**实验四：中断与系统调用**

**实验者：张海涛，张晗宇，张浩然，张镓伟**

**学号：15352405,15352406,15352407,15352408**

**院系：数据科学与计算机学院**

**专业：15级软件工程（移动信息工程）**

**指导老师：凌应标**

**【实验题目】**

在设计的操作系统中添加中断和系统调用

**【实验目的】**

1. 学习中断机制知识，使用中断处理程序设计。

2. 设计一个汇编程序，实现时钟中断处理程序。

3. 扩展MyOS3，增加时钟中断服务，利用时钟中断实现与时间有关的操作

4. 实现简单的系统调用

**【实验要求】**

1. 操作系统工作期间，利用时钟中断，在屏幕24行79列位置轮流显示’|’、’/’和’\’，适当控制显示速度，以方便观察效果。

2. 编写键盘中断响应程序，原有的你设计的用户程序运行时，键盘事件会做出有事反应：当键盘有按键时，屏幕适当位置显示”OUCH! OUCH!”。

3. 在内核中，对33号、34号、35号和36号中断编写中断服务程序，分别在屏幕1/4区域内显示一些个性化信息。再编写一个汇编语言的程序，作为用户程序，利用int 33、int 34、int 35和int 36产生中断调用你这4个服务程序。

**【实验方案】**

**一.虚拟机配置方法**

无操作系统，10M硬盘，4MB内存，启动时连接软盘

**二.软件工具和作用**

Notepad++:用于生成.汇编语言文件

Nasm：用于编译.asm类型的汇编语言文件，生成bin文件

Vmware Workstation 12Player: 用于创建虚拟机，模拟裸机环境

TCC：用于由.c文件生成.obj文件

TNASM：用于由.asm文件生成.obj文件。

Tlink：用于链接obj文件生产.com可执行文件

**三.实验原理**

（1）中断技术

计算机硬件系统提供中断技术，支持CPU与外部设备的并发工作，也利用中断技术处理硬件错误、支持程序调试、实现软件保护和信息安全等。

中断(interrupt)是指对处理器正常处理过程的打断。中断与异常一样，都是在程序执行过程中的强制性转移，转移到相应的处理程序。

硬中断（外部中断）——由外部（主要是外设[即I/O设备]）的请求引起的中断

时钟中断（计时器产生，等间隔执行特定功能）

I/O中断（I/O控制器产生，通知操作完成或错误条件）

硬件故障中断（故障产生，如掉电或内存奇偶校验错误）

软中断（内部中断）——由指令的执行引起的中断

中断指令（软中断int n、溢出中断into、中断返回iret、单步中断TF=1）

异常/程序中断（指令执行结果产生，如溢出、除0、非法指令、越界）

（2）系统调用

操作系统提供的服务可以用多种方式供用户程序使用

①子程序库静态链接：采用子程序调用方式，如汇编语言中用call指令调用操作系统提供服务的子程序，静态链接到用户程序代码中，这种方式优点是程序执行快，最大缺点是用户程序内存和外存空间占用多，子程序库管理维护工作复杂。

②内核子程序软中断调用：采用软中断方式，如汇编语言中用int指令调用操作系统提供服务的子程序，系统服务的子程序在内核，这种方式的优点是服务由系统提供，程序效率较高，且被所有用户程序代码共享，有利于节省内存，最大缺点是需要防止内核再入或内核设计为可再入，且用户程序陷入内核和内核返回用户程序的开销较大。

③子程序库动态链接：采用动态链接技术，操作系统在运行时响应子程序调用，加载相应的子服务程序并链接致用户地址空间，这种方式优点是可由多方提供服务程序，服务更内容丰富，增加和变更服务方便，最大缺点是链接耗时多，程序响应变慢，实现复杂。

BIOS调用

①与内核子程序软中断调用方式原理是一样的

②每一种服务由一个子程序实现，指定一个中断号对应这个服务，入口地址放在中断向量表中，中断号固定并且公布给用户，用户编程时才可以中断调用，参数传递可以使用栈、内在单元或寄存器。

