中国科学技术大学计算机学院

实验报告

实验题目：lab1 multiboot 启动

学生姓名：王舒

学生学号：PB19071472

完成日期：2021.3.21

一、原理说明

1.启动协议

在启动一个操作系统之前，需要一个引导机制帮助加载 OS。而 Multiboot 规范指出了引导程序和操作系统之间的接口，这样符合引导规范的程序就能引导任何符合规范的操作系统。Multiboot 可以跳过一些低端的启动过程，使 CPU 进入保护模式。

OS 映像是引导程序载入到内存的初始二进制映像，它必须包括一个 Multiboot 头，需要按照规范表来设定参数。其中，本次实验只用到了前三个。Magic 是标志头的魔数，它必须等于十六进制值 0x1BADB002。Flags 指出 OS 映像需要引导程序提供或支持的特性，可以设置它的各个位。Checksum 是一个用于检查的数，是一个 32 位的无符号值，当与其他的 magic 域（也就是 magic 和 flags）相加时，结果必须是 32 位的无符号值 0（即 magic + flags + checksum = 0）。

在本次实验中，.S 文件的前面一部分就是在设置这几个数字，从而可以进入系统态，进行接下来代码段的操作。

2.QEMU

QEMU 是一个操作系统模拟器，可以模拟出一台能够独立运行操作系统的虚拟机。简单地说，它可以虚拟出 CPU、内存及 I/O 设备等硬件。因此，利用 QEMU 就可以设计操作系统，运行我们编译好的链接。

依据 QEMU 开发者的博客，vCPU（虚拟出的 CPU）调用 KVM（Keyboard Video Mouse）的接口来执行任务的流程如下： QEMU 发起 ioctrl 来调用 KVM 接口，KVM 则利用硬件扩展直接将虚拟机代码运行于主机之上，一旦 vCPU 需要操作设备寄存器，vCPU 将会停止并退回到 QEMU，QEMU 去模拟出操作结果。虚拟机内存会被映射到 QEMU 的进程地址空间，在启动时分配。在虚拟机看来，QEMU 所分配的主机上的虚拟地址空间为虚拟机的物理地址空间。QEMU 在主机用户态模拟虚拟机的硬件设备，vCPU 对硬件的操作结果会在用户态进行模拟，如虚拟机需要将数据写入硬盘，实际结果是将数据写入到了主机中的一个镜像文件中。

QEMU 在 Linux 和 Windows 系统下都可以运行，本次实验中为了方便编译，在 Linux 环境下安装并运行，模拟 i386 的 32 位微处理器。

3.VGA

视频图形阵列（Video Graphics Array）是一个使用模拟信号的电脑显示标准。如今，这个标准已经十分过时了，但仍然是最多被支持的一个标准。它扩展为 256 色的 EGA 式色版，这 256 种颜色是可以改变的，因为 VGA 在指定色版颜色时，一个颜色频道有 6 个 bit，红、绿、蓝各有 64 种不同的变化，因此总共有 262,144 种颜色，这其中的任何 256 种颜色可以被选为色版颜色。这样，VGA 就可以显示 256 种颜色了。

VGA 使用的显存，透过一个窗口对应于 PC 的主存，它们的真实地址为 0xA000 和 0xC000 之间的存储器。而本次实验中 VGA 的显存地址为 0xB8000，为彩色文字模式和 CGA 兼容模式。

在显示文字时，VGA 支持 16 种前景色和 8 种背景色，并可以选择是否闪烁。每个字符

需要 2 个字节，分别存放字符的 ASCII 码和显示属性。

将需要显示的内容直接写进 VGA 显存中，可以输出信息。在给出的例子中，“movl $0x2f4b2f4f,0xB8000”，其中两个“2f”决定了每个字符的显示属性（前景色和后景色），而“4b”“4f”分别是“K”“O”的 ASCII 码。使用 movl 指令，就可以写入后一个数字代表的显存中。

4.串口

串行接口（Serial Port）是采用串行通信方式的扩展接口，即数据一位一位地顺序传送。而 UART 是一种异步收发传输器，是硬件的一部分，它可以将要传输的资料在串行通信和并行通信之间加以转换，是一种串口。

在本次实验中，实例代码实际上就是把需要输出的数据存到一个寄存器中，再把端口地址存到另一个寄存器中，最后再到对应的地址（即端口）输出存好的数据。基于串口的特性，每次只能输出一个字符。

