实验五报告

【个人信息】

院系：数据科学与计算机学院　　专业、年级：17 级计算机科学与技术
学号：17341097　　姓名：廖永滨　　指导教师：凌应标

【实验题目】

1. 解释系统调用的意义及规划一组系统调用功能

2. 示范一个简单的系统调用实现

3. 为用户程序开发提供一些常用的服务，并写进 21 号中断中，扩展功能号作用。

【实验目的】

1. 加深对中断机制知识的理解，掌握中断处理程序设计的要求。

2. 扩展丰富操作系统。

【实验要求】

1. 实验五在实验四基础上进行，保留或扩展原有功能，实现部分新增功能。

2. 新增功能号的划分。

【实验方案】

1. 实验工具：

Notepad++：编写程序时使用的编辑器；
Subline：可以以 16 进制的方式打开并编辑任意文件；
TAMS 汇编工具：可以将汇编代码编译成对应的二进制代码；
NAMS 汇编工具：可以将汇编代码编译成对应的二进制代码；
TCC 编译器：可以将 c 代码编译成对应的二进制代码；
TLINK 链接器：将多个.obj 文件链接成.com 文件
VmWare 虚拟机：创建裸机环境，生成虚拟磁盘

2. 实验基本操作方法：

a. 使用 TCC 编译命令：tcc -mt -c - o cfile.obj cfile.c >ccmsg.txt
b. TASM 汇编命令：tasm afile.asm afile.obj > amsg.txt
c. 链接命令：tlink /3 /t cfile.obj afile.obj , final.com
d. NASM 汇编指令：NASM afile.asm

3. 实验原理

中断机制基础知识，中断程序的编写规范。

【实验过程】

1、实现的内容：

该实验中有 6 个程序，分别为 loading，MyOS，a1，a2, a3, a4。

lording 作为系统引导程序，放在软盘第一个扇区。

MyOS 为操作系统，由 TASM 汇编语言与 TURBO C 语言汇合编写，放在软盘第二个到第二十个扇区（占据 19 个扇区）。

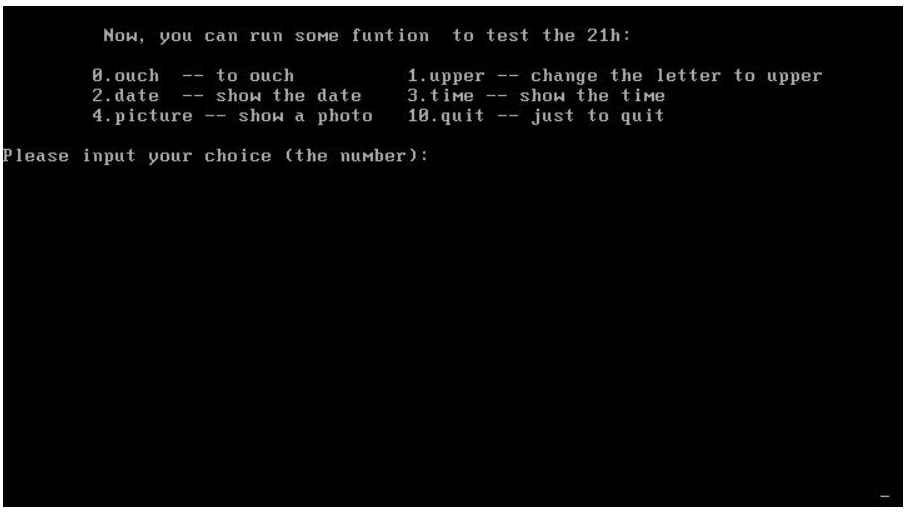
a1，a2, a3, a4 为用户程序，延用实验三的四个程序。分别放在软盘第二十一、第二十二、第二十三和第二十四个扇区，主要用于演示键盘中断。

运行时首先启动引导程序 lording,把系统 MyOS 装载到内存 8100h 中，并把计算机的控制权交给 MyOS 系统，系统立即将时钟中断向量的偏移地址设置为时钟中断程序的地址，系统开始定时执行时钟中断程序，然后设置 33h，34h,35h,36h 中断向量的偏移地址，设置 21h 中断向量偏移地址，作为系统服务准备接受用户调用，接着等待用户输入指令。

