

具有中断处理的内核

具有中断处理的内核

个人信息

实验题目

实验目的

实验要求

实验内容

实验方案

所用工具

虚拟机配置

实验原理

代码关键部分

标准库

cstdio.c

内核

kfun.asm

ckfun.c

kernel.asm

统计字符串程序

count.asm

实验过程

生成com文件

写盘并且运行

运行虚拟机

实验总结

参考资料

个人信息

- 院系：数据科学与计算机学院
- 年纪：2018
- 姓名：王天龙
- 学号：18340168

实验题目

具有中断处理的内核

实验目的

1. PC系统的中断机制和原理
2. 理解操作系统内核对异步事件的处理方法
3. 掌握中断处理编程的方法
4. 掌握内核中断处理代码组织的设计方法
5. 了解查询式I/O控制方式的编程方法

实验要求

1. 知道PC系统的中断硬件系统的原理
2. 掌握x86汇编语言对时钟中断的响应处理编程方法

3. 重写和扩展实验三的的内核程序，增加时钟中断的响应处理和键盘中断响应。
4. 编写实验报告，描述实验工作的过程和必要的细节，如截屏或录屏，以证实实验工作的真实性

实验内容

1. 编写x86汇编语言对时钟中断的响应处理程序：设计一个汇编程序，在一段时间内系统时钟中断发生时，屏幕变化显示信息。在屏幕24行79列位置轮流显示'|'、'/'和'\'(无敌风火轮)，适当控制显示速度，以方便观察效果，也可以屏幕上画框、反弹字符等，方便观察时钟中断多次发生。将程序生成COM格式程序，在DOS或虚拟环境运行。
2. 重写和扩展实验三的的内核程序，增加时钟中断的响应处理和键盘中断响应。在屏幕右下角显示一个转动的无敌风火轮，确保内核功能不比实验三的程序弱，展示原有功能或加强功能可以工作。
3. 扩展实验三的的内核程序，但不修改原有的用户程序，实现在用户程序执行期间，若触碰键盘，屏幕某个位置会显示"OUCH! OUCH!"。
4. 编写实验报告，描述实验工作的过程和必要的细节，如截屏或录屏，以证实实验工作的真实性








实验方案

所用工具

实验平台是windows10系统，使用的软件有：虚拟机软件vmware workstation pro、汇编语言编译器tasm、c语言编译器tcc、链接器tlink、可视化编译十六进制文件内容工具winhex、代码编辑器visual studio 2019

虚拟机配置

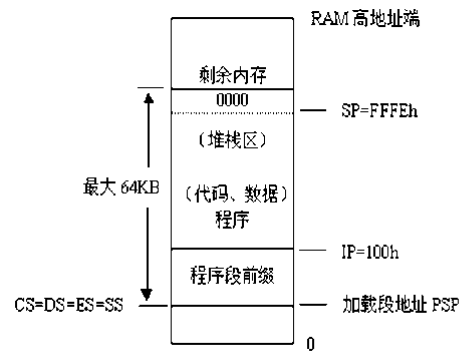
虚拟机为1cpu，内存为4MB，使用一个1.44MB大小的软盘，选择从软盘启动

设备	摘要
 内存	4 MB
 处理器	1
 CD/DVD (IDE)	自动检测
 软盘	正在使用文件 D:\computer_op...
 网络适配器	NAT
 声卡	自动检测
 显示器	自动检测

