实验五 实现系统调用 实验报告

数据科学与计算机学院　2017 级计算机科学与技术　17341146　王程钥

1　实验题目

实现系统调用

2　实验目的

掌握设计系统调用的方法
掌握系统调用的调用方法

3　实验要求

1）在 21h 号中断编写几个系统调用程序，分别实现相关功能。
2）在内核增加一个过程作为系统调用的总入口，用以获取参数和分析功能号，再根据功能号产生分枝结构，根据系统调用号决定选择对应的分支完成相应的服务。通常每个分枝实现一种系统调用功能，简单的功能可以用汇编实现，也可以用 c 程序实现。

4　实验方案

4.1 实验环境

编程环境：Dosbox+TCC+TASM+Tlink,NASM
16 进制编辑器：Hex Editor
虚拟机：WMware Workstation
虚拟机环境

- 操作系统　MS-DOS
- 内存　1MB
- 硬盘　102MB
- 处理器　1 核心

编译命令

```
nasm -f bin guide.asm -o guide.com
tasm myos.asm myos.obj
tcc -mt -c –omain.obj main.c
tlink /3 /t myos.obj main.obj,test.com,
```

4.2 编写系统中断

通过中断程序实现一些个性化的功能，并将其加载到 int21h 的位置。出于编码量的考虑，系统中断采用 TASM+TCC 实现。

以下为我实现的 21h 号中断功能表。

功能号	功能
00h	在随机位置输出“OUCH”
01h	显示一首诗
02h	显示几句歌词

03h	清屏
04h	屏幕上滚
05h	屏幕下滚
4ch	退出操作系统

4.3 系统中断调用总入口设计

根据实验要求，需要在内核中设计一个系统中断调用的总入口程序，用于系统中断的调用及相关功能的测试。我设计了一条“system”命令用于调用系统中断的相关功能，还设计了一个“system.h”库方便系统中断的调用。

4.3 程序存储设计与地址存放

因为内核写得比较大，总是超扇区，所以我将存储扇区位置进行了一些微调，加大了内核的扇区数。

以下为存储扇区位置。

功能	程序名	存储扇区
引导程序	guide.com	1
内核	test.com	2-18
文件夹	file.txt	19
用户程序 1	stone1.com	20
用户程序 2	Stone2.com	21
用户程序 3	stone3.com	22
用户程序 4	stone4.com	23
脚本程序 1	shell1.bat	24
脚本程序 2	shell2.bat	25
8 号中断处理程序	newint8.com	26
9 号中断处理程序	newint9.com	27
21h 号中断处理程序	newint33.com	28~31

以下为内存地址。

功能	地址
引导程序	7c00h
内核	A100h
用户程序	E100h
8 号中断处理程序	9500h
9 号中断处理程序	9900h
33 号中断处理程序	C500h

5 实验过程

本次实验以实验四实现的修改过相关中断程序的内核为基础进行修改，加入了自己编写的系统中断程序，修改或删除了原有的几个中断程序，并修改了操作系统的几条命令。

本次实验删除了实验四中的 34,35,36 号中断，并将 33 号中断修改为系统中断。

本次实验增加了 system 命令，用于中断程序部分功能的调用。

5.1 系统中断汇编部分。

汇编部分将 ax，bx，cx，dx 四个寄存器的值传入 Ax，Bx，Cx，Dx 四个全局变量，后调用系统中断的 main 函数_int21h_main。以下为相关代码。

```
extrn _int21h_main:near
extrn _Ax
extrn _Bx
extrn _Cx
extrn _Dx
.8086
_TEXT segment byte public 'CODE'
DGROUP group _TEXT,_DATA,_BSS
       assume cs:_TEXT
org 0100h
start:
    push ds
    push es
    push bp
    push si
    push di

    sti
    mov si,ax
    mov ax,0c40h
    mov ds,ax
    mov es,ax
    mov [_Ax],si
    mov [_Bx],bx
    mov [_Cx],cx
    mov [_Dx],dx
    call near ptr _int21h_main

    mov al,20h
    out 20h,al
    out 0A0h,al
    pop di
    pop si
    pop bp
    pop es
    pop ds
    iret

_TEXT ends
```

```
;************DATA segment*************
_DATA segment word public 'DATA'
_DATA ends
;*************BSS segment*************
_BSS   segment word public 'BSS'
_BSS ends
;**************end of file***********
end start
```

5.2 在随机位置输出“OUCH!”

