操作系统原理实验

实验二

加载用户程序的监控系统

姓名： 吴侃

学号： 14348134

班别： 2014 级计算机系一班

日期： 2016.03.06 – 2016.03.09

目录

一、实验目的

实现一个监控系统,能够运行不同的用户程序,并且能从用户程序返回监控系统。

二、实验要求

设计四个（或更多）有输出的用户可执行程序

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，如同时控制两个运动的轨迹，或炫酷动态变色，个性画面，如此等等，自由不限。还要在屏幕某个区域特别的方式显示你的学号姓名等个人信息。

三、实验环境

物理机操作系统: Arch Linux 4.4.3-1

虚拟机软件: VMware Workstation 12 Pro

虚拟机配置: CPU: i7-4702MQ @ 2.20GHz, 使用单核单线程, 内存:4 MB, 硬盘:32 MB.

实验工具:

汇编工具: NASM 2.11.08

C++编译器: clang++ 3.7.1

四、实验方案

1. 监控系统的执行

监控系统为主引导程序，当机器读取其主引导区时,将监控系统载入内存物理地址: 7c00h 处，验证主引导程序有效后(55aah), jmp 0x0000:0x7c00.

2. 监控程序调用用户程序

a. 将用户程序写入内存

使用 13H 中断的 02H 功能，即读取扇区功能。设置要读取的扇区数量(ah)，驱动器号(dl)，磁头号(dh)，柱面号(ch),起始扇区号(cl)。其中，磁头号和柱面号的起始编号都是 0，扇区号的起始编号为 1.

磁头 0,柱面 0,扇区 1 为监控系统；之后的是用户程序，在这里，

我将用户程序放在磁头 0，柱面 0，扇区 2 之后包括扇区 2 的位置。当设置好参数使用 13H 中断的 02H 功能后，对应扇区中的

程序将被写入一个固定的内存地址(偏移量为UserProgramOffset)。

b. 执行用户程序

这里的实现方式为 :将用户程序写入内存后,将计算机的控制权交给用户程序,即跳转到用户程序存放的地址，并开始执行用户程序。

3. 用户程序的编写

a. 返回监控程序

这里实现了两个软中断, int 20h 和 int 21h. 这里定义: 21h 软中断为直接返回监控程序; 20h 软中断为当按下 Ctrl + Z 时返回监控程序。由于使用了中断, 从裸机程序转为用户程序的改写非常简单，只需要在用户程序的开头加上 org 0a100h 说明用户程序指令将被写入到内存从物理地址 0a100h 起始处，然后在用户程序的循环中加入 int 20h, 即使用软中断以提供返回监控程序功能，使用 jmp 指令跳转回监控程序的监控指令代码段。

b. 用户程序实现

Running Ball(45 度飞翔的字符):

在屏幕中设置了运动范围，当字符碰到运动范围边缘时,改变对应的速度。多个字符的运动采用了 nasm 宏实现类 C++函数的技术。

My Name：

使用类似函数的写法，设置画笔的起始点(pos)，运动方向(vel)，以及画的点数(cx).

4. 键盘中断

键盘中断为 16h，当 ah 设置为 01h 时，即检测按键. 当有键被按下, ZF = 0；如果没有键被按下, ZF = 1. ZF 位表示结果是否为零. 使用 jz 可以处理没有按键的状态。

当有键被按下，设置 ah 为 00h，再次调用 16h 中断，按下的键的扫描码存放在 ah 中，ascii
码存放在 al 中。经查表，ctrl + z 的对应扫描码为 2c1ah. 如果不调用 16h 中断的 00h 功能，键盘缓冲区的按键信息将不会被清除。.

5. 中断向量表

实模式下的中断向量表的入口点集中存放在内存从物理地址

0x00000 开始，到 0x003ff 结束. 每个中断在中断向量表中占 2 个字，分别为中断处理程序的偏移地址和段地址. 共 256 个中断, 编号为 i 的中断的入口点位于物理地址 I * 4 处。

6. 写入镜像

一个 1.44M 的软盘，它有 80 个磁道，每个磁道有 18 个扇区，两面都可以存储数据。软盘镜像中同面同磁道的相邻扇区连续存储。

因此使用 C++编写了一个简易的文件合成程序 Writer, 使用：

./writer 14348134os wkcn1 wkcn2 wkcn3 wkcn4 kan

命令, 即可将这些文件依次合并到 disk.img 文件。

五、实验操作

使用 nasm 编译出各个汇编程序，再使用自己编写的 writer 程序合并。即输入命令：./writer 14348134os wkcn1 wkcn2 wkcn3 wkcn4 kan， 生成一个 disk.img 文件。

