

操作系统作业 PA1 实验报告

221900180 田永铭

2024 年 3 月 6 日

0.1 虚拟机创建和系统安装过程描述

1. 下载 **qemu** 原本在 ubuntu 虚拟机内，通过指令下载并且编译，不过缺失太多依赖，安装失败。后选择在 qemu 官网下载 exe 直接运行，取得成功。

2. 下载镜像 分别从微软官网下载 windows-7 镜像和 Ubuntu 官网下载 ubuntu-22 镜像。

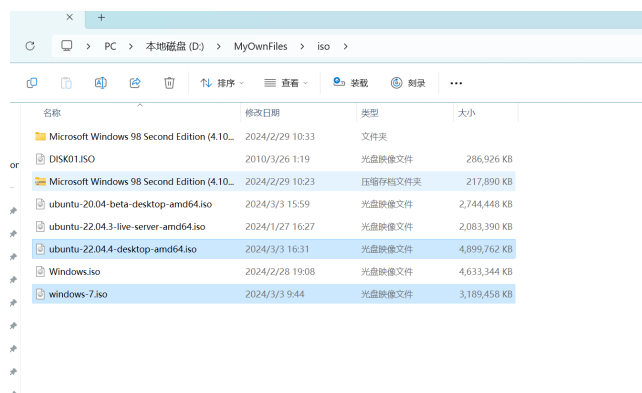


图 1: 下载的镜像如上图所示，选中的下面两个为实际采纳的

3. 先安装 **windows** 操作系统 在 qemu 文件位置打开 cmd, 输入 “qemu-img create os.img 35G” 创建一个虚拟磁盘，再输入 “qemu-system-x86_64 -m 2048 -smp 2 os.img D:\MyOwnFiles\iso\windows-7.iso” 命令进行安装 (后面跟的路径为本地 iso 文件地址)。安装时，进行磁盘分区，预留出一半空间给 linux 系统。下图为安装成功的 windows-7 图片展示。

4. 再安装 **linux** 操作系统 在 qemu 文件位置打开 cmd，输入 “qemu-system-x86_64 -m 4096 -smp 2 myos.img -cdrom D:\MyOwnFiles\iso\ubuntu-22.04.4-desktop-amd64.iso -boot d” 安装 linux，选择安装到之前预留的分区内。下图为安装成功的 linux 图片展示。

5. 检验安装是否成功 再次利用命令 “qemu-system-x86_64 -m 2048 -smp 2 os.img”，可以选择进入两种操作系统中的一种，分别可以成功进入。