当调用系统中断的 00h 号功能时，在屏幕右上角的 1/4 区随机位置输出“OUCH!”。输出位置通过如下公式产生，xx 为横坐标，yy 为纵坐标。

```
xx=(xx*23+rand())%13;
yy=(yy*23+rand())%31;
```

随机数通过系统时钟的秒数实现，通过 int1ah 号中断的 02h 号功能获取。随机数生成程序如下。

```
int rand()
{
  int res;
  asm mov ah,02h
  asm int 1ah
  asm mov res,dx
  return res;
}
```

输出部分直接借用内核的输出函数，因为没有输入功能所以不需要提供退格，上下左右键等功能。代码如下。

```
void putchar(char c,char col)
{
  int pos;
  asm push ax
  asm push bx
  asm push cx
  asm push dx
  if(c>=32)
  {
    asm mov ah,09h
    asm mov al,c
    asm mov bh,0
    asm mov bl,col
    asm mov dh,x
    asm mov dl,y
    asm mov cx,1
    asm int 10h
  }
```

```
  if(c==13)pos=(x+1)*80;
  else pos=x*80+y+1;
  x=pos/80;y=pos%80;
  move(x,y);
  asm pop dx
  asm pop cx
  asm pop bx
  asm pop ax
}
//输出字符
//没有退格上下左右

void puts(char *s,char col)
{
  int i;
  for(i=0;s[i]!='\000';i++)putchar(s[i],col);
  putchar(13,col);
}
//输出字符串
```

5.3 显示一首诗或一首歌

当调用系统中断 01h 号功能时，在屏幕上显示一首古诗《登鹳雀楼》。当调用系统中断的 02h 号功能时，在屏幕上显示刘若英的歌曲《后来》的前四句。因为汇编显示中文不太方便，所以以上信息都采用拼音显示。

我采用了类似打字机的方式将诗句和歌词显示在屏幕上。首先我调用 int10h 中断的 13h 号功能，在屏幕相应位置用白色显示相应的诗词。然后我写了一个循环，人工在同一个位置再将这句诗以不同颜色输出，从而达到效果。每个字第一个字母大写，用红色输出，其他字母用黄色输出。第二次输出时每两个字之间有延迟，这可以通过循环空转实现。

以下为相关代码。

按特定格式输出字符串：

```
void print_format(char *s)
{
  int i;char X=x,Y=y;
  puts(s,0x0F);move(X,Y);
  for(i=0;s[i]!='\000';i++)
  {
    if('A'<=s[i]&&s[i]<='Z')putchar(s[i],0x0c);
    else putchar(s[i],0x0e);
    sleep(10000);
  }
}
```

01h 功能，输出一首诗：

```
void print_poem()
{
```

```
  clear_screen(0,40,12,79);
  move(2,50);puts("Deng He Que Lou",0x0d);
  move(3,55);puts("Meng Hao Ran",0x0d);
  move(5,50);print_format("Bai Ri Yi Shan Jin,");
  move(6,50);print_format("Huang He Ru Hai Liu.");
  move(7,50);print_format("Yu Qiong Qian Li Mu,");
  move(8,50);print_format("Geng Shang Yi Ceng Lou.");
}
```

02h 功能，输出歌词:

```
void print_song()
{
  clear_screen(0,40,12,79);
  move(2,50);puts("Hou Lai",0x0d);
  move(3,60);puts("Liu Ruo Ying",0x0d);
  move(5,40);print_format("Hou Lai");
  move(6,50);print_format("Wo Zong Suan Xue Hui Le");
  move(7,40);print_format("Ru He Qu Ai");
  clear_screen(5,40,7,79);
  move(5,40);print_format("Ke Xi Ni Zao Yi Jing");
  move(6,50);print_format("Xiao Shi Zai Ren Hai");
  clear_screen(5,40,7,79);
  move(5,40);print_format("Hou Lai");
  move(6,50);print_format("Zhong Yu Zai Yan Lei Zhong");
  move(7,40);print_format("Ming Bai");
  clear_screen(5,40,7,79);
  move(5,40);print_format("You Xie Ren");
  move(6,50);print_format("Yi Dan Cuo Guo Jiu Bu Zai.");
}
```