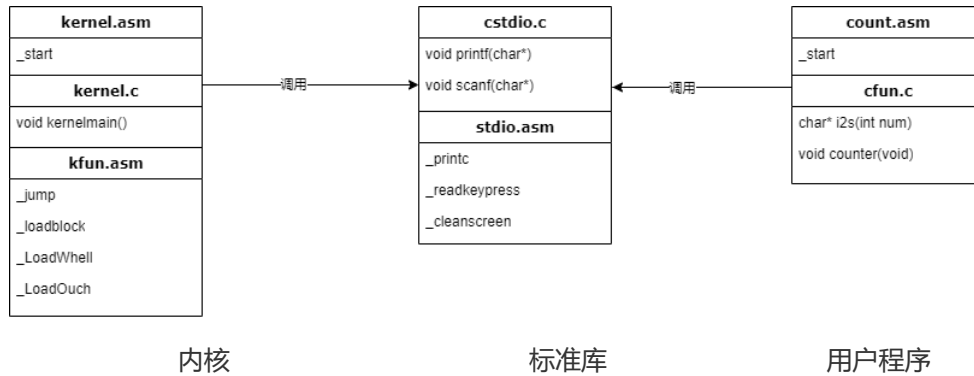
实验原理

要实现自己定义操作系统对时间中断和键盘中断的响应，就要修改中断向量表，让中断向量表的地址指向要执行的代码，这样在中断响应过程中，系统就会执行。中断向量表在内存中最低端的1K字节。时间中断的中断号是8，所以我们要把 $8 \times 4 = 32$ 的内容修改为处理时钟中断的代码的偏移量，把 $8 \times 4 + 2$ 的内容修改为代码的段地址；同理，键盘中断的终端号是9，所以要修改 $9 \times 4 = 36$ 和 $9 \times 4 + 2 = 38$ 两个地方的内容。在屏幕右下角显示一个风火轮，用循环显示'|'、'/'、'\来实现，字符之间的变换由时间中断驱动，为了显示效果，每5次时间中断切换一次。在处理键盘中断的时候，不仅要打印"OUCH! OUCH!"，还要调用原来的9号中断，这是为了不影响从键盘输入，这就要求在修改中断向量表之前，要先保存原来处理中断响应的代码的地址，并在自己的代码里调用。

这次实验还优化了用户程序返回操作系统的过程。用户程序时COM文件，被装载到段的0x100处，前面的0x100个字节时程序前缀(PSP)，PSP里面有返回操作系统的指令，是加载用户程序是由系统加上去。用户程序返回时，返回到程序段前缀，然后经过程序段前缀返回到操作系统。栈顶的0000用于程序结束时ret返回PSP。



代码关键部分



标准库

这部分的代码和实验三基本没有变化，主要改进了scanf函数，让其可以处理退格键(Backspace)，从而实现删除字符的功能。

cstdio.c

这个文件里的函数利用stdio.asm里面的函数实现更加高级的功能。

char* scanf(void)函数的改进地方

```
extern char readkeypress(void);
extern void printc(char c);
void scanf(char *s)/*传进一个地址，把字符串写在那里*/
{
    int i = 0;
    char c;
    while (1)
    {
        c = readkeypress();
        if (c == 0x0d)/*读到回车就停止*/
        {
            s[i] = 0;/*字符最后面置为0*/
            /*换行*/
            printc(0x0a);
            printc(0x0d);
            break;
        }
        else if (c == 0x08)/*Backspace*/
        {
            if (i == 0)/*输入的字符个数为0，无法退格*/
            {
                continue;
            }
        }
    }
}
```

```

    }
    i--; /*字符数减一*/
    printf(0x08); /*让光标回退一个，即回到要删除的字符的位置*/
    printf(' '); /*打印一个空格，看起来就像字符不见了*/
    printf(0x08); /*回退一格*/
    continue;
}
s[i] = c;
i++;
printf(c);
}
return;
}
}

```

内核

kfun.asm

这里增加了装载处理中断响应的函数_LoadWhell、_LoadOuch，以及改进_jump函数

_LoadWhell函数的实现

这个函数修改中断向量表，让8号中断向量指向显示风火轮的代码

```

public _LoadWhell

wheelDelay equ 5;风火轮延迟

_LoadWhell proc near
    push ax
    push es
    CLI;把IF标志位置为0，关中断，防止在装载过程中跳转
    xor ax,ax
    mov es,ax
    lea ax,wheel
    ;0:20h和0:22h分别放偏移量和段地址
    mov [es:20h],ax
    mov ax,cs
    mov [es:22h],ax
    mov es,ax
    STI;开中断

    pop es
    pop ax

    ret
wheel: ;当有时钟中断请求到来时，系统就会跳转到这里执行

    push ds;保护寄存器
    push es
    push ax
    push bx

    mov ax,cs
    mov ds,ax

    ;'|','/','\'轮流显示
    ;每5个中断响应到来显示下一个

```

```

    lea bx, count; 判断count的值是否为0, 为0则显示下一个字符
    mov al, [bx]
    dec al
    mov [bx], al
    cmp al, 0
    jnz exit
    mov al, wheelDelay
    mov [bx], al

    lea bx, show

    mov al, [bx]
    xor ah, ah
    inc ax
    cmp ax, 4
    jnz next1
    mov ax, 1
next1:
    mov [bx], al
    add bx, ax; 获取要显示的字符的偏移量, [bx]为要显示的字符

    mov ax, 0b800h
    mov es, ax

    mov al, [bx]
    mov ah, 07h; 白字黑底
    mov bx, 3998; 第24行79列的显存地址时0b800h: 3998, 即(24*80+79)*2=3998
    mov es: [bx], ax

exit:
    ; 发送EOI给8259A
    mov al, 20h
    out 20h, al
    out 0a0h, al

    ; 恢复寄存器
    pop bx
    pop ax
    pop es
    pop ds

    iret; 中断返回用iret, 比retf多了一个恢复标志寄存器的内容
_LoadWheel endp

; 数据段如下
count label byte
    db wheelDelay
show label byte
    db 1; 记录显示第几个字符, [show+[show]]为要显示的字符
    db '|'
    db '/'
    db '\'