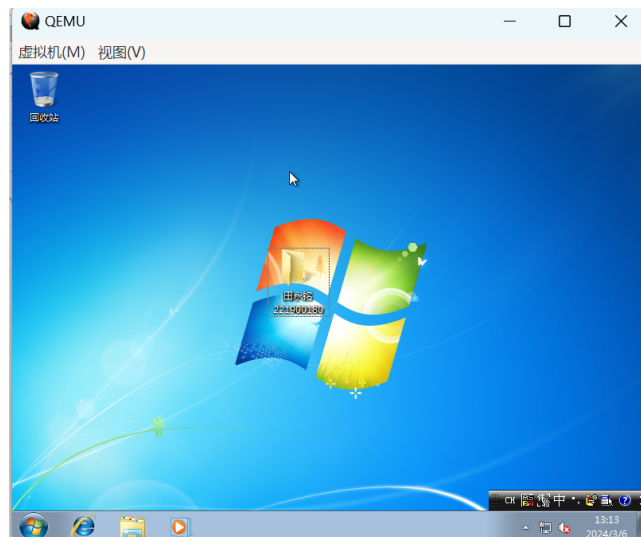


图 2: windows-7

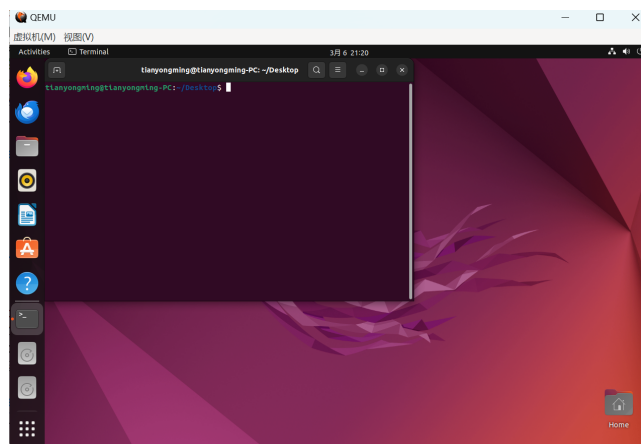


图 3: linux

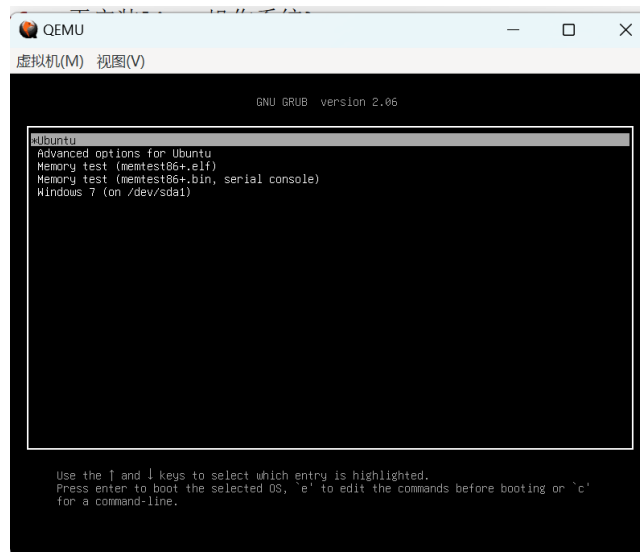


图 4:

0.2 读取虚拟磁盘 MBR

1. 下载工具 通过 <https://xiazai.zol.com.cn/detail/47/465664.shtml> 下载 windows 的 dd 读取软件。

2. 进行读取 在包含 qemu 的目录下创建 mbr.bin, 在命令行输入 `dd if=myos.img of=mbr.bin bs=512 count=1`, 打开 powershell 输入 `Format-Hex "D:/MyOwnFiles/qemu/mbr.bin"`, 读取磁盘 MBR 的前 512 个字节。以下分别是只装了 windows7 和装了双系统的读的 MBR:

图 5: win7-mbr

图 6: 双系统 mbr

-m i386 -M addr16,data16,intel mbr.bin”,得到 mbr 的汇编语言分别如后图所示 (太长了，只截取了一段)。

```

00000000 <.data>:
0: 33 c0      xor    ax,ax
2: 8e d0      mov    ss,ax
4: bc 00 7c  mov    sp,0x7c00
7: 8e c0      mov    es,ax
9: 8e d8      mov    ds,ax
b: be 00 7c  mov    si,0x7c00
e: bf 00 06  mov    di,0x600
11: b9 00 02  mov    cx,0x200
14: fc        cld
15: f3 a4      rep movs BYTE PTR es:[di],BYTE PTR ds:[si]
17: 50        push   ax
18: 68 1c 06  push   0x61c
1b: cb        retf
1c: fb        sti
1d: b9 04 00  mov    cx,0x4
20: bd be 07  mov    bp,0x7be
23: 80 7e 00 00 cmp    BYTE PTR [bp+0x0],0x0
27: 7c 0b      jl     0x34
29: 0f 85 0e 01 jne    0x13b
2d: 83 c5 10  add    bp,0x10
30: e2 f1      loop  0x23
32: cd 18      int    0x18
34: 88 56 00  mov    BYTE PTR [bp+0x0],dl
37: 55        push   bp
38: c6 46 11 05 mov    BYTE PTR [bp+0x11],0x5
3c: c6 46 10 00 mov    BYTE PTR [bp+0x10],0x0
40: b4 41      mov    ah,0x41
42: bb aa 55  mov    bx,0x55aa
45: cd 13      int    0x13
47: 5d        pop    bp
48: 72 0f      jb     0x59
4a: 81 fb 55 aa cmp    bx,0xaa55
4e: 75 09      jne    0x59
50: f7 c1 01 00 test   cx,0x1
54: 74 03      je     0x59
56: fe 46 10  inc    BYTE PTR [bp+0x10]
59: 66 60      pusha
5b: 80 7e 10 00 cmp    BYTE PTR [bp+0x10],0x0
5f: 74 26      je     0x87
61: 66 68 00 00 00 00 push   0x0
67: 66 ff 76 08 push   DWORD PTR [bp+0x8]
6b: 68 00 00  push   0x0
6e: 68 00 7c  push   0x7c00
71: 68 01 00  push   0x1
74: 68 10 00  push   0x10
77: b4 42      mov    ah,0x42
79: 8a 56 00  mov    dl,BYTE PTR [bp+0x0]

```

图 7: 部分反汇编结果

2. 进行分析 MBR 功能大致如下:(1) 首先检查硬盘中分区表是否完好。(2) 从分区表查找可引导的“活动”分区。(3) 将活动分区中第一逻辑扇区数据加载到内存中。

结合反汇编代码具体来看:

1. **第一条关键词句 jmp short 0x65** 第一行就是转移指令，跨过了很大一块区域，原因是这一段空间用作 BIOS 参数块。开头直接跳到 65 指