5.4 屏幕清屏，上滚，下滚

屏幕清屏、上滚、下滚这三个功能在实验二、三的时候就已经实现，调用 int10h 中断的 06h 和 07h 功能即可。这里将其装入系统中断中。参数调用见下表。

屏幕清屏				
ah	ch	cl	dh	dl
03h	左边界	上边界	右边界	下边界
屏幕上滚				
ah	ch	cl	dh	dl
03h	左边界	上边界	右边界	下边界
屏幕下滚				
ah	ch	cl	dh	dl
03h	左边界	上边界	右边界	下边界

相关代码如下。

```
//(a,b)为左上角坐标，(c,d)为右下角坐标
void clear_screen(char a,char b,char c,char d)
```

```
{
  asm mov ah,06h
  asm mov al,0
  asm mov ch,a
  asm mov cl,b
  asm mov dh,c
  asm mov dl,d
  asm mov bh,07h
  asm int 10h
}
//屏幕清屏

void rool_up(char a,char b,char c,char d)
{
  asm mov ah,06h
  asm mov al,1
  asm mov bh,07h
  asm mov ch,a
  asm mov cl,b
  asm mov dh,c
  asm mov dl,d
  asm int 10h
}
//屏幕上滚

void rool_down(char a,char b,char c,char d)
{
  asm mov ah,07h
  asm mov al,1
  asm mov bh,07h
  asm mov ch,a
  asm mov cl,b
  asm mov dh,c
  asm mov dl,d
  asm int 10h
}
//屏幕下滚
```

5.5 退出操作系统

我将实验四中通过 15h 号中断实现的退出操作系统的代码移植到了系统中断中，通过 4ch 功能调用。相关代码如下。

```
void sys_exit()
{
  asm mov ax,5301h
```

```
  asm xor bx,bx
  asm int 15h
  asm mov ax,530Eh
  asm xor bx,bx
  asm mov cx,102h
  asm int 15h
  asm mov ax,5307h
  asm mov bx,1
  asm mov cx,3
  asm int 15h
}
```

5.6 系统中断入口与测试方法

我在操作系统中加入了“system”指令，当输入该指令后进入系统中断调用函数system_call。该区域内有如下几个功能。

ouch	输出“OUCH!”
poem	输出古诗
music	输出歌词
cls	清右上四分之一屏
exit	退出系统调用

当进入系统调用部分后，输入提示由“>>”变成“system>>”，这需要对输入部分的退格键，左右键功能进行一些修改。我加入全局变量 POS 表示当前输入提示的长度，左键和退格键不能移动到该位置之前。同时进入 system 模式后右上角四分之一去会被清屏。

对于屏幕上滚，下滚，清屏和退出操作系统的功能，我将其系统中断的入口写在了“system.h”库里，通过调用相关函数，调用系统中断。

以下为相关代码

System_call 函数:

```
void system_call()
{
  char str[BUFLEN],op,X,Y;
  clear_screen(0,40,15,79);have_run=1;
  puts("  'ouch'   --- show ouch");
  puts("  'poem'   --- read a poem");
  puts("  'music'  --- listen to music");
  puts("  'cls'    --- clear the screen");
  puts("  'exit'   --- exit system work");
  while(1)
  {
    printf("system>>");POS=8;
    gets(str);
    if(strcmp(str,"exit"))break;
    if(strcmp(str,"ouch"))op=0;
    else if(strcmp(str,"poem"))op=1;
    else if(strcmp(str,"music"))op=2;
```

