> - 操作系统实验2020
> - 2020-2-15

1. 实验环境

1.1. 实验环境设备

- 自制OS的CPU：Intel 80386
- 模拟80386平台的虚拟机： QEMU
- 交叉编译的编译器： GCC
- 调试工具：GDB
- QEMU，GCC，GDB的运行平台： Linux
- 编程语言： C，x86 Assembly

1.2. 实验环境搭建

在物理机或VirtualBox上安装发行版Linux系统，例如Debian，Ubuntu，Arch，Fedora（x86，amd64架构皆可），在安装完成的Linux上安装QEMU，Vim，GCC，GDB，Binutils，Make，Perl，Git等工具，以Ubuntu为例

```
$sudo apt-get update
$sudo apt-get install qemu-system-x86
$sudo apt-get install vim
$sudo apt-get install gcc
$sudo apt-get install gdb
$sudo apt-get install binutils
$sudo apt-get install make
$sudo apt-get install perl
$sudo apt-get install git
```

如果有同学使用的是amd64架构，且在代码中使用了标准库，`gcc`使用`-m32`编译选项时需要进行额外配置

第一步：确认64位架构的内核

```
$dpkg --print-architecture
amd64
```

第二步：确认打开了多架构支持功能

```
$dpkg --print-foreign-architectures
i386
```

说明已打开，否则需要手动打开

```
$sudo dpkg --add-architecture i386
$sudo apt-get update
$sudo apt-get dist-upgrade
```

这样就拥有了64位系统对32位的支持

最后安装gcc multilib

```
$apt-get install gcc-multilib g++-multilib
```

我们推荐使用Git进行项目版本管理，在整个OS实验过程中，你可能会尝试多个想法实现课程需求，也可能因为一条路走不通选择另外的思路，这时候Git就为你提供了一个在不同版本之间穿梭的机器，也可能成为你调试不通时的**后悔药**

安装好git之后，你需要先进行配置工作

```
$git config --global user.name "Zhang San"      # your name
$git config --global user.email "zhangsan@foo.com"  # your email
```

现在你可以使用git了，你需要切换到项目目录，然后输入

```
$git init
...
$git add file.c
...
$git commit
...
```

这里只是简单的给出一些示例，具体的git使用方法需要同学们自行学习

- 廖雪峰的git教程

1.3. 代码运行与调试

利用QEMU模拟80386平台，运行自制的操作系统镜像os.img

```
$qemu-system-i386 os.img
```

利用QEMU模拟80386平台，Debug自制的操作系统镜像os.img，选项`-s`在TCP的1234端口运行一个gdbserver，选项`-S`使得QEMU启动时不运行80386的CPU

```
$qemu-system-i386 -s -S os.img
```

另开一个shell，启动GDB，连接上述gdbserver，在程序计数器0x7c00处添加断点，运行80386的CPU，显示寄存器信息，单步执行下一条指令

```
$gdb
$(gdb)target remote localhost:1234
...
$(gdb)b *0x7c00
...
$(gdb)continue
...
$(gdb)info registers
...
$(gdb)si
...
```

实际上在后续的实验中，你可能很难通过程序计数器添加断点，而我们生成的可执行文件也可能没有符号表信息，这时候就需要使用

```
$(gdb)file example  # 可执行文件
```

这样就支持使用行号、函数名等方式添加断点，具体内容请参考gdb手册或自行查阅资料

1.4. 简单上手

在这一小节，我们会DIY一个主引导扇区，然后用qemu启动它，并在屏幕上输出Hello, World!

在配置好实验环境之后，先建立一个操作系统实验文件夹存放本实验代码

```
$mkdir OS2020
```

进入创建好的文件夹，创建一个mbr.s文件

```
$cd OS2020
$touch mbr.s
```

然后将以下内容保存到`mbr.s`中

```
.code16
.global start
start:
    movw %cs, %ax
    movw %ax, %ds
    movw %ax, %es
    movw %ax, %ss
    movw $0x7d00, %ax
    movw %ax, %sp # setting stack pointer to 0x7d00
    pushw $13 # pushing the size to print into stack
    pushw $message # pushing the address of message into stack
    callw displayStr # calling the display function
loop:
```

```
    jmp loop

message:
    .string "Hello, World!\n\0"

displayStr:
    pushw %bp
    movw 4(%esp), %ax
    movw %ax, %bp
    movw 6(%esp), %cx
    movw $0x1301, %ax
    movw $0x000c, %bx
    movw $0x0000, %dx
    int $0x10
    popw %bp
    ret
```

接下来使用gcc编译得到的mbr.s文件

```
$gcc -c -m32 mbr.s -o mbr.o
```

文件夹下会多一个mbr.o的文件，接下来使用ld进行链接

```
$ld -m elf_i386 -e start -Ttext 0x7c00 mbr.o -o mbr.elf
```

我们会得到mbr.elf文件，查看一下属性

```
$ls -al
...
-rwxr-xr-x  1 kingxu kingxu 3588 2月  15 19:50 mbr.elf
-rw-r--r--  1 kingxu kingxu  656 2月  15 19:46 mbr.o
-rw-r--r--  1 kingxu kingxu  594 2月  15 19:43 mbr.s
```

