linux、osx 下搭建操作系统开发环境 v1.1

boxcounter

November 14, 2013

目录

1 前言 **2**

2 创建虚拟磁盘，并分区 **2**

3 安装 **grub2** 到虚拟磁盘 **4**

4 在 **bochs** 中使用虚拟磁盘 **4**

5 编写最简单的系统内核 **5**

6 使用 **grub2** 启动自行编写的操作系统内核 **6**

7 **osx** 中读写 **ext2** 文件系统的虚拟磁盘 **6**

8 附录一：**x64 ubuntu 12.04.2** 中执行 **grub-install** 遇到的问题 **7**

9 附录二：创建虚拟磁盘分区的另外一种方法（**losetup**） **8**

10 主要参考资料 **8**

版本记录

- v1.0 – 2013-09-02，初始发布。
- v1.1 – 2013-11-14，增加 x64 系统下的说明。

1 前言

本文档记录的内容既适用于 x86 也适用于 x64，只是对于后者有一些环境要求。

之前开发使用的方法是：自己的引导代码 + 虚拟软盘。优点是搭建简单，所有代码都是自己编写，可控性强。最近想试试使用 grub 的引导功能，于是花了些时间琢磨。搜出来的相关资料有不少，但是要么是 grub1 的，要么太过零散，要么描述太过简略。总之，没有一篇文章详细的讲述整个配置过程。所以我就在搭建的过程中顺手整理了这么一篇完整的、完全从零开始的方法，其中每一步都有较丰富的说明。

另外，本文档介绍的方法适用于 osx 和 linux，实际上整个过程中大部分必须使用到 linux。也就是说如果要按照本文档来搭建开发环境，linuxer 只需要使用自己的 linux 系统就行，而 osxer 还得备一套 linux 系统（比如虚拟机）。使用 linux 的主要原因是我选择了 ext2 作为文件系统，而 osx 上貌似只有读写 ext2 的 fuse-ext2，没有用于创建 ext2 分区的 fdisk 等工具（如果同好有 osx 的 ext2 创建工具推荐，劳烦分享给我[1]吧，不胜感激）。如果改用 fat32 就没有这个烦恼，整个过程都可以在 osx 下完成，因为 osx 的 fdisk 就可以创建 fat32 分区。

我使用的系统、软件情况：

- 系统：x86 ubuntu 12.04.2/x64 ubuntu 13.10、osx 10.8.4/osx 10.9
- nasm 2.10.09
- bochs 2.6.1
- grub2 2.0.0

如果同好使用的环境不一样，可能需要根据情况自行调整一些细节。

另外，本文提供的命令在显示时候可能会自动折行，所以复制到剪贴板中之后（在折行处）可能会有多余的空格，请同好自行删减。

2 创建虚拟磁盘，并分区

首先说明：

- 这里的目标磁盘的属性是：16 headers, 63 sectors per track, 512 bytes per sector。意味着每一个 cylinder 的大小是 516096 字节（16*63*512）。
- “#cylinders”表示柱面数，主要关系到磁盘大小。比如 10MB 的虚拟磁盘，#cylinders 就为 20。
- 需要在 linux 系统中进行，使用的工具是 kpartx，系统默认没有自带，需要下载。

好了，开始吧。

1. dd if=/dev/zero of=antos.img bs=516096 count=#cylinders
   创建虚拟磁盘。也可以使用 bochs 附带的 bximage 工具来完成。

2. ps aux | grep loop
   默认是搜索不到名为“[loopX]”进程的。如果有发现，那记住输出中的“[loopX]”进程。

3. kpartx -av ./antos.img
   挂载虚拟磁盘，可能没有输出。

4. ps aux | grep loop
   正常情况下，这里会发现一个名为“[loop0]”的进程。说明 antos.img 被挂载到了“/dev/loop0”设备上。如果前面搜索结果中已经有了“[loopX]”进程，那新增加的那个进程就是挂载的设备名。