二、源代码说明（.S 文件）

这份文件大致上可以划分为以下两个部分：

/*multiboot 规范下的参数设置*/

/*代码部分，包括 VGA 输出和串口输出*/

首先，我们需要一个全局可见的开始“标签”，即：

```
.global start
```

然后，设定 multiboot 规范需要的参数，并为 section 起一个名字，然后将头结构用.long 写进去。

```
MAGIC=0x1BADB002
FLAGS=0x0
CHECKSUM=-(0x0+0x1BADB002)

.section ".multiboot_header"/*multiboot header*/

.long MAGIC
.long FLAGS
.long CHECKSUM
```

接下来就可以开始代码段了。其中.code32 表示这是 32 位的代码。

```
.text
.code32/*this means 32 bit*/
start:
```

在 VGA 显示的部分，一条 movl 指令可以输出两个字符，对照 ASCII 表，分别写出所有的指令。最终可以输出绿底白字的“hello world! PB19071472_wang shu”。

```
movl $0x2f652f68,0xB8000
movl $0x2f6c2f6c,0xB8004
movl $0x2f202f6f,0xB8008
movl $0x2f6f2f77,0xB800c
movl $0x2f6c2f72,0xB8010
movl $0x2f212f64,0xB8014

movl $0x2f202f20,0xB8018/*blanks*/

movl $0x2f422f50,0xB801c
movl $0x2f392f31,0xB8020
movl $0x2f372f30,0xB8024
movl $0x2f342f31,0xB8028
movl $0x2f322f37,0xB802c
movl $0x2f772f5f,0xB8030
movl $0x2f6e2f61,0xB8034
movl $0x2f202f67,0xB8038
movl $0x2f682f73,0xB803c
movl $0x2f202f75,0xB8040
```

而在串口输出部分，参照给出的代码，一位一位输出，最终显示在终端“HELLO WORLD PB19071472WANGSHU”。

三、代码布局说明

.ld 文件，即链接描述文件，就像一份“安装说明书”，将编译好的.o 文件按照要求组装起来，写成一份 bin 文件。对它的说明如下：

首先从 start 进入（start 需要和.S 文件中的一致）。

将当前位置跳到 1M 处，这样可以跳过一些低端的启动过程。然后开始写代码部分。先写入 multiboot 头的 12 个字节（名称要和.S 文件中的对应），然后向后对齐 8 个字节，再将.S 文件中对应的代码段写进去。这样，就生成一个 bin 文件，可以直接运行了。

四、编译过程说明

Makefile 用于自动编译和链接，它将汇编文件编译成可执行文件，再把他们通过 ld 文件链接起来，得到 bin 文件。

因此，整个过程如下：

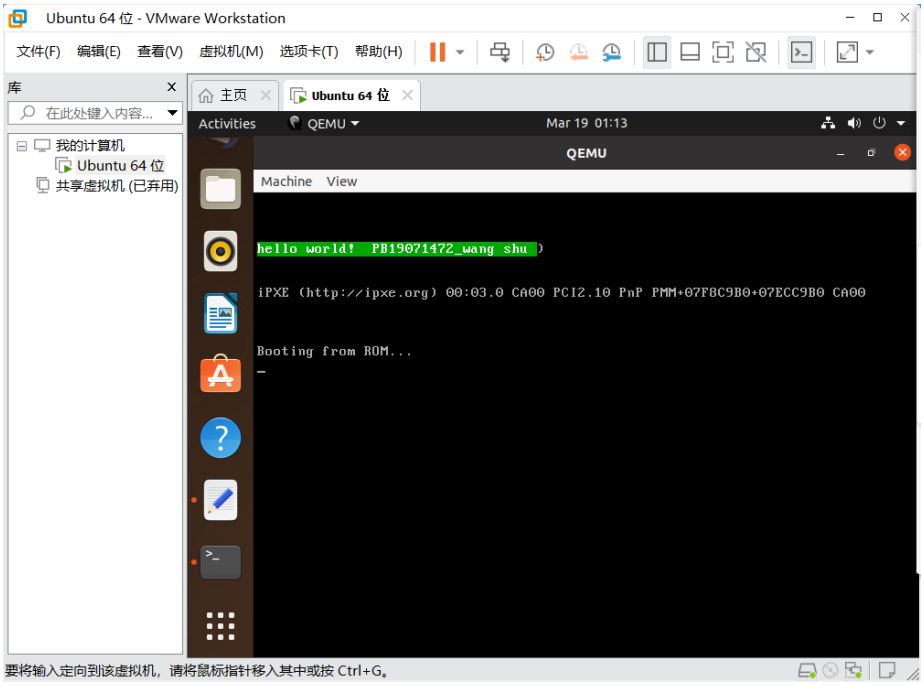
.S 文件在 Makefile 的指导下，被 gcc 编译为可执行文件（.o 文件），再依据写好的 ld 文件，将.o 文件写成 bin 文件。