我们发现mbr.elf的大小有3588byte，这个大小超过了一个扇区，不符合我们的要求

> 不管是i386还是i386之前的芯片，在加电后的第一条指令都是跳转到BIOS固件进行开机自检，然后将磁盘的主引导扇区（Master Boot Record, MBR；0号柱面，0号磁头，0号扇区对应的扇区，512字节，末尾两字节为魔数`0x55`和`0xaa`）加载到0x7c00。

所以我们使用objcopy命令尽量减少mbr程序的大小

```
$objcopy -S -j .text -O binary mbr.elf mbr.bin
```

再查看，发现mbr.bin的大小小于一个扇区

```
$ls -al mbr.bin
-rwxr-xr-x 1 kingxu kingxu 65 2月  15 20:03 mbr.bin
```

然后我们需要将这个mbr.bin真正做成一个MBR，新建一个genboot.pl文件

```
$touch genboot.pl
```

将以下内容保存到文件中

```
#!/usr/bin/perl

open(SIG, $ARGV[0]) || die "open $ARGV[0]: $!";

$n = sysread(SIG, $buf, 1000);

if($n > 510){
  print STDERR "ERROR: boot block too large: $n bytes (max 510)\n";
  exit 1;
}

print STDERR "OK: boot block is $n bytes (max 510)\n";

$buf .= "\0" x (510-$n);
$buf .= "\x55\xAA";

open(SIG, ">$ARGV[0]") || die "open >$ARGV[0]: $!";
print SIG $buf;
close SIG;
```

给文件可执行权限

```
$chmod +x genboot.pl
```

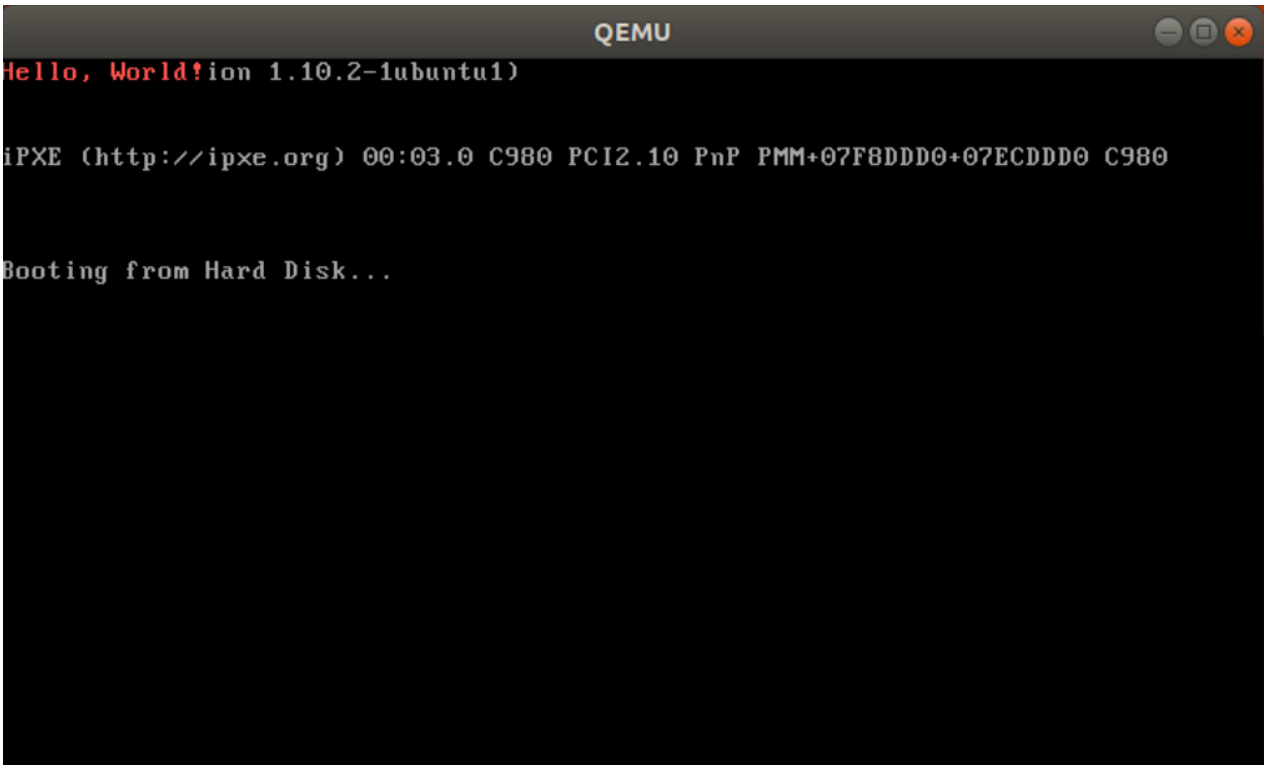
然后利用genboot.pl生成一个MBR，再次查看mbr.bin，发现其大小已经为512字节了

```
$./genboot.pl mbr.bin
OK: boot block is 65 bytes (max 510)
$ls -al mbr.bin
-rwxr-xr-x 1 kingxu kingxu  512 2月  15 20:11 mbr.bin
```