5. fdisk -u -C#cylinders -S63 -H16 /dev/loop0
   为磁盘分区，以 #cylinders=20、单个分区为例：

```
root@ubuntu:~# fdisk -u -C20 -S63 -H16 /dev/loop0
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier 0x136d49ee.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

Command (m for help): o <<<=== Create a new empty DOS partition table
Building a new DOS disklabel with disk identifier 0x5bd665d5.
Changes will remain in memory only, until you decide to write them.
```

[1]我的邮箱：ns.boxcounter[at]gmail.com

```
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

Command (m for help): n <<<=== Create a new partition
Partition type:
p   primary (0 primary, 0 extended, 4 free)
e   extended
Select (default p): <<<=== 回车
Partition number (1-4, default 1): <<<=== 回车
First sector (2048-20159, default 2048): <<<=== 回车
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-20159, default 20159): <<<=== 回车
Using default value 20159

Command (m for help): a <<<=== Toggle the bootable flag (Optional)
Partition number (1-4): 1 <<<=== 分区1

Command (m for help): p <<<=== Print the partition table.

Disk /dev/loop0: 10 MB, 10321920 bytes
16 heads, 63 sectors/track, 20 cylinders, total 20160 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x5bd665d5

Device Boot      Start         End      Blocks   Id  System
/dev/loop0p1   *        2048       20159        9056   83  Linux
<<<=== 如果使用附录2记录的方法，需要记录Start和Blocks的值，本例子中分别是2048和9056。

Command (m for help): w <<<=== Write partition table to our 'disk' and exit
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 22: Invalid argument.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.
<<<=== Ignore any errors about rereading the partition table. Since it's not a physical device we really c
```

6. kpartx -dv ./antos.img
   卸载磁盘，应该输出“loop deleted : /dev/loop0”。

7. kpartx -av ./antos.img
   挂载分区磁盘，这次新创建的分区也会自动挂载。
   正常会输出“add map loop0p1 (252:0): 0 18112 linear /dev/loop0 2048”，表示分区挂载到了“/dev/mapper/loop0p1”设备上。

8. mke2fs -b1024 /dev/mapper/loop0p1
   格式化分区，”-b1024” 表示使用 1KB 的 block。

9. mkdir /tmp/antos
   创建挂载目录。

10. mount -text2 /dev/mapper/loop0p1 /tmp/antos
    挂载分区到目录。

11. ls /tmp/antos/
    如果前面的步骤都成功，会看到名为“lost+found”的目录，说明磁盘和分区都正确的创建了。

此后，就可以使用 kpartx 挂载虚拟磁盘，用 mount 挂载分区到指定目录了。

3 安装 grub2 到虚拟磁盘

假设磁盘挂载到设备“/dev/loop0”上，分区挂载到“/tmp/antos”目录下。执行以下命令:

```
grub-install --no-floppy --modules="biosdisk part_msdos ext2 configfile normal multiboot" --root-directory=/tmp/antos /dev/loop0
```

安装过程中可能会报警告，只要最后输出“Installation finished. No error reported.”就表示安装成功了。
如果使用的系统是 x64 架构的，需要使用新一些的系统，比如 ubuntu 13.10。具体原因请参考附录一。
（在 fedora 等 redhat 系中使用的名称是“grub2-install”）

4 在 bochs 中使用虚拟磁盘

1. 确认虚拟磁盘的属性。
   先挂载虚拟磁盘，然后执行“disk -u -l /dev/loop0”，正常会有如下输出

```
1 Disk /dev/loop0: 10 MB, 10321920 bytes
2 16 heads, 63 sectors/track, 20 cylinders, total 20160 sectors
3 Units = sectors of 1 * 512 = 512 bytes
4 Sector size (logical/physical): 512 bytes / 512 bytes
5 I/O size (minimum/optimal): 512 bytes / 512 bytes
6 Disk identifier: 0x6418cb2e
7
8       Device Boot      Start         End      Blocks   Id  System
9 /dev/loop0p1   *        2048       20159        9056   83  Linux
```