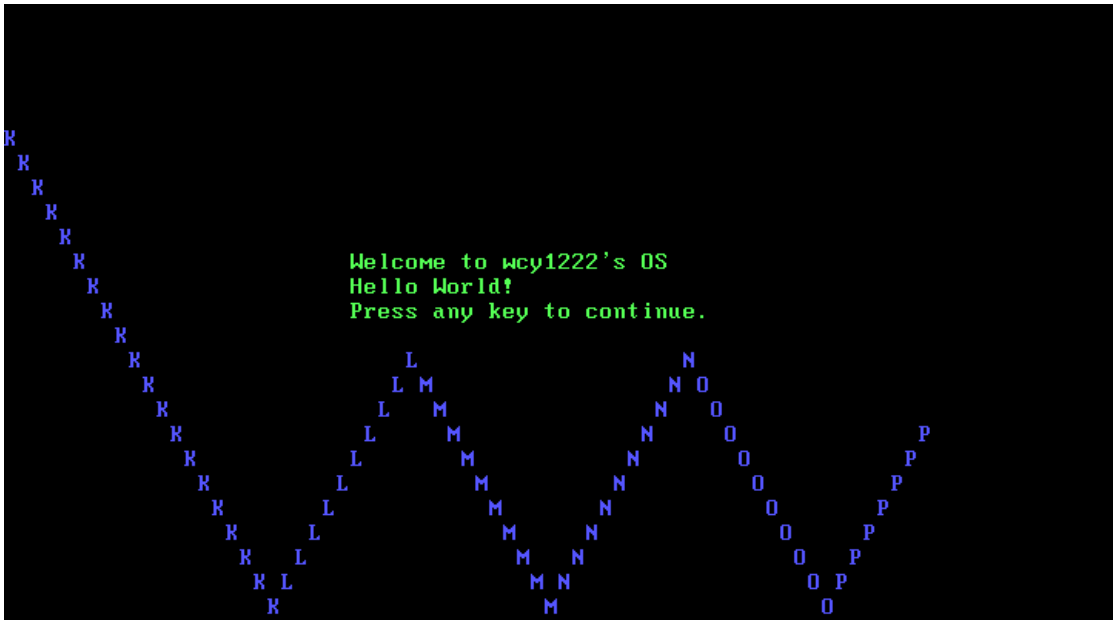
```
    else if(strcmp(str,"cls")){
      clear_screen(0,40,15,79);continue;
    }
    else{puts("Invalid command!");continue;}
    X=x;Y=y;
    asm mov ah,op
    asm int 21h
    move(X,Y);
  }
}
```

“system.h”

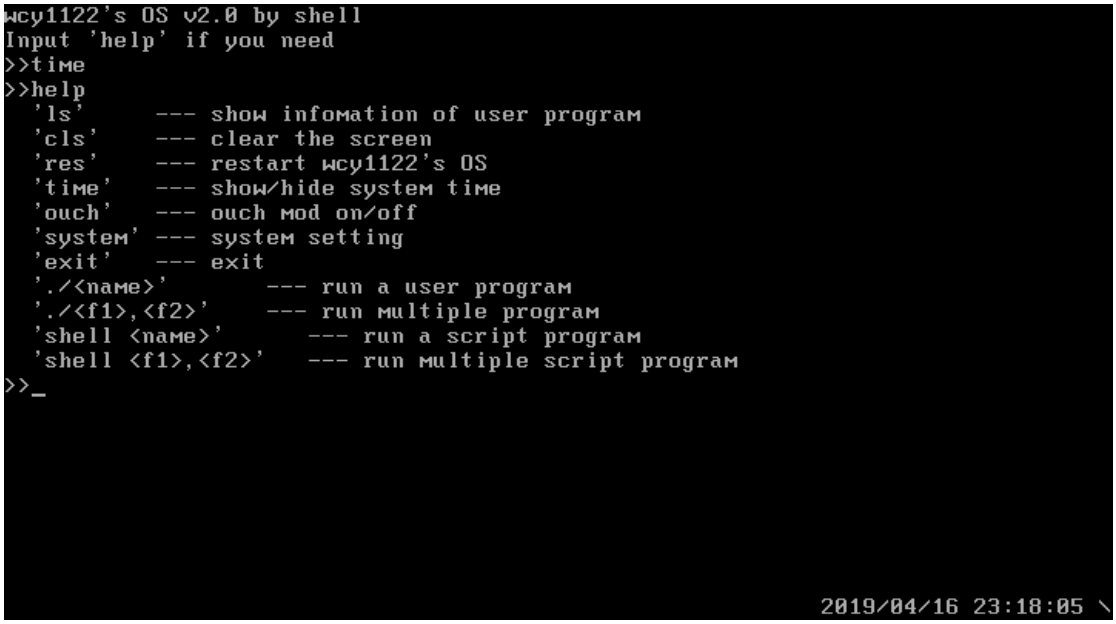
```
#ifndef __SYSTEM_H
#define __SYSTEM_H

void clear_screen(char a,char b,char c,char d)
{
  asm mov ah,03h
  asm mov ch,a
  asm mov cl,b
  asm mov dh,c
  asm mov dl,d
  asm int 21h
}

void rool_up(char a,char b,char c,char d)
{
  asm mov ah,04h
  asm mov ch,a
  asm mov cl,b
  asm mov dh,c
  asm mov dl,d
  asm int 21h
}

void rool_down(char a,char b,char c,char d)
{
  asm mov ah,05h
  asm mov ch,a
  asm mov cl,b
  asm mov dh,c
  asm mov dl,d
  asm int 21h
}
```

```
void sys_exit()
{
  asm mov ah,0x4c;
  asm int 21h;
}

#endif
```