系统调用

①因为操作系统要提供的服务更多，服务子程序数量太多，但中断向量有限，因此，实际做法是专门指定一个中断号对应服务处理程序总入口，然后再将服务程序所有服务用功能号区分，并作为一个参数从用户中传递过来，服务程序再进行分支，进入相应的功能实现子程序。

②这种方案至少要求向用户公开一个中断号和参数表，即所谓的系统调用手册，供用户使用。

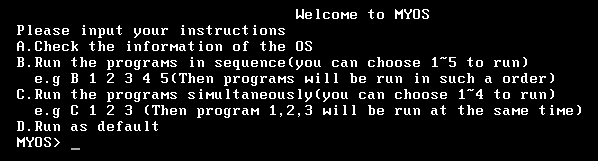
③如果用户所用的开发语言是汇编语言，可以直接使用软中断调用

④如果使用高级语言，则要用库过程封装调用的参数传递和软中断等指令汇编代码

⑤规定系统调用服务的中断号是21h。

**四.程序功能**

开机进入内核，内核提供 3 种指令输入，例如：



**1. 时钟中断：** 在内核运行期间， 会在屏幕右下角24行79列位置循环变

色显示‘|’、‘/’、‘\’。

在进入用户程序之前， 恢复默认中断向量表以停止字符显示；在从用户程序返回内核后， 重新设置中断向量表继续循环变色显示三个字符。

**2. 键盘中断：**在内核进入用户程序前， 修改当前键盘中断向量表， 使用

户程序中每次按键盘都会变色显示“Ouch!Ouch!” ， 且显示位置会不断变成上次显示的位置之后。 从用户程序返回内核后， 恢复默认中断向量表， 以保证内核正常的输入功能。

**3. 系统调用功能：**提供 INT 33H,INT 34H,INT 35H,INT 36H 的系统调用，

可分别在屏幕左上方、 右上方、左下方、右下方显示特定信息。

**4. 显示用户程序的信息：**输入指令A，则屏幕上会显示每个用户程序的相关

信息。

**5. 批处理功能：** 指令 B之后，依次输入想执行的用户程序的编号，范围

1~5，之后程序会依次调用相应的用户程序去执行。其中前4个用户程序为实验3中的4个用户程序，第5个用户程序为新增的准们测试INT 33H,INT 34H,INT 35H,INT 36H4个系统调用的程序。输入指令D会默认依次执行1、2、3、4这4个用户程序。

**6. 分时功能：**指令C之后，依次输入想执行的用户程序的编号，范围1~4，

则程序会同时调用这些用户程序去执行。这里没有程序5是因为程序5

会霸占整个屏幕。

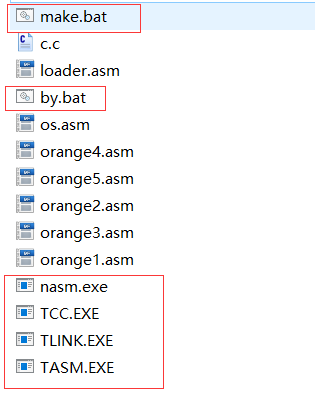
**7. 命令行输入：** 左下的 MYOS>处输入命令， 此处模仿 cmd， 在输入过程中， 若

输入错误， 可以进行删除一定字符以修正输入。如果输入了错误的命令并按

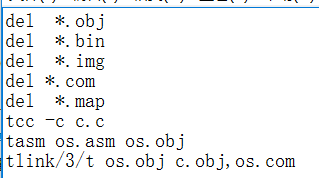
了回车，则会有报错信息。

**五.编译流程**

由于之前几次实验每次编译都要按好多命令，大家深感其烦，所以这次我们使用了脚本完成自动化编译。首先在含代码的文件夹中将新建by.bat 和 make.bat。并将nasm.exe,TCC.exe,TLINK.exe,TASM.exe 复制到该文件夹里。完成后如下图：

