

东莞理工学院

操作系统课程设计报告

院 系： 计算机学院

班 级： 14 软卓

姓 名： 赖键锋

学 号： 201441402130

指导老师： 李伟

日 期： 2016.6 - 2016.7

一、关于操作系统课程设计的相关说明

关于文档撰写的顺序的说明：（每篇文档分为五个部分分别撰写）

1. 相关说明
2. 相关知识的记录和说明
3. 程序关系图或流程图
4. 运行过程及理解
5. 重点知识总结

学习目的：

1. 从实际代码了解操作系统的运行机制。
2. 了解 github 及 Linux 下常用工具的使用。
3. 学习编写技术文档。

学习任务：

1. 认真阅读《一个操作系统的实现》，搭建运行环境；
2. 运行书中程序，划分程序模块；
3. 给出程序注释，并画程序关系图和流程图。

计划的学习过程：

分为八个阶段，每个阶段撰写一篇实验报告文档：

1. 准备工作
2. 保护模式
3. 内核雏形
4. 进程
5. 输入输出系统

6. 进程间通信
7. 文件系统
8. 内存管理

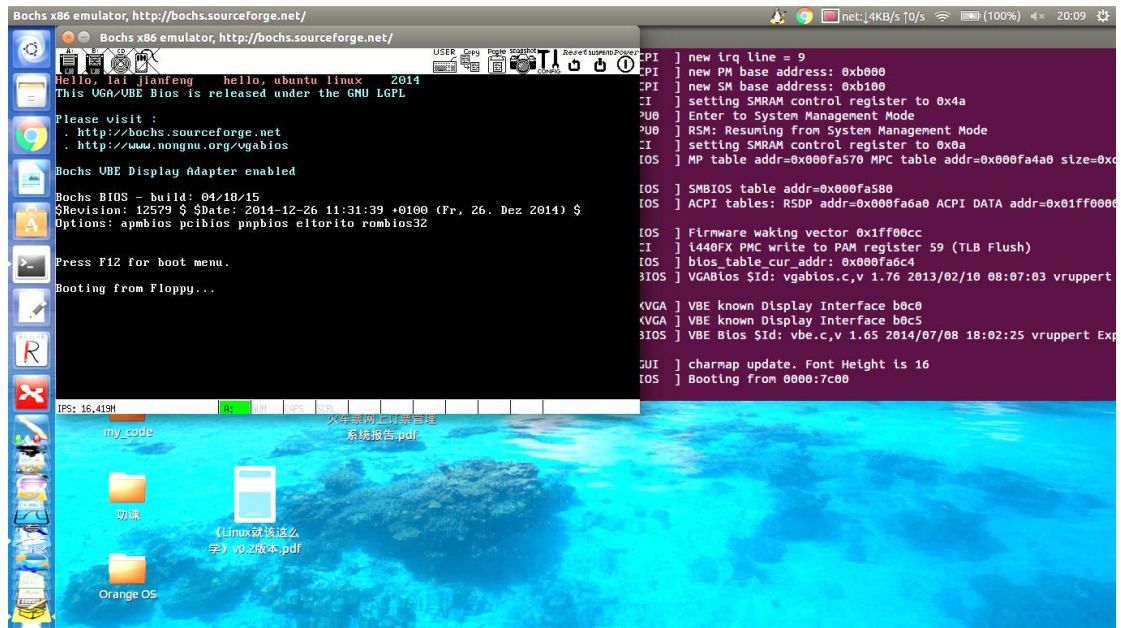
学习环境及工具：

1. 机 器：lenovo G510
2. 系 统：Ubuntu14.04
3. 代码托管：github: https://github.com/whirllys/ORANGE_OS
4. 思维导图：Xmind
5. Markdown：retext
6. 其 他：bochs, bximage, nasm, dd, 截图工具等。

其它说明：

1. 《一个操作系统的实现》要求阅读人员需具备汇编及 C 语言的编程经验，汇编语言我看的是王爽的《汇编语言》；
2. Linux 入门：《Linux 就该这么学》，《鸟哥的 Linux 私房菜》
3. 学 习 git ： 廖 雪 峰 的 git 教 程 ：
<http://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000>
4. 了解 markdown 语法。
5. 本文档的阅读对象：默认是与我年纪相仿的 IT 专业的同学。

我的工作环境：



二、相关知识的记录和说明

计算机开机启动过程:

1. 计算机电源打开, CS: IP 指向 BIOS 程序的首地址, 开始执行 BIOS, 这是一段硬件自检程序;
2. 自检完后它会寻找启动盘, 根据 bochsrc 的设置, 将会从软盘 a.img 程序。计算机会检查软盘的第一个扇区, 如果发现它是以 0xAA55 结束, 且包含一段少于 512 字节的执行码, 则 BIOS 认为它是一个引导扇区;
3. 根据代码的开头: org 07c00h, 这段 512 字节的程序将被装载到 0000:7c00h 处, 然后 CPU 控制权跳转到该程序处, 开始执行引导扇区程序;
4. 这段程序是一段汇编代码:

```

1      org 07c00h          ; 告诉编译器程序加载到7c00处
2      mov ax, cs
3      mov ds, ax
4      mov es, ax
5      call DispStr        ; 调用显示字符串例程
6      jmp $              ; 无限循环
7  DispStr:
8      mov ax, BootMessage
9      mov bp, ax          ; ES:BP = 串地址
10     mov cx, 16          ; CX = 串长度
11     mov ax, 01301h      ; AH = 13, AL = 01h
12     mov bx, 000ch       ; 页号为0 (BH = 0) 黑底红字 (BL = 0Ch, 高亮)
13     mov dl, 0
14     int 10h            ; 10h 号中断
15     ret
16  BootMessage:          db "Hello, OS world!"
17  times 510-($-$$)      db 0 ; 填充剩下的空间, 使生成的二进制代码恰好为512字节
18  dw 0xaa55             ; 结束标志
19

```

主要是把 ES: BP 指向字符串 BootMessage 的地址, 然后设置颜色和串长度等, 然后调用 int 10h (BIOS 中断) 来显示字符串。

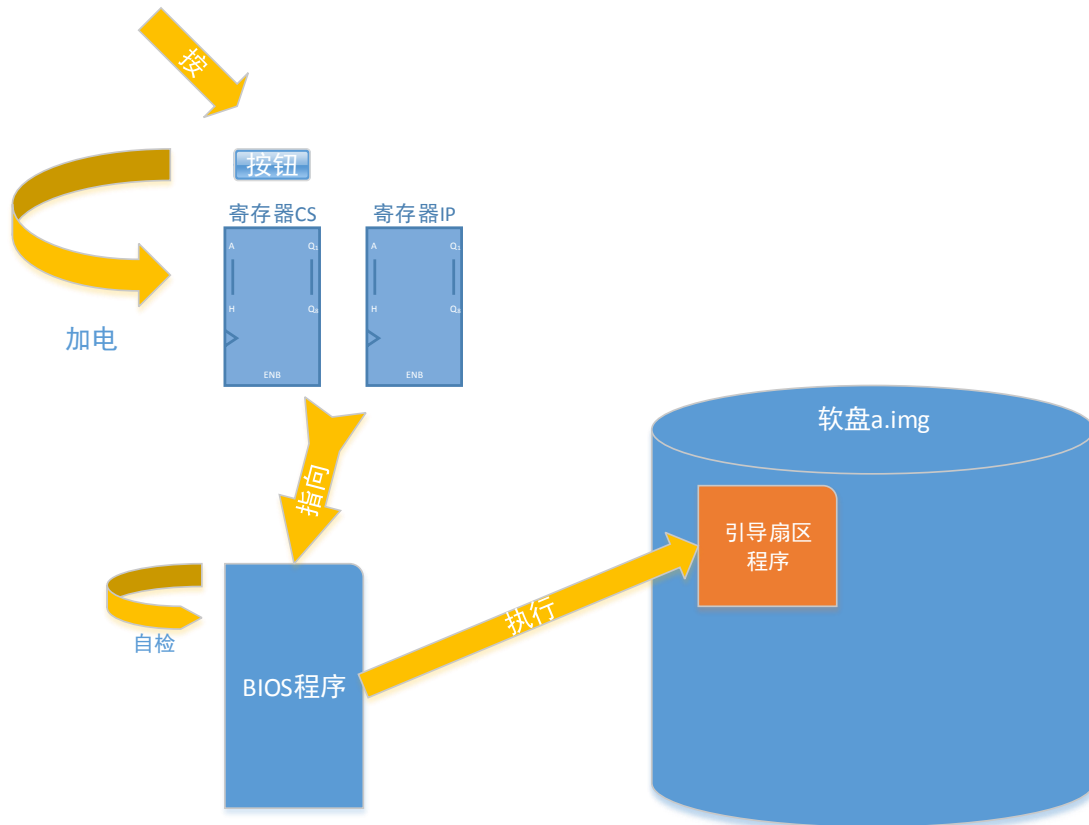
bochsrc 注意: 在我的机器上, 第 10 行需添加 file=, 最后一行应该改为: keyboard: keymap=/usr/share/bochs/keymaps/x11-pc-us.map, 这样才能运行。