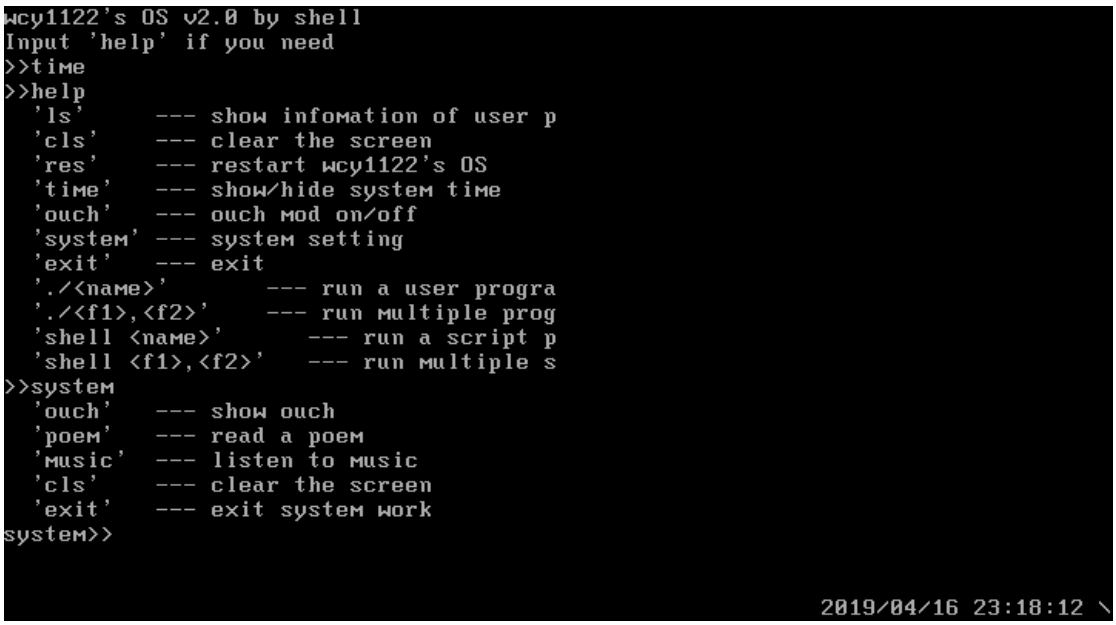
5.7 运行结果



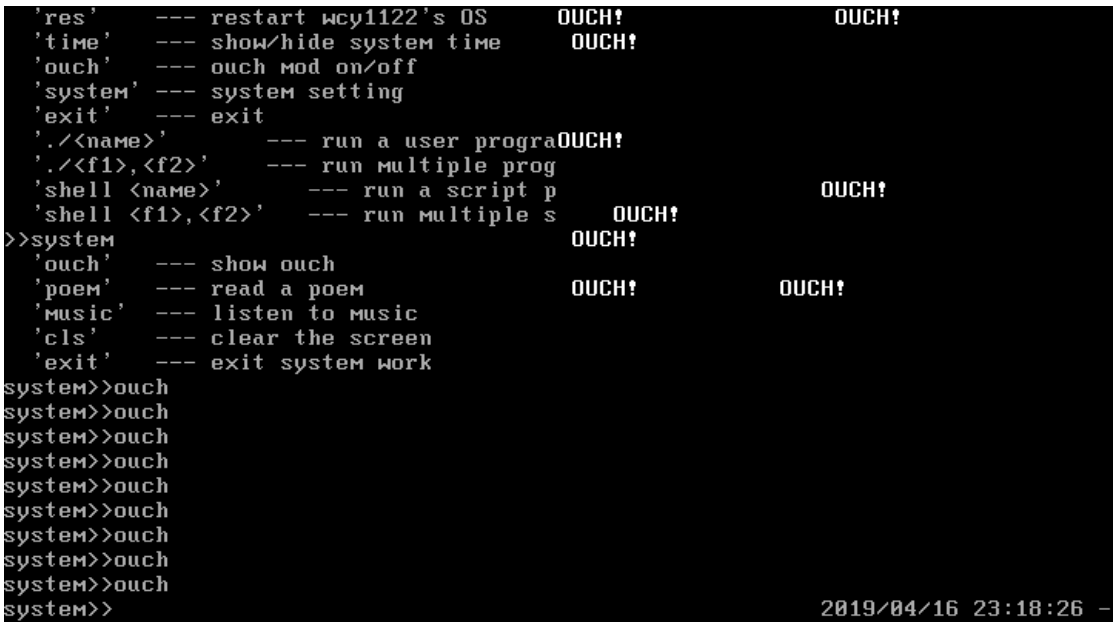
进入操作系统



显示风火轮和系统时间，后显示 help，命令提示



进入系统调用



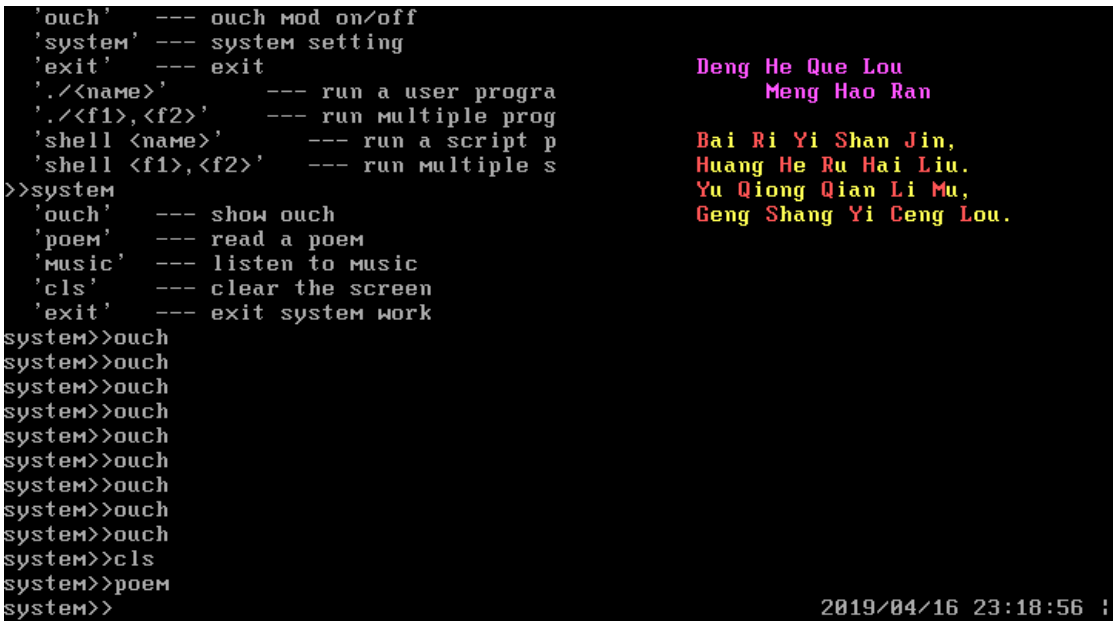
连续多次调用 ouch

```
  'time'   --- show/hide system time
  'ouch'   --- ouch mod on/off
  'system' --- system setting
  'exit'   --- exit
  './<name>'       --- run a user progra
  './<f1>,<f2>'    --- run multiple prog
  'shell <name>'      --- run a script p
  'shell <f1>,<f2>'   --- run multiple s
>>system
  'ouch'   --- show ouch
  'poem'   --- read a poem
  'music'  --- listen to music
  'cls'    --- clear the screen
  'exit'   --- exit system work
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>cls
system>>_                                          2019/04/16 23:18:32 -
```