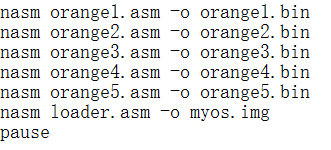


在make.bat文件中键入如下内容：



前面几句del的几句诗删除以\*后面位后缀结尾的那些编译生成的文件，然后是内核的C与汇编的混合编译。

在by.bat中键入如下内容：



这是用来编译用户程序和引导程序的，最后的pause是为了防止命令完成后

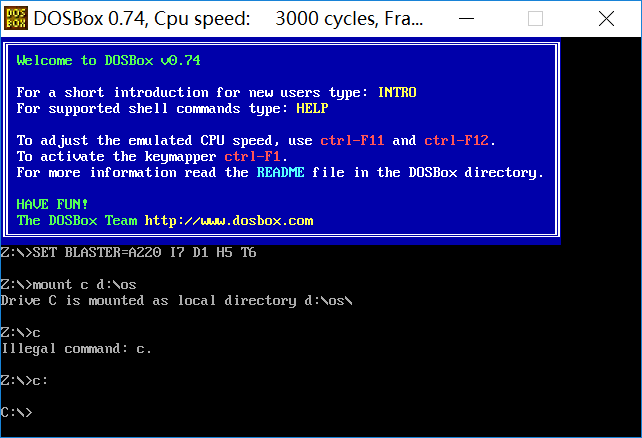
cmd框消失而看不到如果代码有错而产生的错误信息。

之所以不把make.bat和by.bat的内容写在一起是因为,make.bat是要在

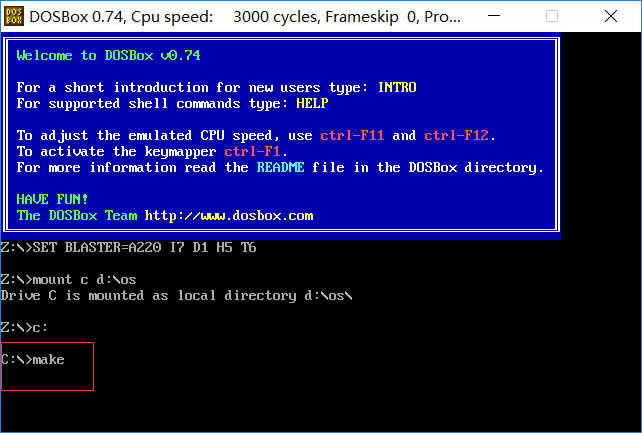
DOSBOX中运行的，而在DOSBOX中无法运行nasm，所以nasm之内另外写在