实验四已有的功能和程序不再累述。这里展示新出的 21h 中断服务对应的几个功能号。

（！！此外，我还实现了 TAB 指令补全功能，比如输入 h 再按 tab，系统会补全一个 help 指令出来，由于无法展示效果，故只放在代码解析处！！）

服务中断 21h 的对应功能号的概述：



解释：0 号功能：在屏幕上输出 ouch

1 号功能：将大写字符串变成小写

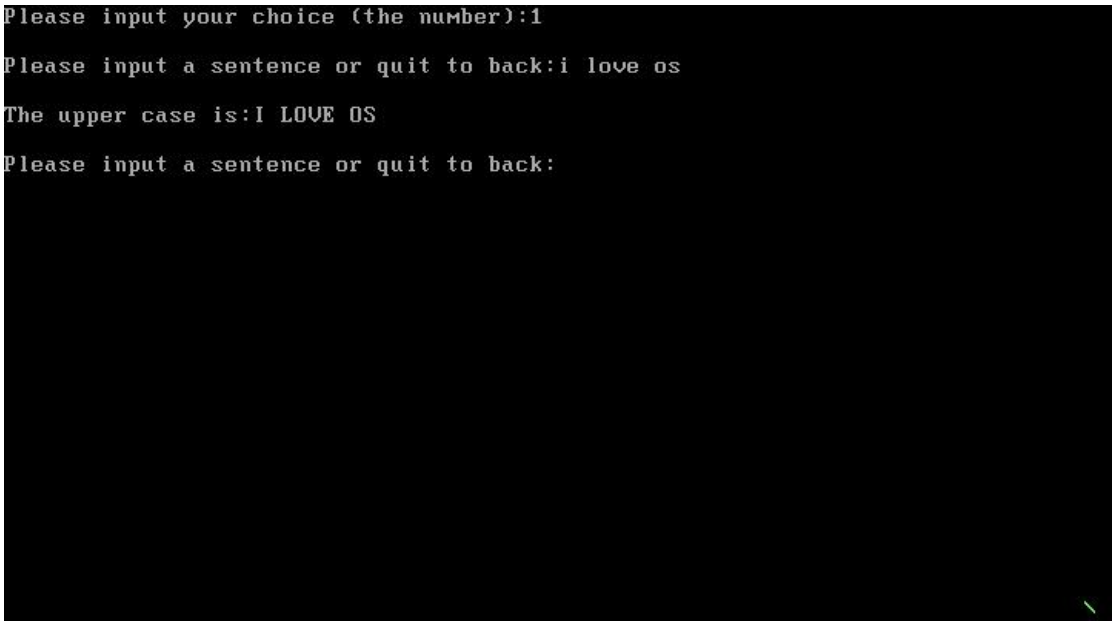
2 号功能：输出当前的系统日期

3 号功能：输出当前的系统时间

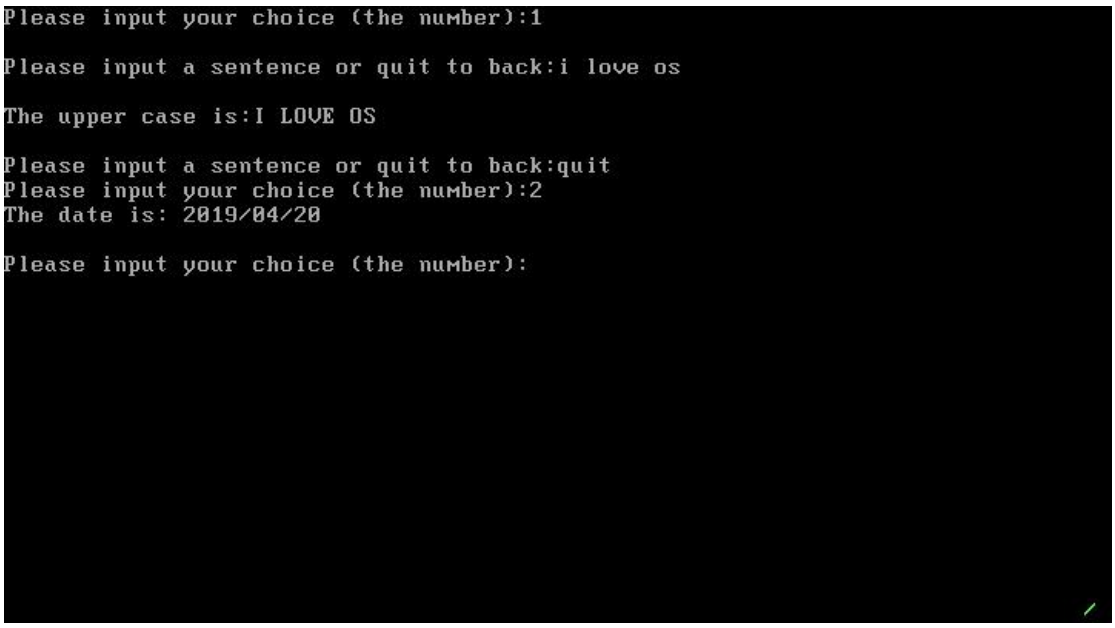
4 号功能：显示一个字符图案

当输入 10 时代表退出 21h 中断号的测试