一个MBR已经制作完成了，接下来就是查看我们的成果

```
$qemu-system-i386 mbr.bin
```

弹出这样一个窗口



1.5. 可能的坑

- 在操作系统实验过程中，**谨慎打开gcc编译优化**。
- 在没有得到允许的情况下，**不能**在实验中使用任何C标准库。

2. 相关资料

2.1. 相关网站

- 维基网站 http://wiki.osdev.org
- 问答网站 http://stackoverflow.com

2.2. 相关手册

- Intel 80386 Programmer's Reference Manual
- GCC 4.4.7 Manual
- GDB User Manual
- GNU Make Manual
- System V ABI
- Linux Manual Page

2.3. 相关课件

- 课程说明
- Lab1课件
- Lab2课件
- Lab3课件

- Lab4课件
- Lab5课件

3. 实验列表

目前实验具体内容还没有确定，这个是上学年的实验列表用作参考。

3.1. Lab1系统引导

实现一个简单的引导程序

3.2. Lab2系统调用

实现一个简单的系统调用

3.3. Lab3进程切换

实现一个简单的任务调度

3.4. Lab4进程同步

实现一个简单的同步机制

3.5. Lab5文件系统（选做）

实现一个简单的文件系统

4. 作业规范与提交

4.1. 作业规范

- **学术诚信**: 如果你确实无法完成实验, 你可以选择不提交, 作为学术诚信的奖励, 你将会获得10%的分数; 但若发现抄袭现象, 抄袭双方(或团体)在本次实验中得0分.
- 实验源码提交前需清除编译生成的临时文件，虚拟机镜像等无关文件
- 请你在实验截止前务必确认你提交的内容符合要求(格式, 相关内容等), 你可以下载你提交的内容进行确认. 如果由于你的原因给我们造成了不必要的麻烦, 视情况而定, 在本次实验中你将会被扣除该次实验得分的部分分数, 最高可达50%
- 实验不接受迟交，一旦迟交按**学术诚信**给分
- **本实验给分最终解释权归助教所有**

4.2. 提交格式

每次实验的框架代码结构如下

```
labX-STUID              #修改文件夹名称
├── labX
│   ├── Makefile
│   └── ...
└── report
    └── 171220000.pdf      # 替换为自己的实验报告
```

- labX中的X代表实验序号，如lab1，labX/目录存放最终版本的源代码、编译脚本
- report/目录存放实验报告，要求为pdf格式
- 在提交作业之前先将STUID更改为**自己的学号**，例如`mv lab1-STUID lab1-171220000`
- 然后用`zip -r lab1-171220000.zip lab1-171220100`将`lab1-171220100`文件夹压缩成一个zip包
- 压缩包以**学号**命名，例如`lab1-171220000.zip`是符合格式要求的压缩包名称
- 为了防止出现编码问题, 压缩包中的所有文件名都不要包含中文
- 我们只接受pdf格式, 命名只含学号的实验报告, 不符合格式的实验报告将视为没有提交报告. 例如`171220000.pdf`是符合格式要求的实验报告, 但`171220000.docx`和`171220000张三实验报告.pdf`不符合要求
- 作业提交网站 http://cslabcms.nju.edu.cn

4.3. 实验报告内容

你**必须**在实验报告中描述以下内容

- **姓名、学号、邮箱**等信息, 方便我们及时给你一些反馈信息
- 实验进度。简单描述即可，例如"我完成了所有内容"，"我只完成了xxx"。缺少实验进度的描述，或者描述与实际情况不符，将被视为没有完成本次实验
- 实验结果。贴图或说明都可，不需要太复杂，确保不要用其他同学的结果，否则以抄袭处理
- 实验修改的代码位置，简单描述为完成本次实验，修改或添加了哪些代码。不需要贴图或一行行解释，大致的文件和函数定位就行

你可以**自由选择**报告的其它内容。你不必详细地描述实验过程，但我们鼓励你在报告中描述如下内容:

- 你遇到的问题和对这些问题的思考
- 对讲义或框架代码中某些思考题的看法
- 或者你的其它想法, 例如实验心得, 对提供帮助的同学的感谢等

认真描述实验心得和想法的报告将会获得分数的奖励；思考题选做，完成了也不会得到分数的奖励，但它们是经过精心准备的，可以加深你对某些知识的理解和认识。如果你实在没有想法，你可以提交一份不包含任何想法的报告，我们不会强求。但请**不要**

- 大量粘贴讲义内容
- 大量粘贴代码和贴图, 却没有相应的详细解释(让我们明显看出来是凑字数的)

来让你的报告看起来十分丰富，编写和阅读这样的报告毫无任何意义，你也不会因此获得更多的分数，同时还可能带来扣分的可能。