by.bat中在本机环境下执行。

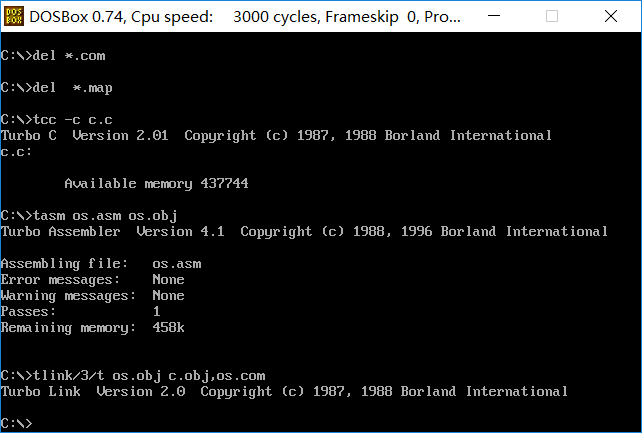
打开DOSBOSX，将这个文件夹创建为一个虚拟盘，在这个虚拟盘下便于操作。



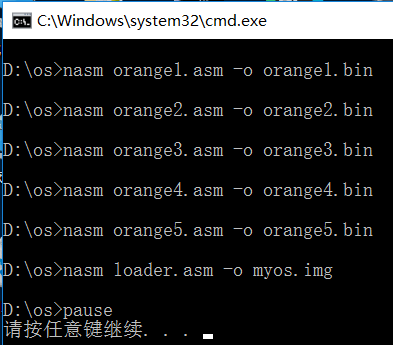
输入命令make并回车执行自动编译。



自动编译效果：



在代码文件目录下双击by.bat继续完成剩下的编译。



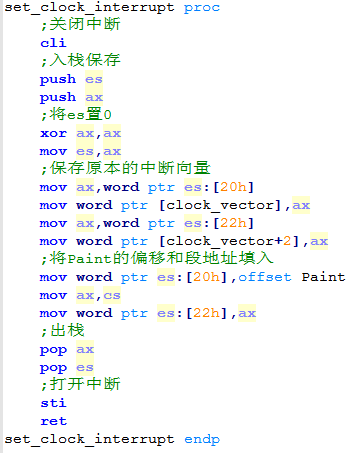
**六.程序关键代码及解释**

【内核】

内核在上一次实验基础上修改，并且增加了时间中断、键盘中断、自己编写的中断程序的内容以及需要用到的一些新函数。

（1）时钟中断

①设置时间中断



**七.程序运行流程**

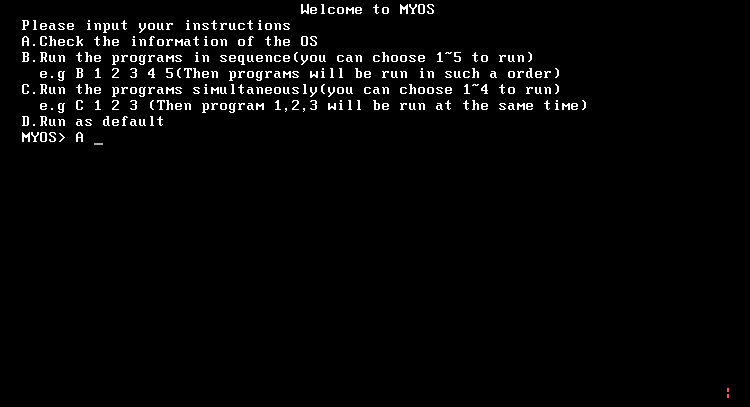
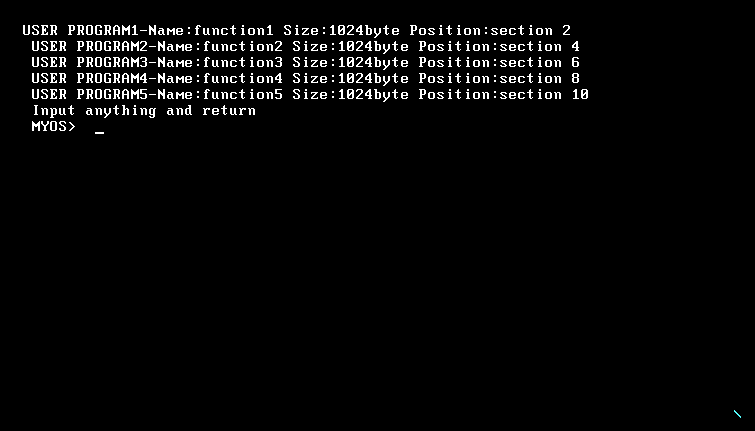
以下为连续的依次操作。

内核界面：

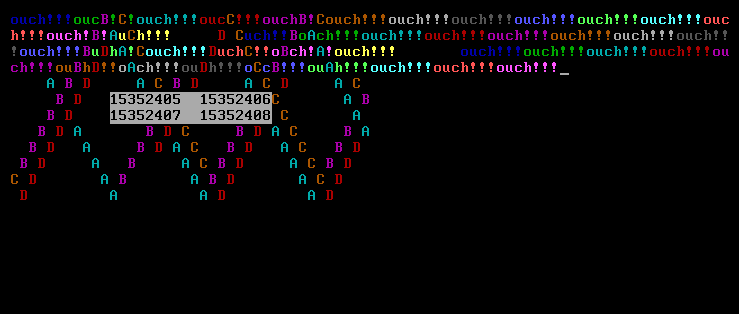
可以看到除了在实验三中已有的内容外，右下角还有一个彩色的字符。



输入命令A，显示程序信息。

按任意键返回再输入B 1 运行用户程序1，可以OUCH!