   第 2 行说明了虚拟磁盘的属性。(前面使用 fdisk 为磁盘分区的时候也有输出同样的内容，如果记下来了，就无需这一步。)

2. 创建 bochs 虚拟机配置文件。
   不带参数运行 bochs，应该会有这样的输出:

```
1. Restore factory default configuration
2. Read options from...
3. Edit options
4. Save options to...
5. Restore the Bochs state from...
6. Begin simulation
7. Quit now

Please choose one: [2]
```

   选择 4，然后输入配置文件名，比如“antos.bxrc”，提示保存成功后退出 bochs。这样就有了一份默认配置的 bochs 虚拟机配置文件。

3. 修改 bochs 虚拟机配置文件，以适应我们的需要。

   (a) 添加虚拟磁盘

   ata0-master: type=none

   改为

   ata0-master: type=disk, path="./antos.img", cylinders=#cylinders,heads=#heads,spt=#sec-per-track

   对应之前获取到的磁盘属性，这一行应该是:

   ata0-master: type=disk, path="./antos.img", cylinders=20,heads=16,spt=63

   (b) 修改启动项为磁盘。

   boot: floppy

   改为

   boot: disk

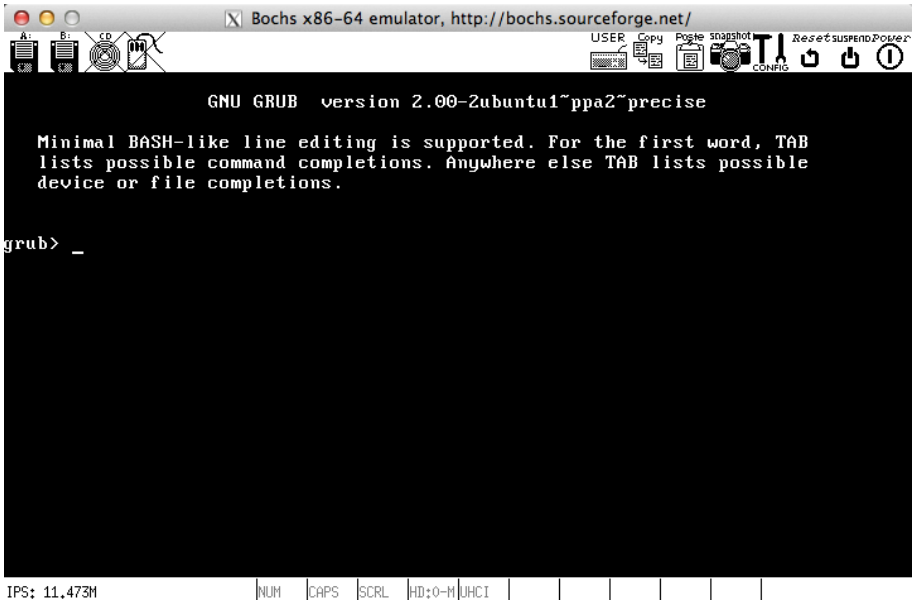
(c) 开启 bochs 的 magic breakpoint。

magic_break: enabled=0

改为

magic_break: enabled=1

到这里，就可以在 bochs 中运行了（命令是“bochs -f antos.bxrc”），并且看到 grub2 的命令提示符，如图：



5 编写最简单的系统内核

kernel.asm

```
[section .kernel]
[bits 32]

load_base equ 0x100000

;
; multiboot header
;
align 8
multiboot_header:
MBH_magic            equ 0xE85250D6
MBH_architecture     equ 0            ; 32-bit protected mode
MBH_header_length    equ multiboot_header_end - multiboot_header
MBH_checksum         equ -(MBH_header_length + MBH_magic + MBH_architecture)

dd MBH_magic
dd MBH_architecture
dd MBH_header_length
dd MBH_checksum

; tags
info_request_tag:
dw 1
dw 0
dd info_request_tag_end - info_request_tag
dd 5
dd 6
info_request_tag_end:
align 8
address_tag:
dw 2
dw 0
dd address_tag_end - address_tag
dd load_base                ; header_addr
dd load_base                ; load_addr
dd 0                        ; load_end_addr
dd 0                        ; bss_end_addr
```

```
address_tag_end:
align 8
entry_address_tag:
dw 3
dw 0
dd entry_address_tag_end - entry_address_tag
dd load_base + kernel_entry
entry_address_tag_end:
align 8
; end tag
dw 0
dw 0
dd 8
multiboot_header_end:

kernel_entry:
xchg bx, bx                ; magic breakpoint
jmp $
```