五、运行和运行结果说明

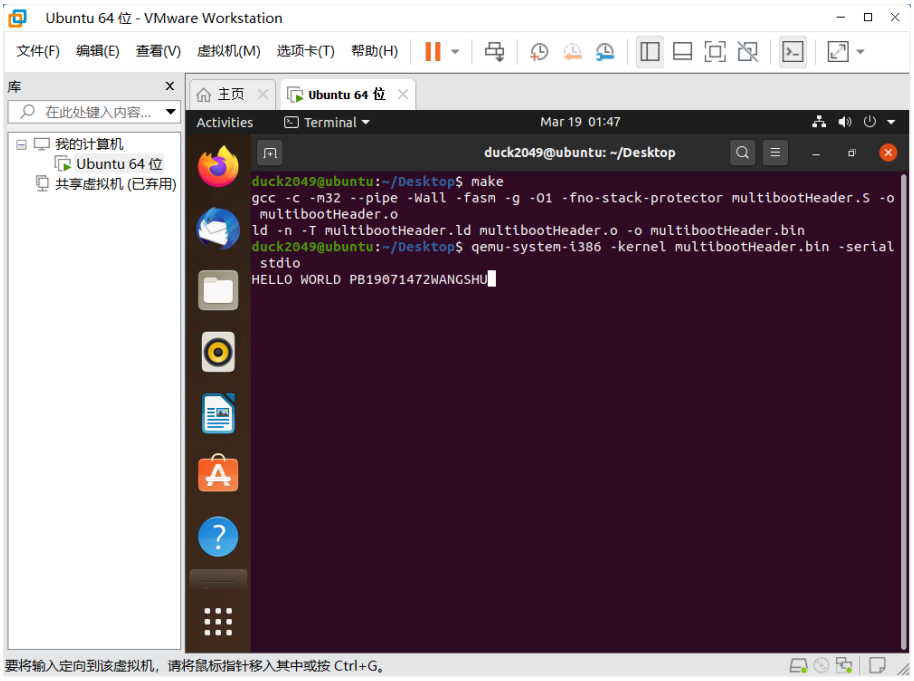
首先，写好三个文件后，在 terminal 里敲入 make，看到编译和链接成功。然后敲入 qemu 的调用命令，其中-serial stdio 表示串口输出结果显示在 terminal 中。如图

```
duck2049@ubuntu:~/Desktop$ make
gcc -c -m32 --pipe -Wall -fasm -g -O1 -fno-stack-protecto
r multibootHeader.S -o multibootHeader.o
ld -n -T multibootHeader.ld multibootHeader.o -o multiboo
tHeader.bin
duck2049@ubuntu:~/Desktop$ qemu-system-i386 -kernel multi
bootHeader.bin -serial stdio
duck2049@ubuntu:~/Desktop$
```

在 qemu 中的显示结果如下



在 terminal 中，串口输出的结果如下



六、遇到的问题和解决方案说明

本次实验虽然难度不大，但由于是第一次接触虚拟机、Linux 系统以及其他许多概念，还是出现了许多问题，耗费时间不短。

我觉得最难的地方就在于理解实验中各个部分之间的逻辑关系。第一次看到这么多奇形怪状的概念时，的确很令人一头雾水。我不得不查阅大量资料，逐渐摸索每一个文件的作用以及各个名词的含义，从最基本的 Linux 操作以及汇编代码学起。在这个过程中，我感到自己的学习能力得到了很大的提升，奇怪的知识增加了很多。

此外，还遇到了以下的问题。

1.虚拟机中的文件无法导出

我使用的虚拟机软件是 VMWare。尝试了建立共享文件夹和安装 VMware tools 后，都失败了，无法将虚拟机中的文件方便地导出。最后在 Linux 中安装了 Synology Drive Client，用 NAS 作为中转，实现了文件的同步。

2.安装虚拟机后电脑频繁死机

在实验中，只要打开虚拟机，就会多次蓝屏，提示 CLOCK_WATCHDOG_TIMEOUT。多方查阅后，删除了虚拟机的几个硬件，情况有所好转。

3.ld 文件和 Makefile 文件格式问题

在编写这两个文件时，多次无法正常通过，后来发现是格式有问题，在一些地方必须加空格，才能正常运行。这种错误让我有些意想不到，也很难发现问题所在。

4.VGA 输出

在写 VGA 输出的指令时，我没有注意到每条指令中，两个字符的前后顺序是相反的，导致修改花费了不少时间。此外，对应的地址也应该不断增加，才能正常显示，这一点在刚开始也被我忽视了。