过都跟老师反映一下。（我去吃早饭了）

总结与反思：  
在这个实验中，我学会了很多解决问题的方法。csdn，man，google，stackoverflow，与同学分享遇到的错误与解决方法等等等等；同时，我学会并熟练掌握了很多shell语法，习惯了命令行的环境。对于实验本身，我加深了对grub引导系统启动整个机制的理解，不过让我逐字逐句地、按部就班地、仔仔细细地说出整个机制的细节，还有许多不足。

实验中遇到的各种脑残问题，除了难以启齿的智障问题，都在前面的实验步骤中有所指明。

最后，整个实验以及吐槽后的讨论，我更加明白了一个道理：computer science 或者 information technology行业的人，确实需要强硬的搜集信息，解决问题的能力。今后应该更加锻炼这种能力！