将 disk.img 文件加载到虚拟机的软盘驱动器，运行虚拟机。

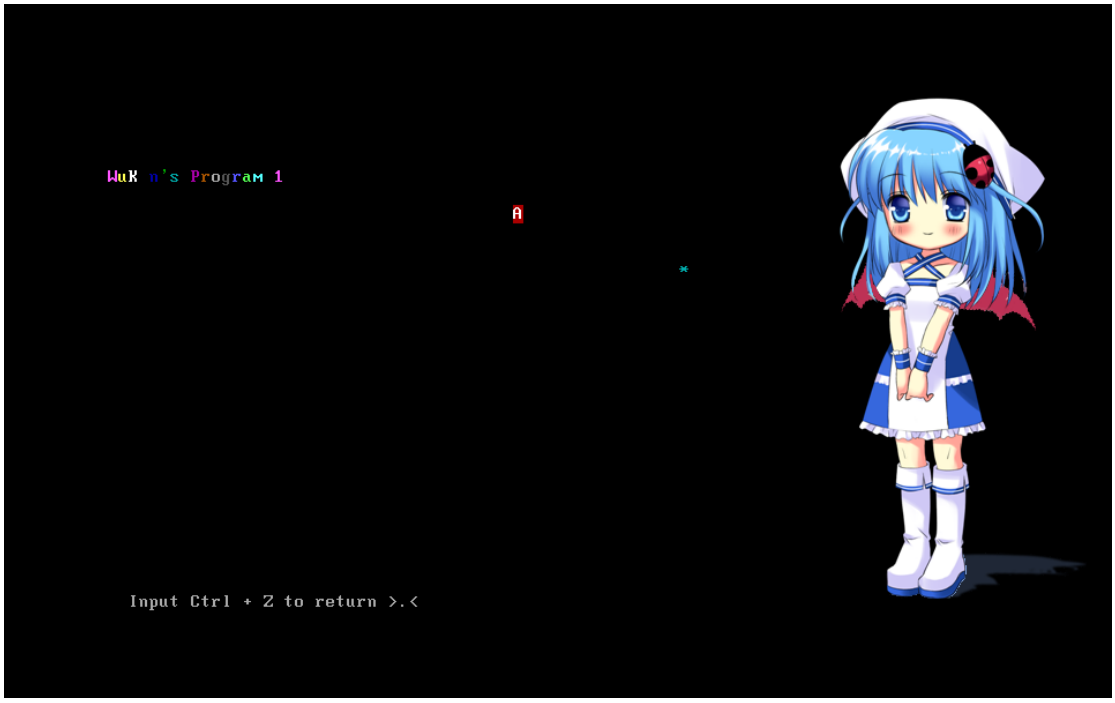
首先进入监控程序界面，如图一：

图一 监控程序界面

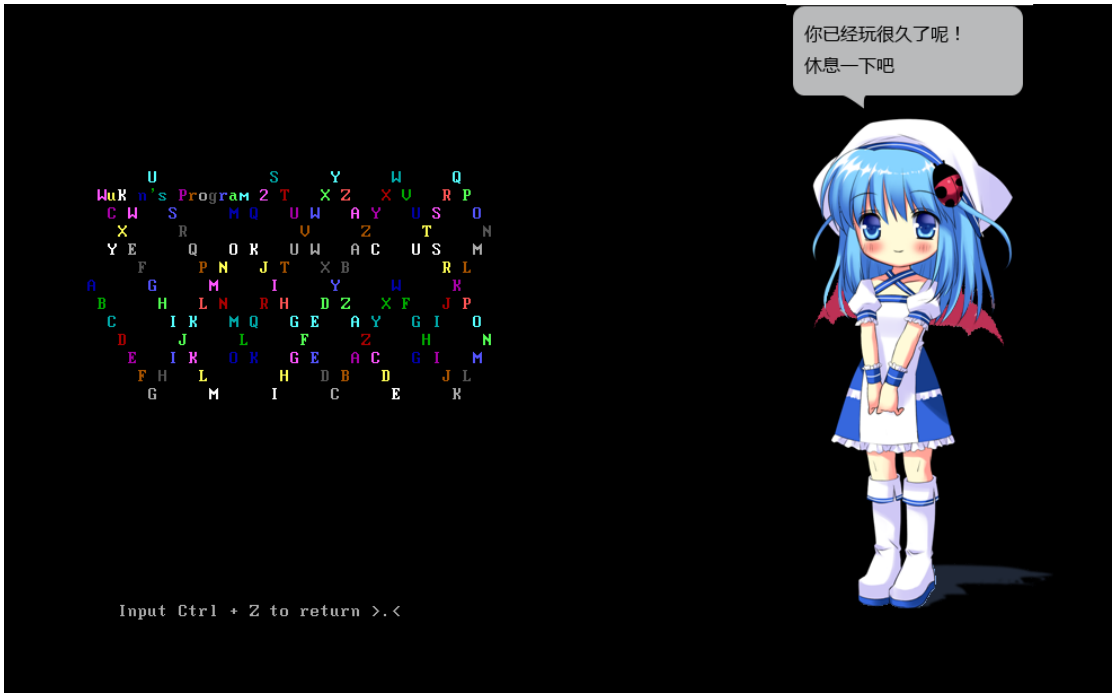
输入数字 1 到 5， 可以选择对应的用户程序。

其中 1 到 4，为飞翔的 45 度字符在四个象限中的程序 ,如图二、三、四、五所示。每个字符都会在其规定象限内运动。有几个字符起始不在规定象限内，但最终会被约束在该象限内。