提示: 在实验过程中我所遇到的错误及解决办法都记录在我 github:

https://github.com/whirllys/ORANGE_OS 上。

三、程序关系图或流程图

开机过程:



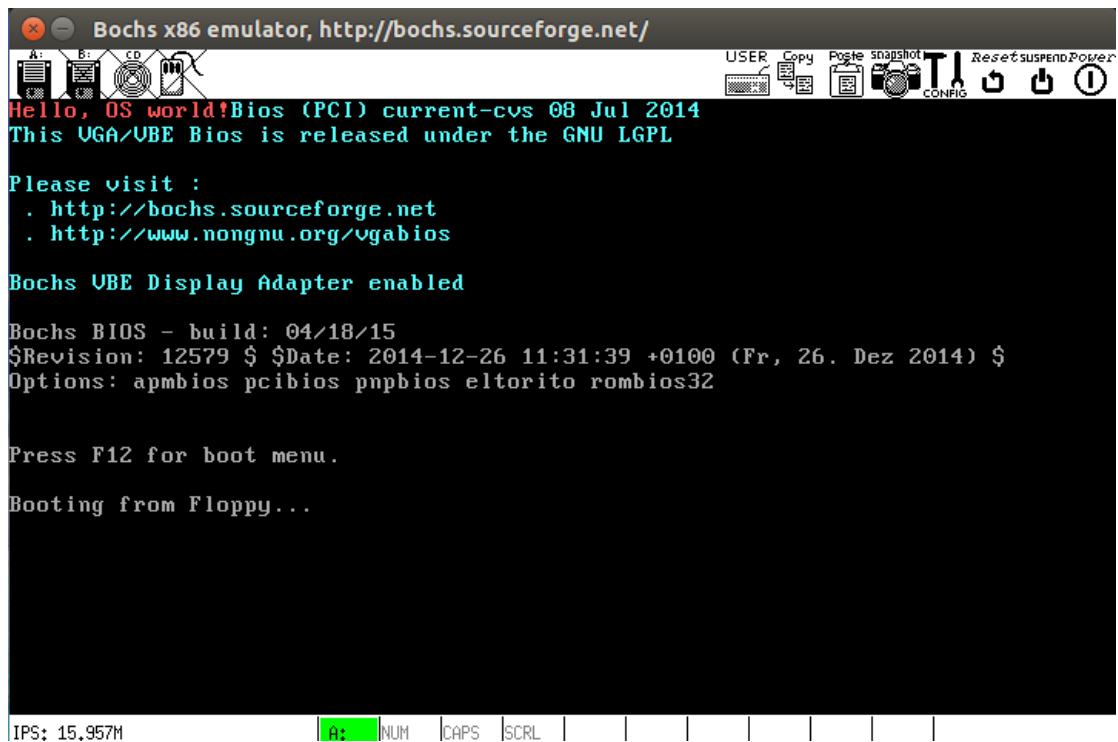
四、运行过程及理解

“最小的操作系统启动:” 步骤:

1. 生成一个虚拟软盘 a.img: `bximage`
2. 编译汇编程序, `nsam boot.asm -o boot.bin`
3. 将二进制文件写入虚拟软盘的第一个扇区:

```
dd if=boot.bin of=a.img bs=512 count=1
```

4. 编写 bochs 虚拟机配置文件 bochsrc:
5. 启动虚拟机: `bochs`
6. 调试: 在 shell 按 c
7. 结果:



The screenshot shows the Bochs x86 emulator window. The title bar reads "Bochs x86 emulator, http://bochs.sourceforge.net/". The window contains a BIOS boot screen with the following text: "Hello, OS world! Bios (PCI) current-cvs 08 Jul 2014", "This UGA/UBE Bios is released under the GNU LGPL", "Please visit :", ". http://bochs.sourceforge.net", ". http://www.nongnu.org/vgabios", "Bochs UBE Display Adapter enabled", "Bochs BIOS - build: 04/18/15", "\$Revision: 12579 \$ \$Date: 2014-12-26 11:31:39 +0100 (Fr, 26. Dez 2014) \$", "Options: apmbios pcibios pnpbios eltorito rombios32", "Press F12 for boot menu.", and "Booting from Floppy...". The status bar at the bottom shows "IPS: 15.957M" and a keyboard layout indicator with "A:" highlighted.

