# 操作系统实验报告

计科（大数据、人工智能方向）17341155 王永康

## （实验五：具有系统调用的内核）

## 一、实验目的

给内核增加一个过程作为系统调用的总入口，使内核可以提供一定的系统调用功能。

## 二、实验内容

- 把int 21h中断号的功能修改为由功能号决定；
- 提供基本的输入、输出函数。

## 三、实验方案

### 操作系统实验工具与环境

- 实验支撑环境
    - 硬件：个人计算机
    - 主机操作系统：Windows10
    - 虚拟机软件：VMware,Dosbox
- 实验开发工具：
    - 汇编语言工具：x86汇编语言、C语言
    - 汇编编译工具：TASM+TCC
    - 磁盘写入工具：Winhex

## 四、实验过程

### 编写21h中断号，根据ah功能不同，调用不同过程

利用中断号33号，修改中断向量表，33号中断向量对应33*4和33*4+2两个地址，把33号中断需要执行的部分的偏移入口地址放置在33*4中，段地址放置在33*4+2中，具体代码如下：

```
```

```
NewINT33 proc
    push ds
    push es
        push bx
        push cx
        push bp
        push es

            cmp ah,4 ;;若大于4，则直接跳出
            ja sret
            add ah,3 ;;扇区号 = 原来的+3
            mov byte ptr cs:[Sectors],ah
            call LOAD_for_INT ;;加载扇区程序，并运行

            sret:

        pop es
        pop bp
        pop cx
        pop bx
    pop es
    pop ds
    iret
    ret
NewINT33 endp
```

### 整理内核，编写`Stdio.h`库

这一次实验中完成了下列函数的编写：(`Stdio.h`库)

```
//Stdio.h
void putChar(char a,int h,int l);
void printf(const char*s);
void getC(char* dt);
void getline(char st[]);
void Clears();
int len(const char*s);
int cmp_equ(char*s,char* t);
```

与之相关的是`klib.asm`

```
public _printChar
public _getChar
public _clear
public _Set_Cursor
```

其中，`Stdio.h`中的函数设计思想是：用C封装C。

以printf()函数为例：

首先，在汇编中实现

```
int _COL = 0;
int _ROW = 0;
char _MESSEAGE;
extern void printChar();
```

这样一个C函数，通过上述的全局变量传递参数。

之后封装成：

```
void putChar(char a,int h,int l)
{
    _MESSEAGE = a;
    _COL = l;
    _ROW = h;
    printChar();
    _DISP_POS_ = _ROW;
}
```

之后，再利用上述`putChar()`封装成`printf()`

```
void printf(const char*s)
{
    int i = 0;
    for(i=0;i<len(s);++i){

        if(s[i] == '\n'){
            _DISP_POS_++;
            putChar('\0',_DISP_POS_,0);
        }
        else{
            putChar(s[i],_DISP_POS_,i);
        }
    }

}
```

这就是这些函数的设计思想，其余函数类似，代码如下：

`void getline(char st[])`

```

```

```
void getline(char st[])
{

    int i = 0;
    int cntt = 0;
    int init_col = _COL;

    while(1){
        _FLAG  = 0;
        getC(ttt+i);

        if(_FLAG  == 1){
            if(_COL>init_col){ //考虑输入之后删除的退格操作
                putChar(' ',_ROW,_COL);
                putChar(' ',_ROW,_COL+1);
                putChar('\b',_ROW,_COL);
                _COL--;
            }


            if(_COL>init_col){
                _COL--;
                i--;
            }
            continue;
        }
        i++;
        _COL++;
        if(ttt[i-1] == '\n'){
            break;
        }
        putChar(ttt[i-1],_ROW,_COL); //回显输入的字符
    }

    putChar('\0',++_DISP_POS_,0);
    for(cntt = 0;cntt<i;++cntt){
        st[cntt] = ttt[cntt];   //最终把结果存入目标变量
    }
    st[i-1] = '\0';
}

```

```
int len(const char*s) //长度函数
{
    int cnt = 0;
    while(s[cnt++]!='\0');
```

```
    cnt--;
    return cnt;
}

int cmp_equ(char*s,char* t)//比较函数
{
    int i = 0;
    if(len(s)!=len(t)) return 0;

    for(i=0;i<len(s);++i){
        if(s[i] != t[i]) return 0;
    }

    return 1;
}

void Clears() //清屏函数
{
    clear();
    _DISP_POS_ = 0; //控制在屏幕上输出行的一个全局变量
```