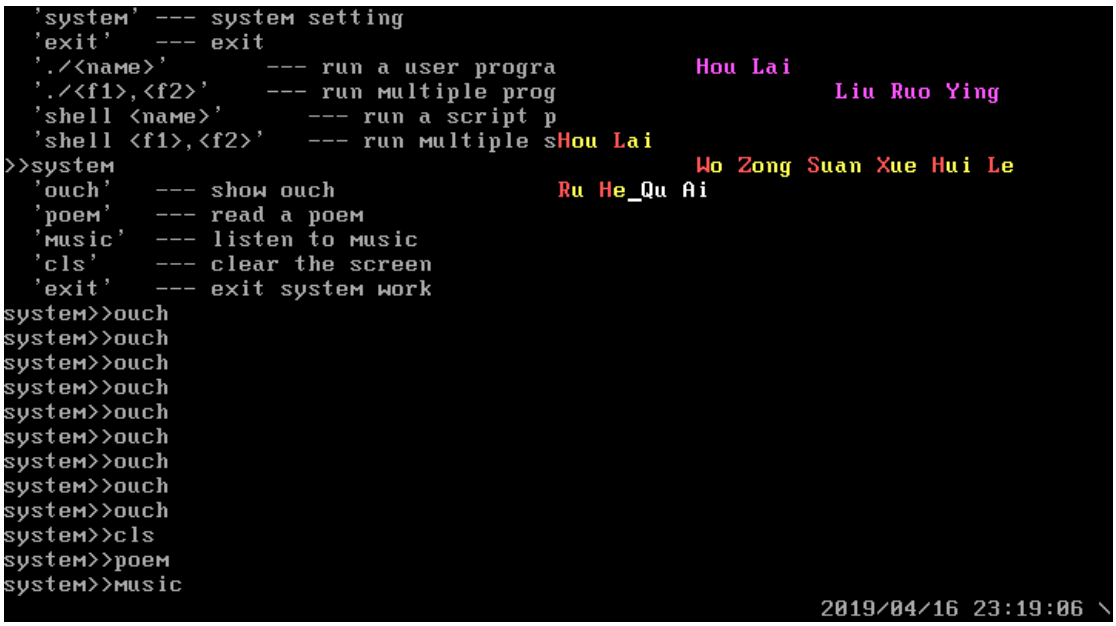
清屏

```
  'ouch'   --- ouch mod on/off
  'system' --- system setting
  'exit'   --- exit                                Deng He Que Lou
  './<name>'       --- run a user progra                Meng Hao Ran
  './<f1>,<f2>'    --- run multiple prog
  'shell <name>'      --- run a script p           Bai Ri Yi Shan Jin,
  'shell <f1>,<f2>'   --- run multiple s           Huang He Ru Hai Liu.
>>system
  'ouch'   --- show ouch
  'poem'   --- read a poem
  'music'  --- listen to music
  'cls'    --- clear the screen
  'exit'   --- exit system work
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>cls
system>>poem
                                                   2019/04/16 23:18:45 /
```

《登鹳雀楼》显示中


'ouch' --- ouch mod on/off
'system' --- system setting
'exit' --- exit
'./<name>' --- run a user progra
'./<f1>,<f2>' --- run multiple prog
'shell <name>' --- run a script p
'shell <f1>,<f2>' --- run multiple s
>>system
'ouch' --- show ouch
'poem' --- read a poem
'music' --- listen to music
'cls' --- clear the screen
'exit' --- exit system work
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>cls
system>>poem
system>>
Deng He Que Lou
Meng Hao Ran
Bai Ri Yi Shan Jin,
Huang He Ru Hai Liu.
Yu Qiong Qian Li Mu,
Geng Shang Yi Ceng Lou.
2019/04/16 23:18:56

显示完毕


'system' --- system setting
'exit' --- exit
'./<name>' --- run a user progra
'./<f1>,<f2>' --- run multiple prog
'shell <name>' --- run a script p
'shell <f1>,<f2>' --- run multiple sHou Lai
>>system
'ouch' --- show ouch
'poem' --- read a poem
'music' --- listen to music
'cls' --- clear the screen
'exit' --- exit system work
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>cls
system>>poem
system>>music
Hou Lai
Liu Ruo Ying
Wo Zong Suan Xue Hui Le
Ru He_Qu Ai
2019/04/16 23:19:06