用户程序1结束后输入任意命令返回再输入B 2 运行用户程序2，可以OUCH!



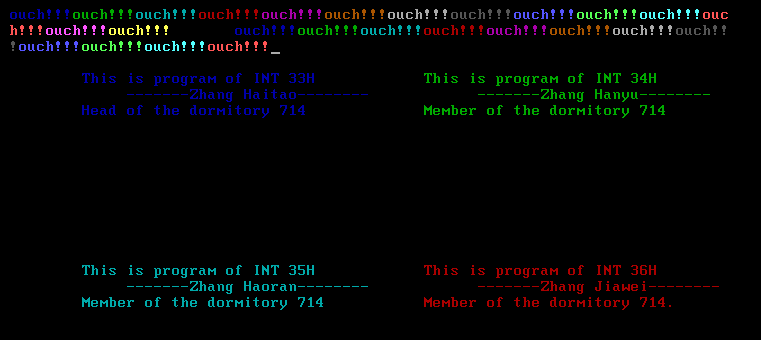
用户程序2结束后输入任意命令返回再输入B 3 运行用户程序3，可以OUCH!



用户程序3结束后输入任意命令返回再输入B 4 运行用户程序4，由于这个程序只是显示一个日期，一瞬间就执行完了，所以是没有机会ouch!的



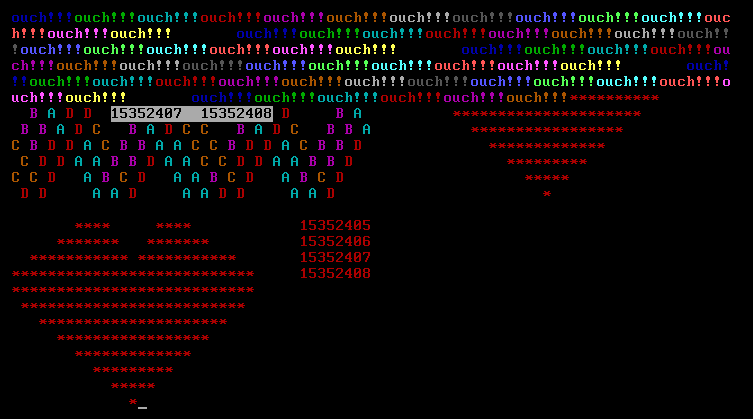
用户程序4结束后输入任意命令返回再输入B 5 运行用户程序5，检测4个系统调用，可以ouch!



用户程序5结束后输入任意命令返回再输入B 1 2 3 4 依次运行用户程序1 2 3 4，实现批处理功能，可以ouch!









批处理结束后输入任意命令返回再输入C 1 2 3 同时运行用户程序1 2 3，实现分时功能，可以ouch!