```

_LoadOuch函数的实现

这个函数修改中断向量表，让9号中断向量指向显示"OUCH! OUCH!"的代码，实现有键盘按下时，在屏幕的23行，60列开始显示"OUCH! OUCH!"，松开按键时，字符串消失，同时还不影响正常的字符输入。

```
public _LoadOuch

_LoadOuch proc near
    push ax
    push bx
    push es

    ;先获取原来原来的9号中断向量
    xor ax,ax
    mov es,ax
    mov ax,es:[24h]
    push ax
    mov ax,es:[26h]
    push ax

    ;把9号中断向量保存到int9处
    lea bx,int9
    pop ax
    mov [bx+2],ax
    pop ax
    mov [bx],ax

    mov bx,cs;bx为段地址
    lea ax,OUCH;ax为偏移量

    CLI;关中断，装填新的中断向量
    mov es:[24h],ax
    mov es:[26h],bx
    STI

    pop es
    pop bx
    pop ax

    ret

OUCH:
    ;保护寄存器
    push ds
    push es
    push ax
    push bx
    push cx
    push si

    mov ax,cs
    mov ds,ax

    mov ax,0b800h
    mov es,ax

    lea si,ouchMsgLength;获取字符串长度，即打印次数
    mov cx,[si]
```

```

lea si,ouchMsg;获取首字符偏移量
mov bx,3800;(23*80+60)*2=3800, 第23行第60列
mov ah,07h

again1;;打印字符串
mov al,[si]
mov es:[bx],ax
inc si
inc bx
inc bx
loop again1

;延时一段时间后,情况打印的字符串
mov cx,50000
delay2:
push cx
mov cx,3000
delay1:
loop delay1
pop cx
loop delay2

;通过在原来位置打印空格清空字符串,达到键盘松开,字符串消失的效果
lea si,ouchMsgLength
mov cx,[si]
mov bx,3800
mov ah,07h
mov al,' '
again2;;清空字符
mov es:[bx],ax
inc bx
inc bx
loop again2

;模仿int的工作方式,调用原来的int 9,读键盘的扫描码

;保存标志寄存器
pushf
;把IF,TF标志位置为0
pushf
pop ax
and ah,11111100b
push ax
popf
;把返回地址压栈
mov ax,cs
push ax
lea ax,ouchRetp
push ax
;获取要跳转到的cs,ip,把它们压栈,用retf跳转过去
lea bx,int9
mov ax,[bx+2]
push ax
mov ax,[bx]
push ax
retf

```

```

ouchRetp:

;由于在原来的9号中断里就已经有发EOI的操作了，所以这里无需再发一次
;    mov al,20h
;    out 20h,al
;    out 0a0h,al
;恢复寄存器
    pop si
    pop cx
    pop bx
    pop ax
    pop es
    pop ds

    iret;返回
_LoadOuch endp

;数据段如下
ouchMsg label byte
    db "OUCH! OUCH!"
ouchMsgLength label byte
    dw $-ouchMsg
int9 label byte
    dw 0,0;记录原来int 9的ip, cs

```

自定义键盘中断的时候，为了不影响从键盘输入，还要调用原来的9号中断来处理。调用时，要模仿int的工作流程，即把标志寄存器压栈，清空IF,TF标志位，把返回的段地址、偏移量压栈，然后跳转。

_jump函数的改进

在实验三的时候，用户程序返回操作系统时，是简单粗暴地返回到操作系统开始的地方，而不是跳转点。这里进行了改进，实现从那里跳过去，就跳回到那里。用户程序被加载到段开始偏移0x100个字节处，前面的0x100个字节是程序前缀(里面就一条retf指令)，由操作系统加载附加上去的，用于返回操作系统。在跳转之前把ds,es寄存器压栈保护，并且把用户栈给初始好(ss=用户段地址，sp=0x0000)，并把操作系统的栈的段地址、栈顶指针压进去，然后压一个0x0000，给用户程序用retf返回到程序前缀，然后压返回点的段地址及偏移量。完成上述工作后，进行跳转。用户程序返回时，先用ret指令跳转到程序段前缀执行，程序段前缀里用retf远跳转回到操作系统。