整个源码的绝大部分都是按照 multiboot2 规范创建 multiboot header。
编译方法：“nasm kernel.asm -o kernel.bin”。编译过程中可能会报警告，无视它。
将虚拟磁盘挂载到某个目录，然后将 kernel.bin 拷贝到分区的根目录，即和/boot 目录同一层目录。

6 使用 grub2 启动自行编写的操作系统内核

假设分区挂载到“/tmp/antos”目录下，那么需要创建的 grub 配置文件是“/tmp/antos/boot/grub/grub.cfg”，将以下几行文本贴进去：

```
set default=0
insmod ext2
set root=(hd0,1)
set timeout=10
menuentry "antos 0.0.1" {
insmod ext2
set root=(hd0,1)
multiboot2 (hd0,1)/kernel.bin
boot
}
```

其中 (hd0,1) 表示咱之前创建的虚拟磁盘的第一个分区，kernel.bin 就是前面编译的系统内核文件。
现在可以启动咱的 bochs 虚拟机了，执行“bochs -f antos.bxrc”。再输入 c 继续执行后，应该就能看到 bochs 从咱的虚拟磁盘引导，然后可以看见 grub 的选择界面，最后会中断到咱系统内核的“xchg bx, bx”指令，这是 bochs 内置的主动中断指令，即 magic breakpoint 机制。如下：

```
00449811185i[CPU0 ] [449811185] Stopped on MAGIC BREAKPOINT
(0) Magic breakpoint
Next at t=449811185
(0) [0x000000100053] 0010:0000000000100053 (unk. ctxt): jmp .-2 (0x00100053)      ; ebfe
00449811185i[XGUI ] Mouse capture off
<bochs:2>
```

ok，整个配置过程就完毕了，整个过程都是在 linux 中完成的。使用 fat32 的 osx 同好可以使用类似的方法来完成。整个过程每一步的功能都写的很清楚了，看到这里理理思路应该就明白整个流程了。

7 osx 中读写 ext2 文件系统的虚拟磁盘

最后说说 osx 相关的内容，因为我不想每次做开发的时候都需要开个 linux 虚拟机。以下是在 osx 下读写虚拟磁盘的方法，比如更新的 kernel.bin 等等。linuxer 可以无视这一步。

1. 安装 osxfuse 和 fuse-ext2
   fuse-ext2 默认只能以只读方式挂载设备，所以需要进行以下修改使其默认以可读可写方式挂载设备：
   sudo vi /System/Library/Filesystems/fuse-ext2.fs/fuse-ext2.util
   搜索定位到 Mount 函数，为其名为“OPTIONS”的变量增加额外的“rw+”选项。
   比如：原内容为：

```
function Mount ()
{
  LogDebug "[Mount] Entering function Mount..."
  # Setting both defer_auth and defer_permissions. The option was renamed
  # starting with MacFUSE 1.0.0, and there seems to be no backward
  # compatibility on the options.
  OPTIONS="auto_xattr,defer_permissions"
  ...
}
```

改为:

```
function Mount ()
{
  LogDebug "[Mount] Entering function Mount..."
  # Setting both defer_auth and defer_permissions. The option was renamed
  # starting with MacFUSE 1.0.0, and there seems to be no backward
  # compatibility on the options.
  OPTIONS="auto_xattr,defer_permissions,rw+"
  ...
}
```

2. hdiutil attach -nomount antos.img
   挂载磁盘到设备，会输出如下信息:

```
/dev/disk1            FDisk_partition_scheme
/dev/disk1s1          Linux
```