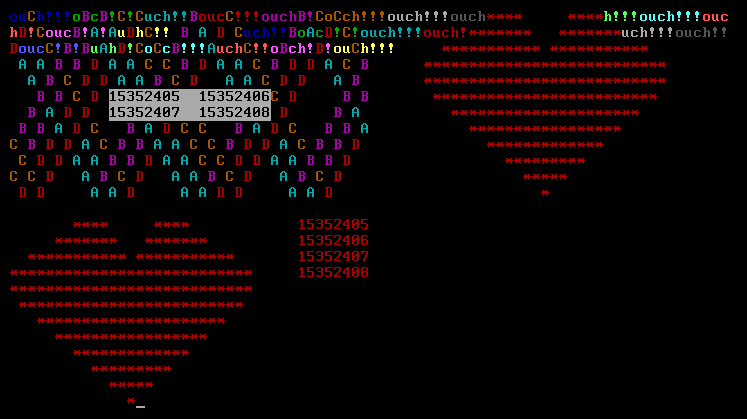


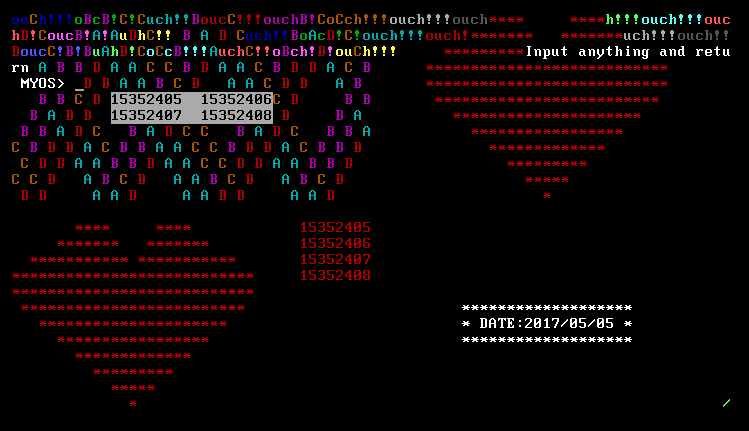
运行结束后任意命令返回再输入D，会按照默认批处理运行用户程序1、2、

3、4。可以ouch！

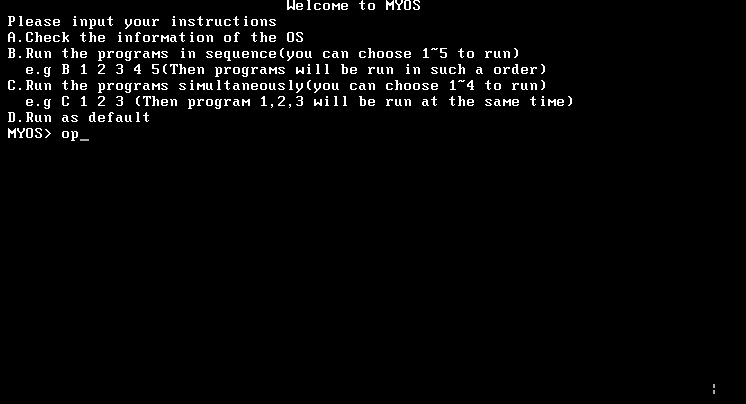








内核界面输入错误命令，会有报错信息：





**【实验总结】**

**张浩然的实验总结：**

这次的实验遇到的困难是比较多的。首先是需要搭建很多的新环境:tcc、tasm、tlink。为了研究如何在我的电脑上运行花了很多工夫，最后使用DOSBOX来运行这些程序。然后这次的要求比较的多，刚看要求不太有头绪哪一个要求具体该用什么样的方式去实现。再者就是这次的实验又提到了内核、磁盘映像之类的新的内容，怎么编写它们也是一个需要解决的困惑。  
这次实验中遇到最大的问题就是我们的同时显示多个用户程序出了问题，它依旧是执行完一个用户程序才往下面走。经过很长时间的研究和尝试后，我们才发现原来问题出在用户程序中。用户程序缺少判断运行模式（依次还是同时）来区别操作的部分，所以无法实现预期的效果。  
由于缺少很多汇编语言的基础，以及汇编语言的思想和习惯了解还不够，每次的实验都是在一边模仿，一边学习，一边尝试的。感觉这样的确有很多地方的感受会很深，但也有一些地方迷迷糊糊就过去了，要是下次依旧遇到了相同的问题或者用法，自己未必能准确而又理解透彻地去运用。可能还是不能忙于单纯的研究每周的实验内容和实验要求，需要抽出更多的时间研读相关的课外参考教材，多向老师咨询。