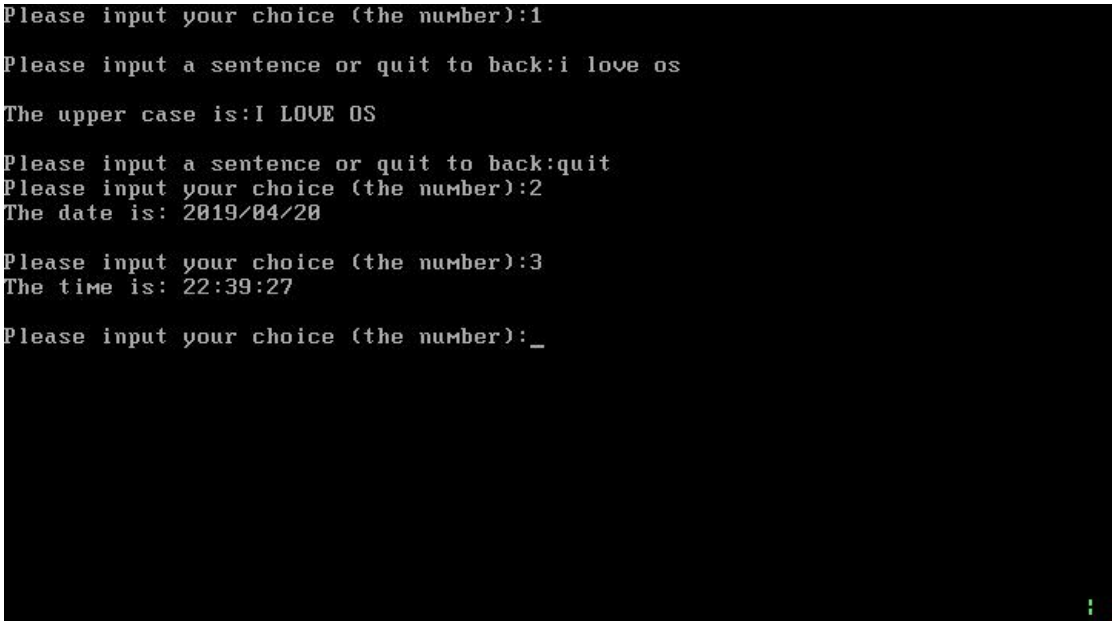
21h，0 号功能



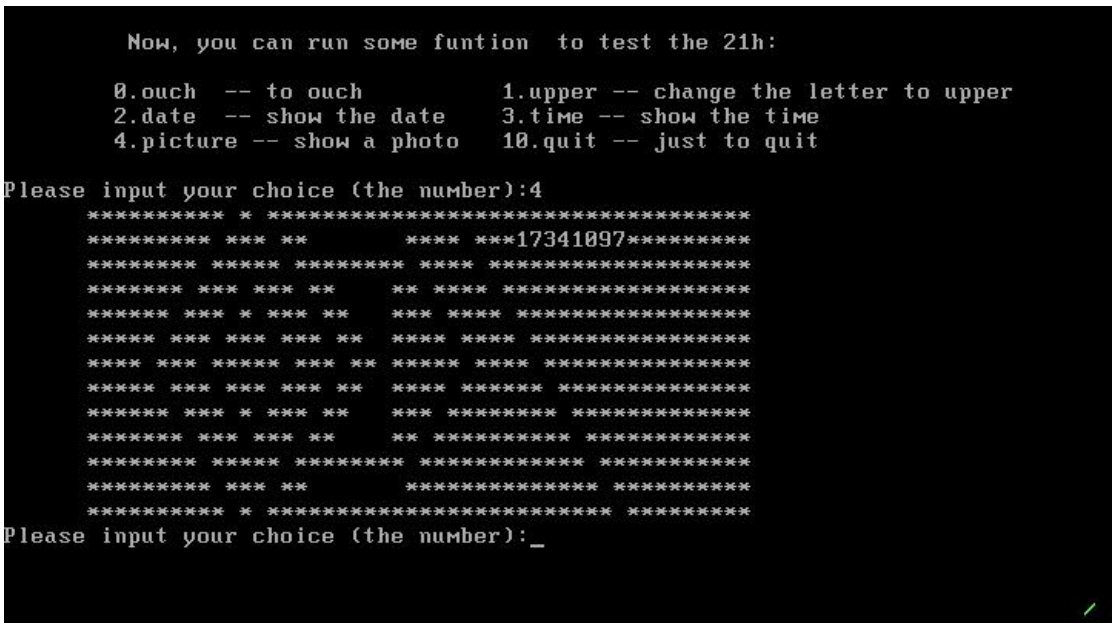
21h，1 号功能



21h，2 号功能



21h，3 号功能



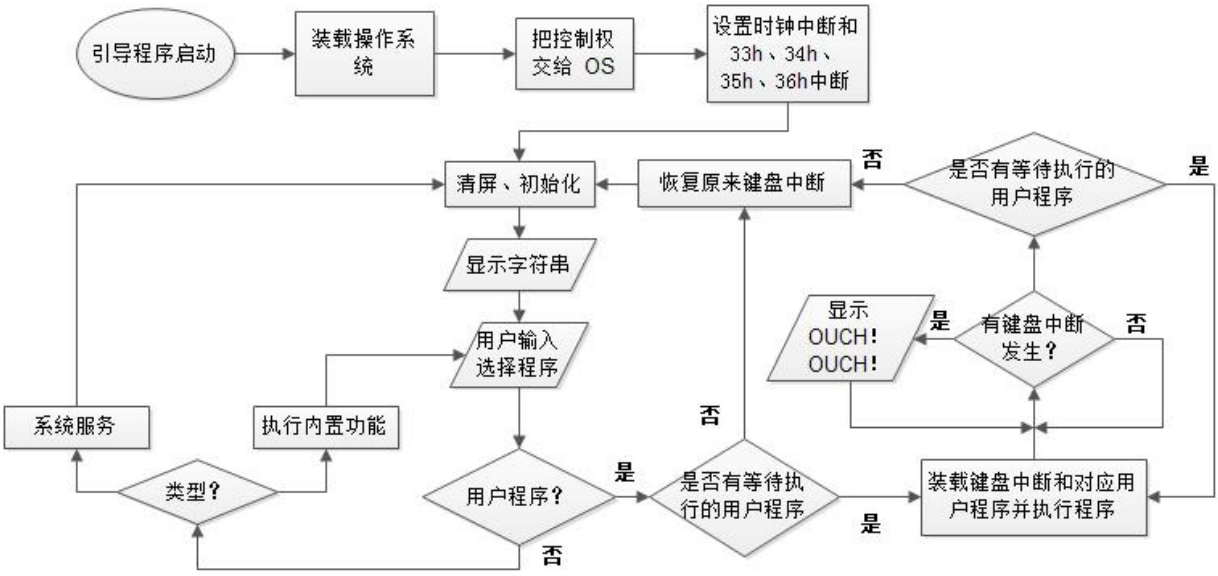
21h，4 号功能

2、中断设置的详细说明：

时钟中断，键盘中断，33h 中断，34h 中断，35h 中断，36h 中断在实验四中已经说明，此处不再说明。

21h 服务中断（新增）：入口地址为 ip:33*4 cs:33*4+2 ，该中断在系统启动后立刻装载到对应中断向量，可以通过 int 21h 进行调用，调用时会根据传入的 ah 功能号进行相应的回馈（0 输出 ouch，1 小写变大写，2 输出日期，3 输出时间，4 输出图案）

3、程序流程图（同实验四，系统服务由 33h，34h，35h，36h 扩展了一个 21h）



内置服务程序的测试流程如下：（相比实验四的中断调用会复杂一点，因为对应中断号的功能有的是用 c 实现的）

输入 int21 -> 输入功能号 i -> 调用 clib.asm 库中相应的汇编函数 -> 汇编函数执行语句 mov ah,i 然后 int21 调用 -> 系统取到 int21 地址并执行相应的中断代码 -> 用 call near 找到 kernal.c 中的实现函数，或是直接汇编代码实现对应函数功能