5 为我的名字的彩色绘制，如图六。

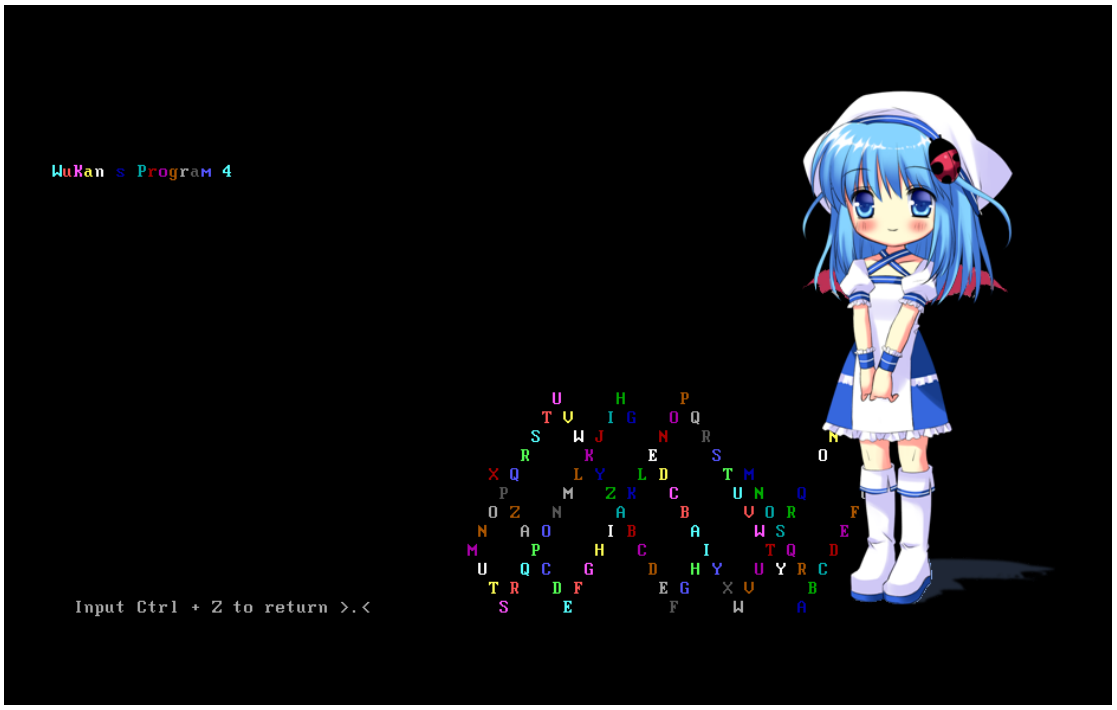


图二 第一象限两个字符运动

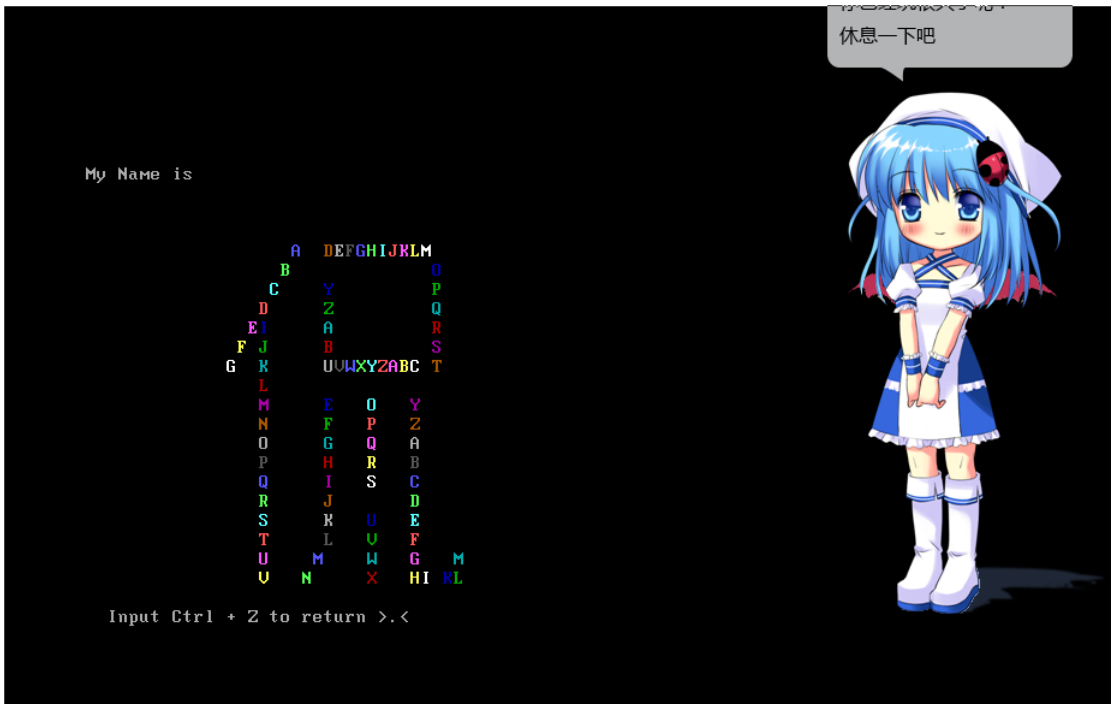


图三 第二象限一个彩色字符运动

图四 第三象限三个字符运动



图五 第四象限两个彩色字符运动



图六 动态写出我的名字

在用户程序执行的过程中， 按下 Ctrl + Z， 可以返回监控程序。

六、实验总结

我觉得这次的实验，难点在于如何在监控程序中调用用户程序，并且能从用户程序中返回。

我开始的做法是，使用 call 调用用户程序，并且使用 ret 返回。但这样有一个不足，即用户程序需要编写一些代码以提供返回判断、返回功能。而且移植性不好，比如当要修改返回判断时，需要修改大量的程序。

因此，我想能不能让用户程序能很方便地从裸机程序修改为用户程序，即将返回处理命令写在监控程序中，用户程序只需简单地调用

监控程序中的指令，我使用了**软中断**来解决这个问题。

经查资料，软中断由**中断向量表**管理，实模式下从内存的物理地址 0x00000 开始，到 0x003ff 结束. 将中断程序的偏移地址和段地址写入对应内存即可。

我在监控程序写了 20h 和 21h 中断，将它们加入中断向量表中。这样，用户程序只需简单修改就可以与监控程序良好地切换了。在设计过程中，我也遇到了一些小问题，比如用错寄存器，没有注意到一个变量修改后会影响到其它区域，需要用栈保护一些寄存器，调试了很久才发现错误发生在其它位置。

