中山大学数据科学与计算机学院
操作系统实验课程

实 验 报 告

教　　师　凌应标

学　　号　17341038

姓　　名　傅畅

实验名称　实验项目二：加载用户程序的监控程序

实 验 二
加载用户程序的监控程序

姓名：傅畅　　**学号：**17341038

邮箱：fuch8@mail2.sysu.edu.cn

实验时间：周五（3-4 节）

目录

一、 实验目的

1) 设计四个（或更多）有输出的用户可执行程序

2) 修改参考原型代码，允许键盘输入

3) 自行组织映像盘的空间存放四个用户可执行程序

二、 实验要求

（一）设计四个有输出的用户可执行程序

分别在屏幕 1/4 区域动态输出字符，如将用字符‘A’从屏幕左边某行位置 45 度角下斜射出，保持一个可观察的适当速度直线运动，碰到屏幕相应 1/4 区域的边后产生反射，改变方向运动，如此类推，不断运动；在此基础上，增加你的个性扩展，

（二）允许键盘输入

用于指定运行这四个有输出的用户可执行程序之一，要确保系统执行代码不超过 512 字节

三、 实验方案

（一）引导程序的编写

3.1.1 支持用户输入指令

BIOS 中所支持的 16h 中断，其 00 功能阻塞输入，可以用来等待用户输入一个字符。

3.1.2 根据键盘输入指令跳转至对应程序

事先确定好每个用户程序在内存中加载的位置，根据用户输入的字符，使用 call 指令调用对应的目的地址即可

（二）用户程序编写

3.2.1 增加参数以代码复用

其实在 16h 中，01 功能用来检查键盘缓冲区。与 00 功能通国判断语句组合以下，就能非阻塞地输入了。

3.2.2 支持跳回主程序

在用户程序执行的过程中，使用非阻塞的输入，判断是否为特殊的约定字符，并决定是否跳转。

四、 汇编代码编写

（一）程序关键模块说明

- Loader Start

Start 段主要完成引导程序的显示

```
1                   mov ax,0x0003
2                   int 0x10
3
4                   mov ax, cs
5                   mov ds, ax
6                   mov bp, Message
7                   mov ax, ds
8                   mov es, ax
9                   mov cx, MessageLength
10                  mov ax, 0x1301
11                  mov bx, 0x07
12                  mov dh, 0
13                  mov dl, 0
14                  int 10h
```

代码 1: Loader Start 段

- getchar

等待用户输入一个字符，并决定所跳转的用户程序段，加载对因扇区

```
1           Input:
2           mov ax, 0x0100
3           int 16h
4           je Input
5               mov ax, 0x0
6               mov [Choose],ax
7               int 16h
8               cmp al,'1'
9               je LoadnEx
10              inc byte [Choose]
11              cmp al,'2'
12              je LoadnEx
13              inc byte[Choose]
14              cmp al,'3'
15              je LoadnEx
16              inc byte [Choose]
17              cmp al,'4'
18              je LoadnEx
19          jmp Input
```

代码 2: Loader Input 段

- LoadnEx

获取输入后，加载扇区并跳向指定地址

```
1       LoadnEx:
```

```
2         mov ax, cs
3         mov es, ax
4         mov bx, OffSetOfUserPrg
5         add bx, [Choose]
6         add bx, [Choose]
7         mov ax, [bx]
8         mov bx, ax
9         mov al, [Choose]
10        mov cl, 2
11        add cl, al
12        mov ah, 2
13        mov al, 1
14        mov dl, 0
15        mov dh, 0
16        mov ch, 0
17
18        int 13h
19
20        call bx;???
21        jmp Start
```

代码 3: Loader LoadnEx 段

- 用户程序中输入

用户程序在“弹跳”的过程中，非阻塞地等待输入，如果该字符为’#’, 就 ret, 跳回引导程序。

```
1         mov ax, 0x0100
2         int 16h
3         je endpress
4             mov ax,0x0
5             int 16h
6             mov byte [es:0x20],al
7             mov byte [es:0x21],0x04
8             cmp al, '#'
9             je Return
10        endpress:
```

代码 4: user.asm show 段

（二）生成引导盘

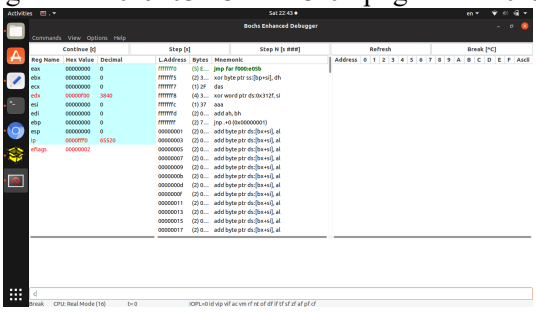
使用 nasm -f bin *.asm -o * 分别生成 fd0, vfuser1, vfuser2,vfuser3,vfuser4，即引导程序和用户程序的二进制文件然后使用 cat vfuser1» fd0，将用户程序简单拼接在引导程序后即可