《后来》显示中

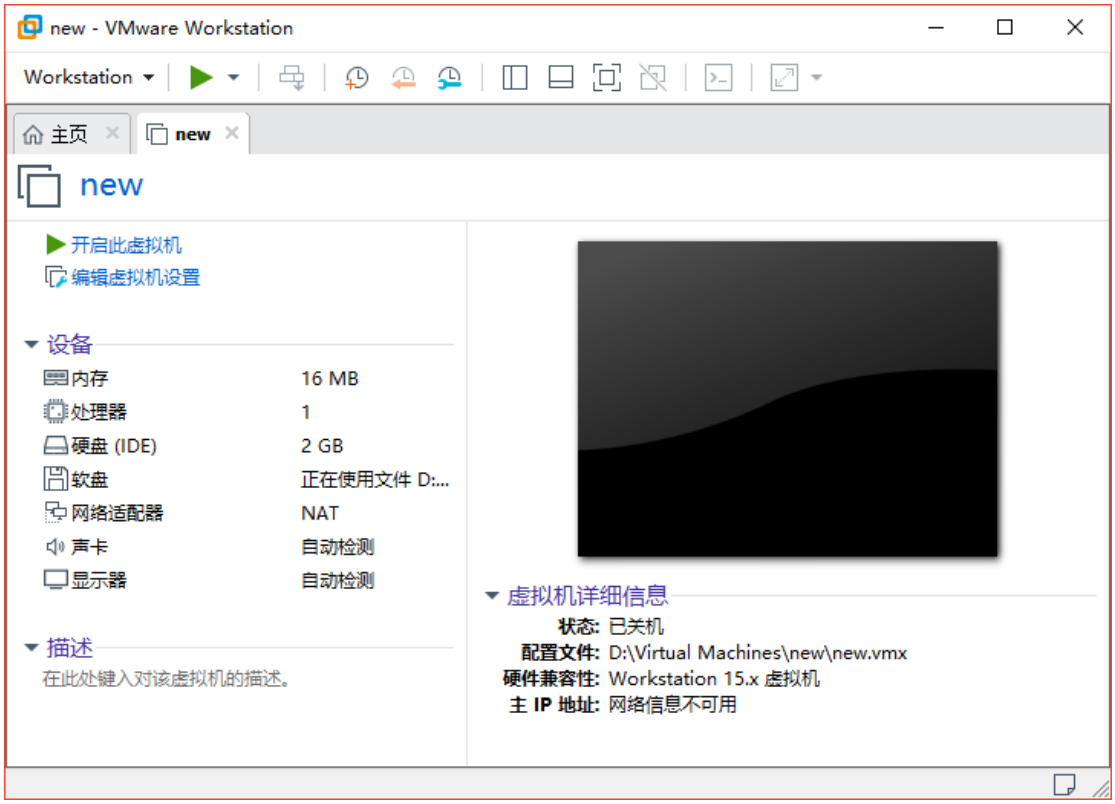
```
  'system' --- system setting
  'exit'   --- exit
  './<name>'       --- run a user progra          Hou Lai
  './<f1>,<f2>'    --- run multiple prog                    Liu Ruo Ying
  'shell <name>'      --- run a script p
  'shell <f1>,<f2>'   --- run multiple sYou Xie Ren
>>system                                          Yi Dan Cuo Guo Jiu Bu Zai.
  'ouch'   --- show ouch
  'poem'   --- read a poem
  'music'  --- listen to music
  'cls'    --- clear the screen
  'exit'   --- exit system work
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>cls
system>>poem
system>>music
system>>_                                         2019/04/16 23:19:24 \
```

显示完毕

```
  'exit'   --- exit
  './<name>'       --- run a user progra
  './<f1>,<f2>'    --- run multiple prog          Hou Lai
  'shell <name>'      --- run a script p                   Liu Ruo Ying
  'shell <f1>,<f2>'   --- run multiple s
>>system                                 You Xie Ren
  'ouch'   --- show ouch                          Yi Dan Cuo Guo Jiu Bu Zai.
  'poem'   --- read a poem
  'music'  --- listen to music
  'cls'    --- clear the screen
  'exit'   --- exit system work
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>ouch
system>>cls
system>>poem
system>>music
system>>exit
>>                                                2019/04/16 23:19:40 -
```

退出系统调用

清屏后，准备按键退出操作系统



成功退出操作系统

6 实验总结

中断向量数量有限，引入系统调用可以拓展中断的数量和功能。本次实验需要自己实现系统调用并进行测试。相较于之前的几次实验，本次实验总体来说还是相对比较简单和顺利的，并没有像前几个实验一样遇到很多问题。本次实验大概是实验四，即中断实验的拓展，也是在为下个实验，多进程实验做准备吧。

本次实验的工作量也不是很大，大概就是把之前实现好的一些代码移植到系统中断，并进行一些 C 语言的编程。通过 C+汇编联合编程，可以减少很多的代码量。汇编百来行的代码在 C 语言中只要是十几行就能实现了。

对于本次实验，我还是比较满意的。在系统调用中断中，我实现了显示字符串，显示古诗，显示歌词，屏幕上滚，屏幕下滚，屏幕清屏，退出操作系统等功能，写出了系统终端的调用入口，完成了老师布置的任务并进行了一些简单的拓展。

通过本次实验，我了解了系统中断的原理和调用方式，更熟悉了中断程序的加载，调用的方式。我更熟悉了 C+汇编联合编程的方法，也体会到了联合编程的优势。

参考文献

[1] BIOS 中断大全 https://www.cnblogs.com/coderCaoyu/p/3638713.html