我觉得汇编中，尤其要知道寄存器的值在什么时候，哪个例程中会被改变。

我也思考了**多任务系统的实现方式**，也查找了相关资料. 我的想法是:

将多个程序读入不同的内存位置，每个程序的起始位置用于保存各个寄存器，当前执行的指令位置. 然后在每个用户程序中调用一个中断，返回监控程序后，存储当前寄存器状态，让监控程序执行另一个用户程序，这样不断交替执行。

我遇到的问题：由于用户程序的起始点不一致，而且用户程序无法预知其被写入的内存地址，应如何设置用户程序中的段地址。这个问题我将会在之后尝试解决。

参考文献：

1. <x86 PC 汇编语言，设计与接口> - 作者：（美国）马兹迪（Muhammad Ali Mazidi）（美国）考西（Danny Causey）（美国）马兹迪（Janice Gillispie Mazidi） 译者：高升 合著者：王筱珍
2. <x86 汇编语言-从实模式到保护模式> - 李忠 王晓波 余 洁 著
3. 键盘组合键扫描码 - 瘦皮猴的日志 - 网易博客

附录

文件描述：

文件名	描述
disk.img	监控程序和用户程序的软盘镜像
14348134os.asm	监控程序源码
wkcn1.asm	飞翔的 45 度字符第一象限源码
wkcn2.asm	飞翔的 45 度字符第二象限源码
wkcn3.asm	飞翔的 45 度字符第三象限源码
wkcn4.asm	飞翔的 45 度字符第四象限源码
kan.asm	我的名字源码
writer.cpp	合并二进制文件的程序源码