五、 实验过程与结果

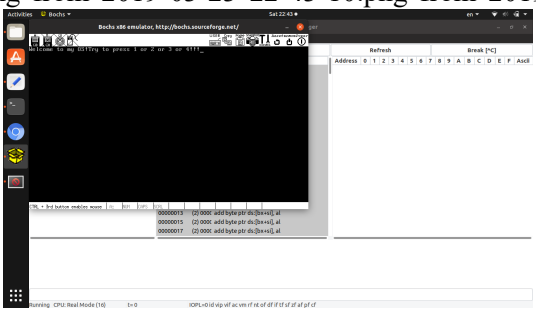
（一）实验执行过程与结果

from 2019-03-23 22-42-23.png from 2019-03-23 22-42-23.png from 2019-03-23 22-42-23.bb from 2019-03-23 22-4

(a) 编译

from 2019-03-23 22-43-02.png from 2019-03-23 22-43-02.png from 2019-03-23 22-43-02.bb from 2019-03-23 22-43-02.bb 

启动之后：

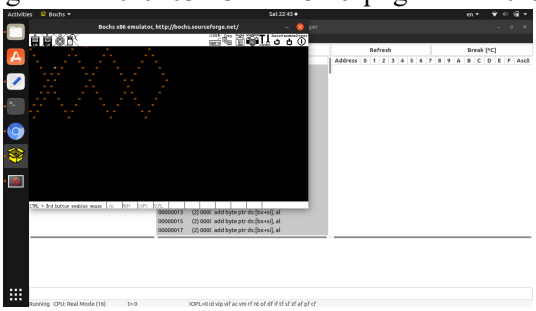
from 2019-03-23 22-43-10.png from 2019-03-23 22-43-10.png from 2019-03-23 22-43-10.bb from 2019-03-23 22-43-10.bb 

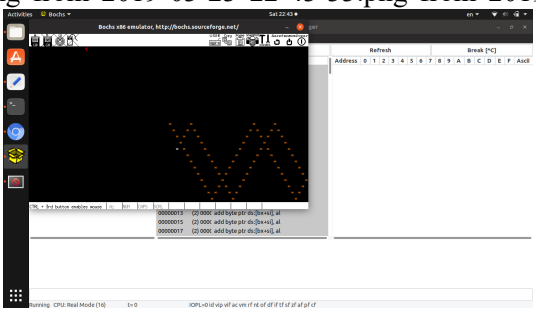
（二）疑难问题解决

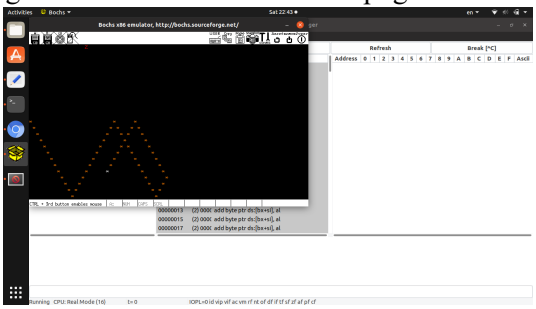
• 代码版本管理不善，一些 bugs 被反反复复的在各个不同的文件夹中的同一份代码里 debug，效率及其底下

• 对中断功能的了解不够，带着误解去使用这些中断的功能导致自己浪费大量的时间。

• 用户程序的起始物理地址没有很好的在各个用户程序中妥善修改，导致用户程序不能被正确执行。

from 2019-03-23 22-43-19.png from 2019-03-23 22-43-19.png from 2019-03-23 22-43-19.bb from 2019-03-23 22-43-19.bb 

from 2019-03-23 22-43-33.png from 2019-03-23 22-43-33.png from 2019-03-23 22-43-33.bb from 2019-03-23 22-43-33.bb 

from 2019-03-23 22-43-41.png from 2019-03-23 22-43-41.png from 2019-03-23 22-43-41.bb from 2019-03-23 22-43-41.bb 

六、 实验总结

本次实验算是从引导扇区向广阔的内存空间跳跃的一个非常重要的一步。之前经常听到主引导记录这个名词，但是对于它的理解并不深，只知道它在系统加载的时候很重要。现在我想自己写一个操作系统，第一步就是它了。这次实验难度比较大的地方，体现在编码难度开始增加，操作系统“大”而“繁”的标签开始显现出来，对代码实现，工程思维水平开始有一定的挑战。代码管理不善，命名不统一不规范所带来的弊端，开始超过以前它所带给我的一些方便了。所以从下个实验开始，科学严谨的编程习惯就是必不可少的了。

七、 参考文献