**张海涛的实验总结：**

这次实验做的事用C语言和汇编语言混合编译操作系统，越来越具有操作系统的雏形了。仔细阅读了参考代码之后，我发现内核中汇编编写的部分是较为底层的功能实现，而c代码主要是负责各用户程序之间的通信，提供上层交互服务。

我们继承上一次写好的用户程序时，遇到一个致命的问题：段地址的设置。子程序的段地址，都要一致设置为磁盘中的存储地址，而有一个子程序需要用到显存，需要设置为0b800h，由于我们忽略了这个问题，导致没有任何显示。

在阅读参考代码的过程中，对于混合编程多了一点认识。

还有关于ptr的用法，查询之后了解到：ptr是临时的类型转换，相当于C语言中的强制类型转换）。在没有寄存器名存在的情况下，既都是在内存，得用操作符 X ptr 指明内存单元的长度，X在汇编指令中可以为byte，word或者DWORD。要不然内存是片连续的区域，操作就乱了。因为ptr指向的都是外部变量或函数，是存储在内存中，使用的时候需要指明格式大小。

本次实验中最困难的部分就是开头第一步：编译。首先是不同编译器的配置，然后是编译流程，最后是编译指令。

配置好编译器，就开始向其他同学了解编译流程，因为不懂得编译原理，所以常常是一头雾水。在一次link时，踩到了一个匪夷所思的坑。tlink编译时是有文件顺序要求的，一直编译失败，就交换了参数顺序，就成功了。这个探索过程还真是挺艰苦。

**张晗宇的实验总结：**

这次的实验虽然是很多周来实现，但是一开始被王师兄的1200行代码吓到了，这次的实验内容主要还是中断程序的调用，在C语言框架部分，加入两个函数，一个用来在右下角显示字符，一个用来在输入时输出ouch，由于对段地址和偏移地址的理解不够到位，使得在最初的尝试中出现乱码现象，不仅仅是用户程序的字符无法显示，时钟中断以及键盘中断的字符显示都出现乱码，最后修改了os的偏移地址以及改变了loader引导程序中的段地址和偏移地址的赋值方式才使字符显示完整，但是用户程序依旧不知如何修改，虽然一开始想用暴力的方法输出字符，但是由于这种做法实质上并没有解决问题，所以一直寻找着其他方法，最后在同学的帮助下，我们修改了用户程序开始的偏移地址才使字符正常显示。由于这次实验涉及到的文件较多，也出现了编译时用错文件的尴尬场景，虽然不是很难，但是由于对偏移地址以及段地址的不熟悉不理解，导致出现了很多的错误，花费了很多的时间。

**张镓伟的实验总结：**

这次的实验过程可以说是一波三折。这次的实验内容是实现中断和系统调

用，有时钟中断，键盘中断和自己编写的4个系统调用。一开始我们选择先完成时钟中断和键盘中断。作为小白我们依然选择先将王师兄的代码移植过来再做修改。但是没想到加入这两个中断之后用户程序的字符显示不全了，而且时钟中断和键盘中断都出现了乱码。我们调试了一个晚上都没有得出结果。后来我们四处求教，有些人说他们是在用户程序直接暴力再输出字符的，然而这种方法并非从本质上解决了问题，我们没有采纳。后来看到微信群中大家说是段地址偏移量的问题，于是我们修改了内核和引导程序的段地址和偏移量以及用户程序中的偏移地址才解决了问题。接着我试着加入第5个用户程序去测试4个中断的内容，然而问题又出现了，加进去之后虚拟机一加载直接就死机了后面发现了是我们增加了扇区数却没有修改起始扇区号导致的，归根究底还是对这些知识不够熟悉导致的。最后程序中其实还有一个小bug，就是我们显示用户程序信息和输错命令的话，本应是在当前界面接着输出相关内容，但不知道为什么现在会先输出一段乱码，调试很久无果，最终我灵机一动，在输出内容前先执行清屏操作不就看不见乱码了吗，一试之下果然可行。这次试验总体来说是同既艰苦又快乐的，苦在debug上，乐在成功后。