4、代码解析（实现方法）

(1)loading.asm：

由于 21h 服务中断的增加，操作系统内核扩大了很多，不得不扩展扇区，故引导程序读入的扇区数由 9 变成了 19。

```
Start:
    mov ax, cs
    mov ds, ax
    mov es, ax

Lording_OS:
    mov ax,cs                   ;段地址 ; 存放数据的内存基地址
    mov es,ax                   ;设置段地址（不能直接mov es,段地址）
    mov bx, offsetofos          ;偏移地址; 存放数据的内存偏移地址
    mov ah,2                    ; 功能号
    mov al,19                   ;扇区数
    mov dl,0                    ;驱动器号 ; 软盘为0，硬盘和U盘为80H
    mov dh,0                    ;磁头号 ; 起始编号为0
    mov ch,0                    ;柱面号 ; 起始编号为0
    mov cl,2                    ;起始扇区号 ; 起始编号为1
    int 13H                     ;调用读磁盘BIOS的13h功能
    ;内核程序已加载到指定内存区域中
```

(2)MyOS.asm：

在实验四的基础之上，新增了 21h 中断的设置，并伪包含了一个 clib.asm 用于编写各中断服务的调用程序（实验五要求的库文件，但不是实先函数，只是用于提供函数入口）。

设置中断其实很简单，先将 ax 初始化，然后将段寄存器 es 赋值为 0，在偏移对应的中断位置处给予对应中断的偏移位置，再在+2 的地方加入段地址 cs，按 PPT 的来做就 ok 了，而且所有中断都是一样设置的。

```
    ;设置 21h 的中断
    xor ax,ax
    mov es,ax
    mov word ptr es:[33*4],offset SERVER    ; 设置 21h 的偏移地址
    mov ax,cs
    mov word ptr es:[33*4+2],ax

    include clib.asm
```

(3)services.asm：

此处为服务中断的核心，实际上和实验四很像，加入了功能号的区分。具体代码如下：（只展示新增的 21h 中断设置部分）

```
;
SERVER:
    push bx
    push cx
    push dx
    push bp

    cmp ah,0
    jnz cmp1
    call MOS_21h_0
    jmp exit_21h
cmp1:
    cmp ah,1
    jnz cmp2
    call MOS_21h_1
    jmp exit_21h

cmp2:
    cmp ah,2
    jnz cmp3
    call MOS_21h_2
    jmp exit_21h

cmp3:
    cmp ah,3
    jnz cmp4
    call MOS_21h_3
    jmp exit_21h
    
cmp4:
    cmp ah,4
    jnz exit_21h
    call MOS_21h_4
    jmp exit_21h
    
exit_21h:
    pop bp
    pop dx
    pop cx
    pop bx

    iret                    ;从中断返回
```

```
;***************.********************
;*  21 号中断 0 号功能              *
;*****************.******************
; 屏幕中央显示 OUCH
MOS_21h_0:

    call Clear

    mov ah,13h  ;  功能号
    mov al,0    ;  光标放到串尾
    mov bl,71h  ;  白底深蓝
    mov bh,0    ;  第0页
    mov dh,12   ;  第18行
    mov dl,38   ;  第46列
    mov bp,offset MES_OUCH  ;  BP=串地址
    mov cx,5    ;  串长为 28
    int 10h     ;  调用10H号中断

    ret

MES_OUCH:
    db "OUCH!"

;***************.********************
;*  21 号中断 1 号功能                *
;*****************.******************
; 字符串转为大写
MOS_21h_1:

    push dx


    mov ax,dx
    push ax                 ;  字符串首地址压栈
    call near ptr _to_upper ;  调用 C 过程
    pop cx

    pop dx

    ret

MOS_21h_2:
    call near ptr _to_date
    ret

MOS_21h_3:
    call near ptr _to_time
    ret

MOS_21h_4:
    call near ptr _to_picture
    ret
```

可以看到，用汇编语言实现的 ouch 就直接写在这里，用 c 语言实现的一些功能则通过 call 调用与之相连的 kernal.c 里的函数！

(4)kliba.asm ：

相较实验四，由于系统内核的扩大导致子程序位置改变，所以 run 函数进行了相应的修改，读取扇区位置的磁头号由 0 改为 1。

```
public _run
_run proc
    push es
    xor ax,ax
    mov es,ax
    push word ptr es:[9*4]                ; 保存 9h 中断
    pop word ptr ds:[0]
    push word ptr es:[9*4+2]
    pop word ptr ds:[2]

    mov word ptr es:[24h],offset keyDo    ; 设置键盘中断向量的偏移地址
    mov ax,cs
    mov word ptr es:[26h],ax


    mov ax,cs
    mov es,ax                             ;设置段地址，存放数据的内存基地址
    mov bx,0B100h                         ; ES:BX=读入数据到内存中的存储地址
    mov ah,2                              ; 功能号
    mov al,1                              ; 要读入的扇区数 1
    mov dl,0                              ; 软盘驱动器号（对硬盘和U盘，此处的值应改为80H）
    mov dh,1                              ; 磁头号
    mov ch,0                              ; 柱面号
    mov cl,byte ptr[_pro]                 ; 起始扇区号（编号从1开始）
    int 13H                               ; 调用13H号中断

    mov bx, 0B100h
    call bx                               ; 跳转到该内存地址

    xor ax,ax
    mov es,AX
    push word ptr ds:[0]                  ; 恢复 9h 中断
    pop word ptr es:[9*4]
    push word ptr ds:[2]
    pop word ptr es:[9*4+2]
    int 9h
    pop es
 run endp
```