最后，写一个测试程序（C与汇编联合的程序）：

`test.asm`

```
extern  macro %1    ;统一用extern导入外部标识符
    extrn %1
endm

extern _main:near

.8086
_TEXT segment byte public 'CODE'
DGROUP group _TEXT,_DATA,_BSS
       assume cs:_TEXT
   org 0C100h
start:
    mov  ax,  cs
    mov  ds,  ax            ; DS = CS
    mov  es,  ax            ; ES = CS
    mov  ss,  ax
    mov  ah,1
    int 21h


    mov ah,2
    int 21h
```

```
23
24      mov ah,3
25      int 21h
26
27      mov ah,4
28      int 21h
29
30
31      call near ptr _main
32
33      ret
34
35      include klib.asm
36
37  _TEXT ends
38  ;************DATA segment*************
39  _DATA segment word public 'DATA'
40
41  _DATA ends
42  ;*************BSS segment*************
43  _BSS    segment word public 'BSS'
44  _BSS ends
45  ;**************end of file***********
46  end start
```

main.c

```
1  #include "Stdio.h"
2  char et[40];
3  void main()
4  {
5      Clears();
6      printf("this is a test!\n");
7
8      getline(et);
9
10     printf("return 0 ..........press any key to
   continue....\n");
11     getline(et);
12
13 }
```