用户程序所在段安排如下：

0xfffe	操作系统的栈顶指针
0xfffc	返回点的段地址
0xfffa	返回点的偏移量
0xff8	0x0000
0x100	用户程序开始
0x00	retf

代码如下：

```

_jump proc near;两个参数,要跳到的cs:ip

```


;在这里就把用户程序的ss=cs, sp=0000h给初始化好, 并且把这里的ss,sp压到用户程序的栈里

```
push bp
mov bp,sp
```

```
push ax
push bx
push cx
```

```
push es
```

```
mov bx,[bp+4];cs
mov ax,[bp+6];ip
```

```
;把PSP写到cs:0处
mov es,bx
lea si,PSPBegin
mov di,0
;movsb 把ds:si->es:di
lea cx,PSPEnd
sub cx,si;循环次数
rep movsb;循环写
```

```
;在这里就把用户程序的堆栈给设好, 把这里的sp压栈, 再把这里的cs, ip压栈
mov cx,sp
;切换到用户栈
mov ss,bx
mov sp,0
```

```
push cx;压sp
push cs;压这里的cs
lea cx,retf
push cx;压返回的ip
```

```
xor cx,cx
push cx;压0x0000, 个用户程序用retf返回到程序段前缀
;把要跳转的cs, ip压栈, 用于retf跳转
push bx;要跳转的cs
push ax;要跳转的ip
retf;跳转
```

retf:

```
;把内核栈切换回来
pop bx;sp
mov ax,cs
mov ss,ax
mov sp,bx;切换完成
```

```
pop es
```

```
pop cx
pop bx
pop ax
```

```
pop bp
```

```
ret
```

;程序段前缀

```

PSPBegin:
    retf
PSPEnd:nop

_jump endp

```

ckfun.c

这里主要增加了装载中断响应的过程，修改如下

```

extern void LoadOuch();
extern void Loadwheel();
/*其他声明*/
void kernelmain()
{
    char c;
    LoadOuch();
    Loadwheel();
    /*其他代码*/
}

```

kernel.asm

这里相对于实验三没有修改

统计字符串程序

用户程序中，最大的修改是返回操作系统时，使用ret，并且用户程序不能修改ss，sp寄存器的值。还有就是就是在用户程序所用的寄存器，要压栈保护

count.asm

这里实现了_start函数，_start的作用是跳转到counter函数，并且当counter函数执行完时，跳转回到监控程序

```

_start:
    ;保护寄存器
    push ax
    push es
    push ds

    mov ax,cs
    mov es,ax
    mov ds,ax
    call near ptr _counter
    ;恢复寄存器
    pop ds
    pop es
    pop ax

    ret

```

其他地方就没什么改动的地方了

实验过程

生成com文件

在DoxBox里使用命令

```
tasm kernel.asm kernel.obj
tasm stdio.asm stdio.obj
tcc -c cstdio.c
tasm kfun.asm kfun.obj
tcc -c ckfun.c
tlink /3 /t kernel.obj stdio.obj cstdio.obj kfun.obj ckfun.obj, kernel.com
```

即可生成内核程序的com执行体。用户程序也类似。

写盘并且运行

写盘式，引导程序在首扇区，监控程序在2、3扇区，用户程序在4-9号扇区，可以用I指令查看用户程序所在的扇区

运行虚拟机

风火轮显示效果



按下键盘

```
M for message, l to list, c to clean screen, 1-5 to select program: aaaaaaaaaa
aa

OUCH! OUCH!
```

```
M for message, l to list, c to clean screen, 1-5 to select program: 1
a
A
A

OUCH! OUCH!
```

实验总结

通过这次试验，加深了对操作系统对中断的处理的理解。同时，清楚了com文件的组织，了解了程序前缀段，并用更加接近Dos操作系统的方式从用户程序返回操作系统。实验过程中，遇到的最大的问题是处理来自键盘的中断请求时，要从60h端口读一个字节，这个字节时按键的扫描码，如果不读，会导致中断请求会一直存在，使操作系统阻塞。但是，简单地从60h端口读一个字节还不行，因为则会影响到从键盘输入，所以，最好的方法就是打印完"OUCH! OUCH!"后，调用原来的9号中断来进行处理。

参考资料

<https://www.kancloud.cn/digest/protectedmode/121463>

https://blog.csdn.net/csdn_blog_lcl/article/details/54926726