由于系统功能变得更负责了，需要新增一些汇编函数加以支持，所以我又写了两个汇编函数（一个用于获取时间，一个用于获取日期）。

```
;***************.*******************
;*  void _getdate()                  *
;***************.*******************
; 获取日期
public _getdate
_getdate proc
    push ax
    push bx
    push cx
    push dx
    
    mov ah,4h
    int 1ah

    mov byte ptr[_ch1],ch       ; 将年高位放到 ch1
    mov byte ptr[_ch2],cl       ; 将年低位放到 ch2
    mov byte ptr[_ch3],dh       ; 将月放到 ch3
    mov byte ptr[_ch4],dl       ; 将日放到 ch4

    pop dx
    pop cx
    pop bx
    pop ax
    ret
_getdate endp

;***************.*******************
;*  void _gettime()                  *
;***************.*******************
; 获取时间
public _gettime
_gettime proc
    push ax
    push bx
    push cx
    push dx
    
    mov ah,2h
    int 1ah

    mov byte ptr[_ch1],ch       ; 将时放到 ch1
    mov byte ptr[_ch2],cl       ; 将分放到 ch2
    mov byte ptr[_ch3],dh       ; 将秒放到 ch3

    pop dx
    pop cx
    pop bx
    pop ax
    ret
_gettime endp
```

(5)Kernal.c 程序:

此处改动最大，主要分为两大部分：

A.内置服务程序的实现函数

```
/*服务程序*/
to_upper(char *p)
{
    while(*p != '\0')
    {
        if(*p >= 'a' && *p <= 'z')
        {
            *p = *p - 32;
        }
        p++;
    }
}
void to_time(){
    print("The time is: ");
    gettime();
    hh = BCD_decode(ch1);
    if(hh == 0) print("00");
    else if(hh >0 && hh < 10) printChar('0');
    printInt(hh);
    printChar(':');
    mmm = BCD_decode(ch2);
    if(mmm == 0) print("00");
    else if(mmm > 0 && mmm < 10) printChar('0');
    printInt(mmm);
    printChar(':');
    ss = BCD_decode(ch3);
    if(ss == 0) print("00");
    else if(ss > 0 && ss < 10) printChar('0');
    printInt(ss);
    print("\r\n\r\n");
}
```

```
void to_date(){
    print("The date is: ");
    getdate();
    yy = BCD_decode(ch1)*100 + BCD_decode(ch2);
    if(yy == 0) print("0000");
    else if(yy >0 && yy < 10) print("000");
    else if(yy > 10 && yy < 100) print("00");
    else if(yy > 100 && yy < 1000) print("0");
    printInt(yy);
    printChar('/');
    mm = BCD_decode(ch3);
    if(mm == 0) print("00");
    else if(mm > 0 && mm < 10) printChar('0');
    printInt(mm);
    printChar('/');
    dd = BCD_decode(ch4);
    if(dd == 0) print("00");
    else if(dd > 0 && dd < 10) printChar('0');
    printInt(dd);
    print("\r\n\r\n");
}
void to_picture(){
    print("      ********** * ********************************       \n\r");
    print("      ********* *** **       **** ***17341097*********    \n\r");
    print("      ******** ***** ******** **** ******************     \n\r");
    print("      ******* *** *** **    ** **** ******************     \n\r");
    print("      ****** *** * *** **   *** **** ******************    \n\r");
    print("      ***** *** *** *** **  **** **** ******************   \n\r");
    print("      **** *** ***** *** ** ***** **** ******************  \n\r");
    print("      ***** *** *** *** **  **** ****** ***************    \n\r");
    print("      ****** *** * *** **   *** ******** ************     \n\r");
    print("      ******* *** *** **    ** ********** ************    \n\r");
    print("      ******** ***** ******** ************ ************   \n\r");
    print("      ********* *** **       ************** ***********  \n\r");
    print("      ********** * ************************ *********     \n\r");


}
```