   说明虚拟磁盘已经挂载到/dev/disk1 设备上了，分区已经挂载到/dev/disk1s1。
   (之所以加上 -nomount 参数，是因为 hdiutil 没法正确地挂载 ext2 分区到目录)

3. /sbin/mount_fuse-ext2 /dev/disk1s1 ./mnt
   挂载分区到目录。到这里，就可以操作分区了。

4. umount ./mnt
   从目录卸载分区。

5. hdiutil detach /dev/disk1
   卸载虚拟磁盘。(注意是磁盘设备，不是分区设备 disk1s1。)

8 附录一: x64 ubuntu 12.04.2 中执行 grub-install 遇到的问题

我在 x64 ubuntu 12.04.2 中执行“安装 grub2 到虚拟磁盘”操作时总是失败，如下:

```
root@x64:~# grub-install --no-floppy --
modules="biosdisk part_msdos ext2 configfile normal multiboot" --root-directory=/tmp/antos /dev/
loop0
Path `/tmp/antos/boot/
grub' is not readable by GRUB on boot. Installation is impossible. Aborting.
```

加上 -deubg 选项后，发现 grub-install 输出了这么几行:

```
+ /usr/local/sbin/grub-probe -t fs /tmp/antos/boot/grub
+ return 1
+ gettext_printf Path `
%s' is not readable by GRUB on boot. Installation is impossible. Aborting.\n /tmp/antos/boot/grub
```

手动执行 grub-probe，输出如下:

```
root@ubuntu:~# /usr/local/sbin/grub-probe -t fs /tmp/antos/boot/grub
/usr/local/sbin/grub-probe: error: disk `lvm/loop0p1' not found.
```

我查阅了很多资料，都没有明确的解决方案。经过了约莫 20 个小时的摸索，最终发现只需要使用新版本的系统自带的 grub 2.0.0 即可。
(注: x64 ubuntu 12.04.2 中的 grub2 是我源码编译的 2.0.0 版，也尝试过使用 trunk 源码编译或者使用系统自带的 1.99，都会报错。)

9 附录二：创建虚拟磁盘分区的另外一种方法（losetup）

需要说明，这种方法较前面介绍的使用 kpartx 的方法繁琐，所以并不推荐（特别是如果要使用多分区）。补充在这里的原因是我最开始搜索到的资料使用的就是 losetup 工具，摸索成功之后才发现 kpartx。

另外，操作过程中有部分步骤和前面讲述的步骤一样，所以省略了那些步骤的说明。

1. dd if=/dev/zero of=antos.img bs=516096 count=#cylinders
2. losetup /dev/loop0 ./antos.img
   这个时候执行“ps aux | grep loop”，会看到一个名为 [loop0] 的进程。(如果 loop0 被占用，可以换一个设备)
3. fdisk -u -C#cylinders -S63 -H16 /dev/loop0
4. losetup -d /dev/loop0
   到这里，虚拟磁盘已经创建完毕了，从设备（“loop0”）上卸载虚拟磁盘，准备格式化分区。
5. losetup -o1048576 /dev/loop0 ./antos.img
   再次挂载，与前面挂载不同的是，这次使用了“-o1048576”参数，目的是跳过前 1048576 字节，来到分区的开始。前面提到要记住 Start 的值，即分区开始扇区号，这里就需要使用它了，1048576=2048*512。
6. mke2fs -b1024 /dev/loop0 9056
   对加载到“loop0”设备上的分区（注意是分区，不是整个磁盘了，前面咱跳到了分区开始处）进行格式化，使用的是 ext2 文件系统。“-b1024”表示使用 1KB 的 block，9056 就是之前的 Blocks 的值，即整个分区的 blocks 数。
7. mkdir /tmp/antos
8. mount -text2 /dev/loop0 /tmp/antos
9. umount /dev/loop0
10. losetup -d /dev/loop0 卸载目录和设备。

10 主要参考资料

- The Multiboot Specification
- grub2 源码
- Mac OS X 下读写 ext2/ext3 文件系统