令。

2. 第二部分关键语句 `test dl,0x80` 引导驱动检查: 确认是软驱还是硬驱。
3. 第三部分关键语句 `a7: 72 3d jb 0xe6||a9: 81 fb 55 aa cmp bx,0xaa55||ad: 75 37 jne 0xe6` 这是在检查是否支持 LBA, 即寻址模式的检查。
4. 第四部分关键语句 `a3: cd 13 int 0x13||c9 66 89 5c 08 mov DWORD PTR [si+0x8],ebx||165: bf 00 80 mov di,0x8000` 中断例程启动, 完成后将磁盘数据写入内存。将编号 80 的驱动器 (c 盘) 的第一个扇区 (512 Byte) 读到内存地址。准备启动盘驱动磁盘数据包并引导内核督导内存 00:8000, 再执行引到内核指令。
5. 第五部分关键语句最后面部分 (结尾是 `55aa`) 此部分为 MBR 磁盘分区表。

具体来看分区部分第 1 字节 80 是引导标志, 表示为活动分区; 第 2、3、4 字节是本分区的起始磁头号、扇区号、柱面号; 第 5 字节是分区类型符; 第 6、7、8 字节是分区的结束磁头号、扇区号、柱面号; 第 9、10、11、12 字节是逻辑起始扇区号, 本分区之前已用了的扇区数; 第 13、14、15、16 字节是本分区的总扇区数。

一个分区表能表示 4 个分区。以下是我的分区详细解释:

以 win-7 的第一块分区为例子: `80 20 21 00 07 DF 13 0C 00 08 00 00 00 20 03 00` 80 代表这个分区位活动分区; 20, 21, 00 表示这个分区起始扇区为 16 进制下的 20 磁头, 21 扇区, 0 柱面; 07 代表分区文件系统为 NTFS; DF, 13, 0C 是相应 16 进制下结束的磁头、扇区、柱面号; 00 08 00 00 反向, $(00000800)_{16}$ 表示其实逻辑扇区和逻辑 0 扇区的差, 说明前面已经有 $(00000800)_{16}$ 个扇区; 00 20 03 00 表示总扇区数字。

对比装 linux 前后的 MBR 经过对比, 可以发现, 我将 linux 的启动分区挂载在 windows 空的分区下后, 原本是 00(表示分区未用) 的分区被 linux 占用。即图片中 win-7 情况下都是 00 的分区在装了 linux 被使用起来, 有了新的数字。

同时, 我发现原本分区表第一个字节的 80 变为了 00, 即变成了非活动分区。再经观察, 得知前两个分区的其他数字完全没有改变, 这也完全符合预期, 即双系统互不干扰, 并没有覆盖。

再对比装 linux 前后的汇编代码，可以发现大体相同，主要的功能都是引导作用。

利用 `fdisk -l` 指令检验分区在 ubuntu 内的终端里，输入 `fdisk -l`，获取磁盘分区信息如下图所示：

```
lanyongming@lanyongming-PC:~$ sudo su
[sudo] password for lanyongming:
root@lanyongming-PC:/home/lanyongming/Desktop# fdisk -l
Disk /dev/loop0: 74.22 MiB, 77859904 bytes, 151592 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 /dev/loop1: 4 KiB, 4096 bytes, 8 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 /dev/loop2: 266.63 MiB, 279584768 bytes, 546064 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 /dev/loop3: 496.98 MiB, 521121792 bytes, 1017816 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 /dev/loop4: 12.32 MiB, 12922880 bytes, 25240 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 /dev/loop5: 91.69 MiB, 96141312 bytes, 187776 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
```

图 8:

```
Disk /dev/loop6: 40.43 MiB, 42393600 bytes, 82800 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 /dev/loop7: 452 KiB, 462048 bytes, 904 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 /dev/sda: 35 GiB, 37580963840 bytes, 73400320 sectors
Disk model: QEMU HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xc5c10e35

Device Boot Start End Sectors Size Id Type
/dev/sda1 2048 206847 204800 100M 7 HPFS/NTFS/exFAT
/dev/sda2 206848 37740735 37541888 17.9G 7 HPFS/NTFS/exFAT
/dev/sda3 37750782 73390271 35647490 17G 5 Extended
/dev/sda5 * 37750784 38334463 583680 285M ef EFI (PAT-12/16/32)
/dev/sda6 38336512 73390271 35051760 16.7G 83 Linux
```

图 9:

可以从图中看到我规划的分区，17.9G 给 window-7，16.7G 给 linux。更加仔细地来看，第一个 `sda1` 起始位置是 2048，刚好是我的 `mbr` 读出来的 $(00000800)_{16}$ 表示前面有多少个分区对应，这个数字在 10 进制下就是图片中的 2048。同时，也可以通过简单的计算，将其他数字都一一对应上，这再一次验证了我读的 `mbr` 是正确的。