B.内置服务程序的测试函数（与 clib.asm 相结合）

```
12  extern void date();
13  extern void time();
14  extern void to_OUCH();
15  extern void picture();
```

```
260 void int21(){
261     cls();
262     print("\r\n         Now, you can run some funtion  to test the 21h:\n\n\r");
263     print("        0.ouch  -- to ouch          1.upper -- change the letter to upper\n\r");
264     print("        2.date  -- show the date    3.time -- show the time\n\r");
265     print("        4.picture -- show a photo   10.quit -- just to quit\r\n\r\n");
266     while(1){
267         print("Please input your choice (the number):");
268         getline(commands,20);
269         if(strcmp(commands,"0")==0){
270             to_OUCH();
271             cls();
272         }
273         else if(strcmp(commands,"1")==0){
274             while(1){
275                 print("\r\nPlease input a sentence or quit to back:");
276                 getline(commands,30);
277                 if(strcmp(commands,"quit")==0) break;
278                 upper(commands);
279                 print("\r\nThe upper case is:");
280                 print(commands);
281                 print("\r\n");
282             }
283         }
284         else if(strcmp(commands,"2")==0)date();
285         else if(strcmp(commands,"3")==0)time();
286         else if(strcmp(commands,"4")==0)picture();
287         else if(strcmp(commands,"10")==0)break;
288     }
289 }
290
```

(6)clib.asm：（库过程，起到封装作用，但实际上只是和实验四一样的调用函数而已）

```
; +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
; ····························clib.asm
; +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

;***************.*******************
;*··void·_to_OUCH()·····················*
;***************.*******************
; 调用 21h 0号功能
public _to_OUCH
_to_OUCH proc
····push ax
····push bx
····push cx
····push dx
····push es

····call Clear

····mov ah,0
····int 21h

····call DelaySome
···
····pop ax
····mov es,ax
····pop dx
····pop cx
····pop bx
····pop ax
····ret
_to_OUCH endp
```

```
;***************.*******************
;*  void _upper()                    *
;***************.*******************
; 调用 21h 1号功能
public _upper
_upper proc
    push bp
    mov bp,sp
    push si
    mov si,word ptr [bp+4]          ; 获得字符串首地址

    push ax
    push bx
    push cx
    push dx
    push es

    mov ah,1
    mov dx,si                       ; 把字符串首地址给 dx
    int 21h
    
    pop ax
    mov es,ax
    pop dx
    pop cx
    pop bx
    pop ax

    pop si
    pop bp
    ret
_upper endp

;***************.*******************
;*  void _to_date()                  *
;***************.*******************
; 调用 21h 2号功能
public _date
_date proc
    push ax
    push bx
    push cx
    push dx
    push es

    mov ah,2
    int 21h
    
    pop ax
    mov es,ax
    pop dx
    pop cx
    pop bx
    pop ax
    ret
_date endp
```

```
;***************.*********************
;*  void _time()                       *
;*****************.********************
; 调用 21h 3号功能
public _time
_time proc
    push ax
    push bx
    push cx
    push dx
    push es

    mov ah,3
    int 21h
    
    pop ax
    mov es,ax
    pop dx
    pop cx
    pop bx
    pop ax
    ret
_time endp

;***************.*********************
;*  void _picture()                    *
;*****************.********************
; 调用 21h 4号功能
public _picture
_picture proc
    push ax
    push bx
    push cx
    push dx
    push es

    mov ah,4
    int 21h
    
    pop ax
    mov es,ax
    pop dx
    pop cx
    pop bx
    pop ax
    ret
_picture endp
```