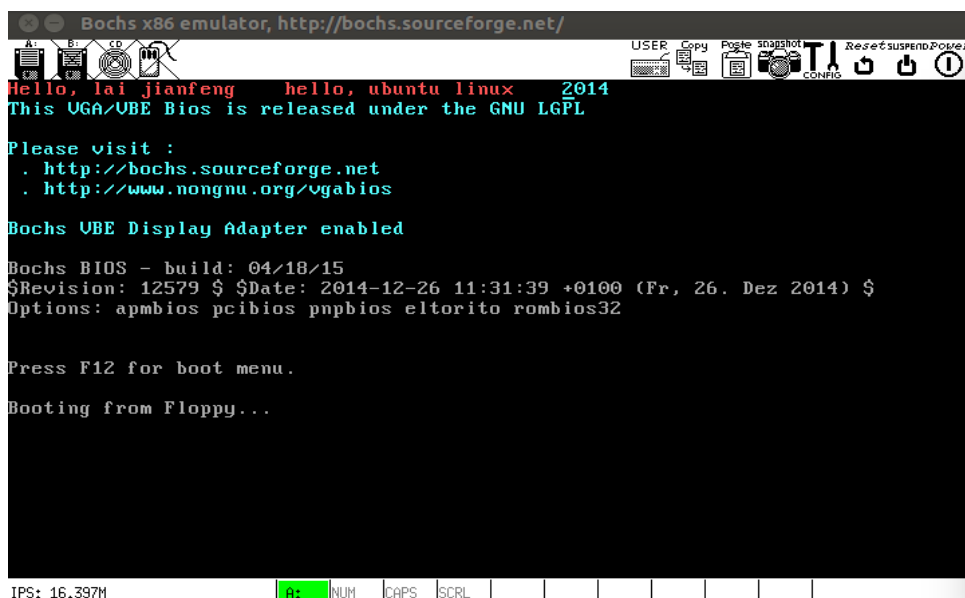
第一行出现了 hello, OS world!

8. 程序修改:

将 hello, OS world, 改成 hello, lai jianfeng hello, Ubuntu linux,

Cx 改成 46;

9. 重新编译运行:



The screenshot shows the Bochs x86 emulator window after modification. The title bar is the same. The BIOS boot screen text is: "Hello, lai jianfeng hello, ubuntu linux 2014", "This UGA/UBE Bios is released under the GNU LGPL", "Please visit :", ". http://bochs.sourceforge.net", ". http://www.nongnu.org/vgabios", "Bochs UBE Display Adapter enabled", "Bochs BIOS - build: 04/18/15", "\$Revision: 12579 \$ \$Date: 2014-12-26 11:31:39 +0100 (Fr, 26. Dez 2014) \$", "Options: apmbios pcibios pnpbios eltorito rombios32", "Press F12 for boot menu.", and "Booting from Floppy...". The status bar at the bottom shows "IPS: 16.397M" and the same keyboard layout indicator.

五、重点知识总结

- 1. 计算机开机启动过程
- 2. bochs 的使用：bochsrc 配置文件和调试

Table 1. 部分Bochs调试指令

行为	指令	举例
在某物理地址设置断点	b addr	b 0x30400
显示当前所有断点信息	info break	info break
继续执行，直到遇上断点	c	c
单步执行	s	s
单步执行（遇到函数则跳过）	n	n
查看寄存器信息	info cpu r fp sreg creg	info cpu r fp sreg creg
查看堆栈	print-stack	print-stack
查看内存物理地址内容	xp /nuf addr	xp /40bx 0x9013e
查看线性地址内容	x /nuf addr	x /40bx 0x13e
反汇编一段内存	u start end	u 0x30400 0x3040D
反汇编执行的每一条指令	trace-on	trace-on
每执行一条指令就打印CPU信息	trace-reg	trace-reg on

- 3. 引导扇区的格式: 位置在软盘的 0 面 0 磁道 1 扇区，512 字节，以 0xAA55 结尾。