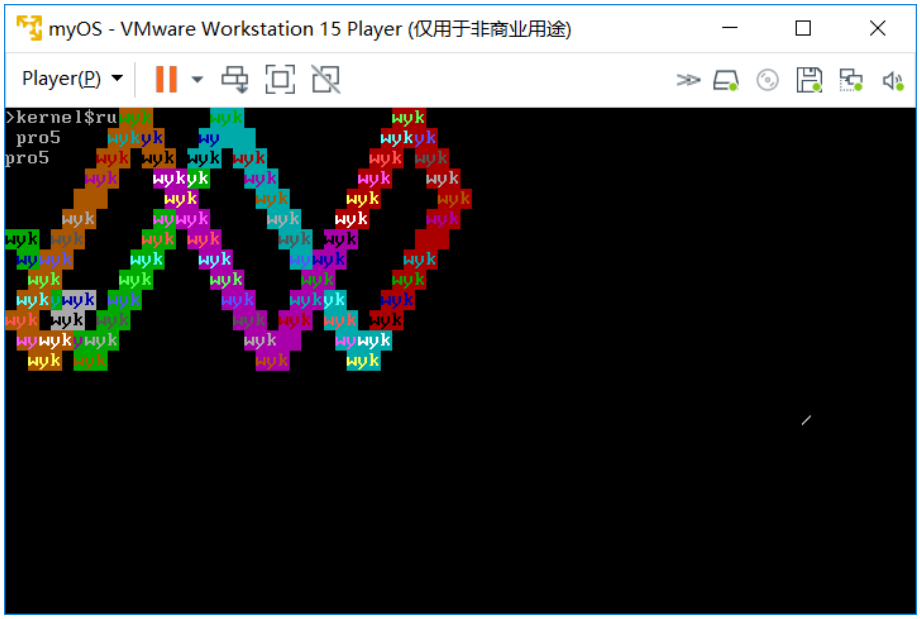
其中上述程序放在第8、9扇区，加载至内存地址0xc100h处。

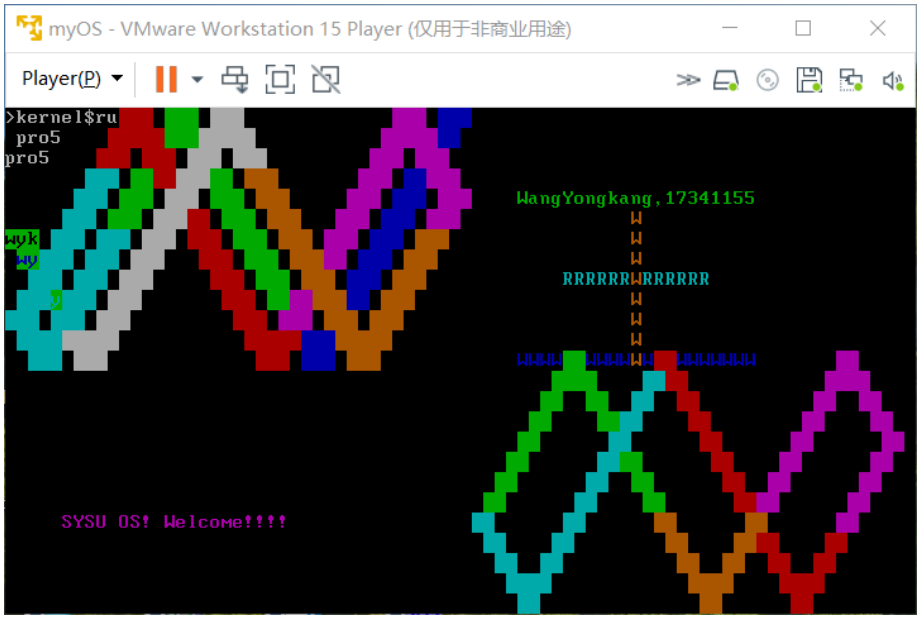
## 四、实验结果

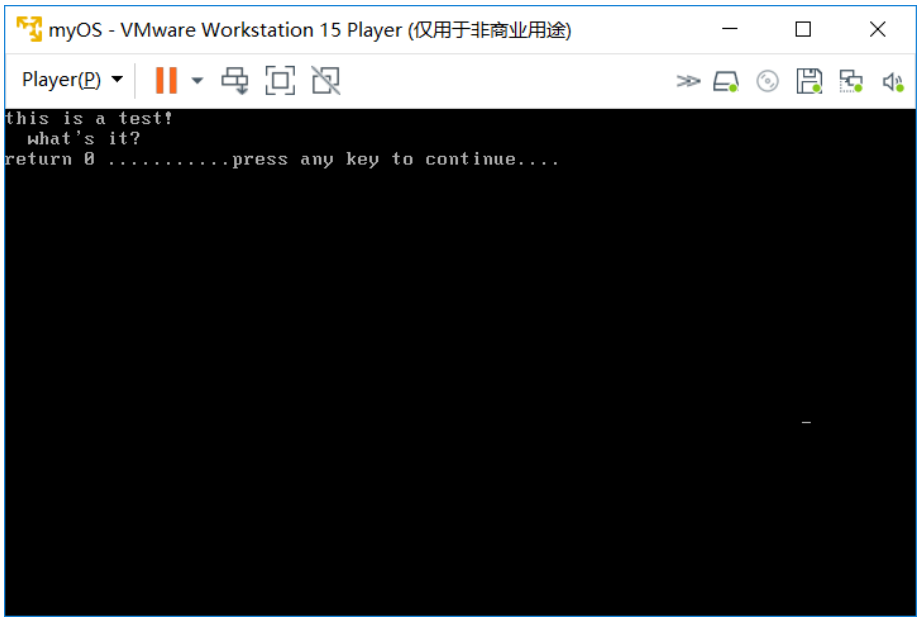
输入`run`

`pro5`

之后运行如图：




## 五、实验总结

这一次系统调用的设计实验，相对较为简单，体会也主要有以下两点：

- **实验循序渐进的过程使得前面实验的基础很重要**

  这一次实验，可以说是实验四的一个拓展，中断稍加修改就可以完成实验。另外就是汇编与C混合编译的基础也是相当重要，这样使得实验中可以少写很多汇编，这样可以使实验过程相对而言较为舒服。由于我自己之前没有把`kernel.asm`代码与一些函数功能分开，这一次实验就这么做了，拆分了它，新增了两个库`system.asm`和`klib.am`，这也使我对C与汇编混合编译的过程更加熟悉。

- **写程序慢慢来，一步一步debug效果更好**

这一次实验，自己在编写的`Stdio.h`后，测试程序去测试，开始一直得不到理想结果，然后debug过程中，也是费了较大劲，因为一下子写了太多，实在是不好定位错误，最后也是反复查找才准确找到，但是，如果一来能一个模块、一个一个函数来测试，这样定位bug肯定会快很多。

## 参考文献

[1] 凌应标. “05实验课.ppt”，中山大学计算机科学系，2015-3