【新增特色】

代码补全功能：

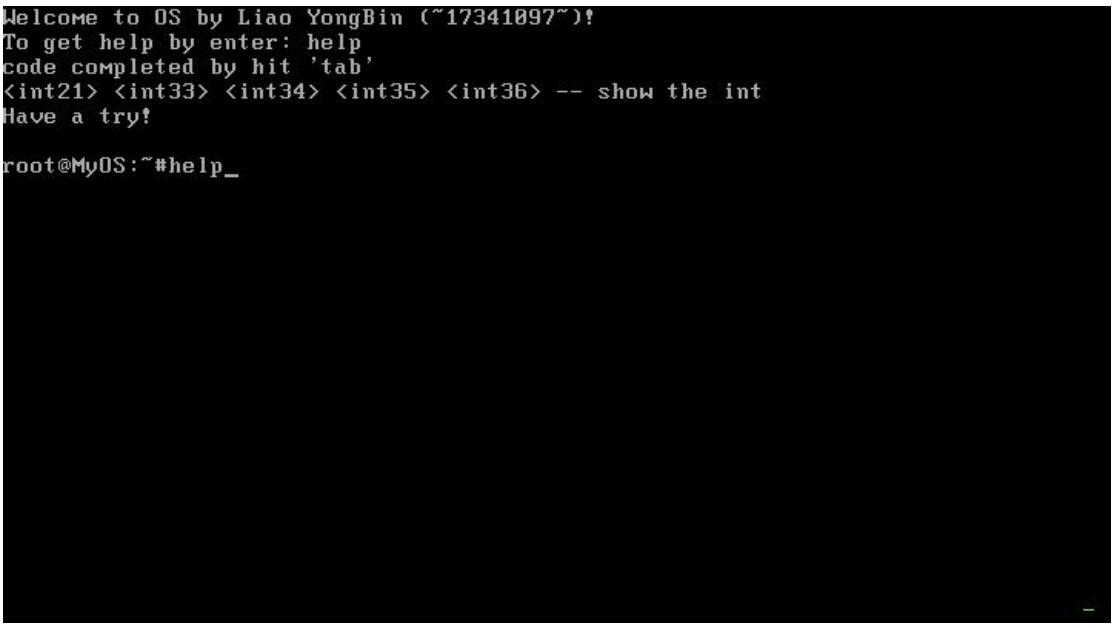
当输入一个残缺的字符串时，按下 tab 键，会自动补全到最相近的指令，实现代码如下：（由于需要补全的指令很有限，所以这个功能有点鸡肋）

```
char tablist[10][100]={"ls","cls","map","help","int21","int33","quit","tab funtion test","",""};

int strlen(char *a);

void tab(int length){
    int i = 0;
    int j = 0;
    int is = 1;
    if(length == 0)
        return;
    for(i = 0;i < 8;i++){
        is = 1;
        if(length >= strlen(tablist[i])){
            is = 0;
            continue;
        }
        for(j = 0;j < length;j++){
            if(commands[j] != tablist[i][j]){
                is = 0;
                break;
            }
        }
        if(is == 1){
            for(j = 0;j < strlen(tablist[i]) - length;j++){
                commands[j+length] = tablist[i][j+length];
                printChar(tablist[i][j+length]);
                count = strlen(tablist[i]);
            }
            return;
        }
    }

}
```

按 tab 前

```
Welcome to OS by Liao YongBin (~17341097~)!
To get help by enter: help
code completed by hit 'tab'
<int21> <int33> <int34> <int35> <int36> -- show the int
Have a try!

root@MyOS:~#h_
```

按 tab 后



【实验心得】

这次实验，最大的收获无异于对中断和异步事件的进一步理解。

在实验四的基础之上，实验五的完成非常的简单。当然，一些比较复杂 c 函数由于我的程序设计没学好，我感觉写不来就放弃了（比如 scanf 函数等）。而其他系统服务程序，写起来还是很轻松的，就是简单的 c 语言与汇编混合，这些在实验一至四已经学得很好了。

此外，由于内核的扩大，我不得不学习了一些软盘的基本知识，如磁道号等等的空间划分，为之后系统的空间管理打下基础。

实验五的关键部分还是写中断，相比实验四只是加入了 ah 功能号的区分，仅仅只要用 cmp 语句就能做到功能划分了，所以还是很轻松的。

此外，我还写了一个代码补全的 tab 功能，写起来还是能发现自己的一些小问题，比如变量初始化，数据保护等等问题。

总的来说，在实验 1-4 的基础之上，实验 5 有点简单过头（当然，要实现一些 c 语言的库函数还是异常困难的）

【代码清单】

Project5
- --操作系统
  - --OS.COM
  - --OS.MAP
  - --KERNAL.OBJ
  - --KERNAL.C
  - --KLIBA.ASM

    --MYOS.ASM
    --MYOS.OBJ
    --services.asm
    --clib.asm
--工具包
    --DOSBox.exe（只是快捷方式）
    --mydoc.conf (DOSBOX 的配置文件)
    --nasm.exe
    --nasmpath.bat
    --TASM.EXE
    --TCC.EXE
    --TLINK.EXT
--软盘文件
    --实验 5.flp
--引导程序
    --loading.asm
    --loading.com
--用户程序
    --a1
    --a1.asm
    --a2
    --a2.asm
    --a3
    --a3.asm
    --a4
    --a4.asm