中国科学技术大学计算机学院
《操作系统原理与设计》

实验题目：shell,中断和时钟

学生姓名：叶子昂

学生学号：PB20020586

完成时间：2022年4月14日

实验题目

shell，中断和时钟

实验目的

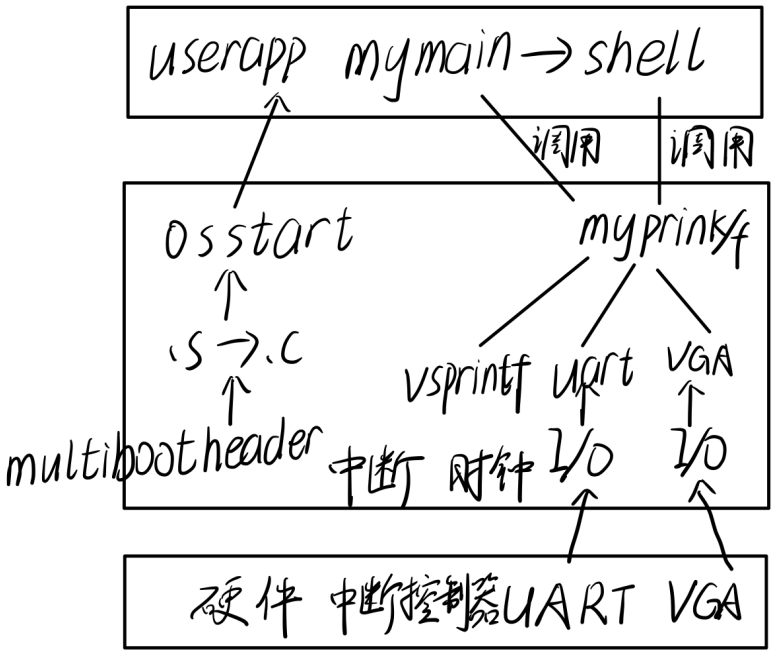
- 实现简单的shell程序，提供一些基本命令
- 中断实现和初始化中断控制器
- 实现时钟周期性中断

实验环境

vlab, wsl2, qemu

实验内容

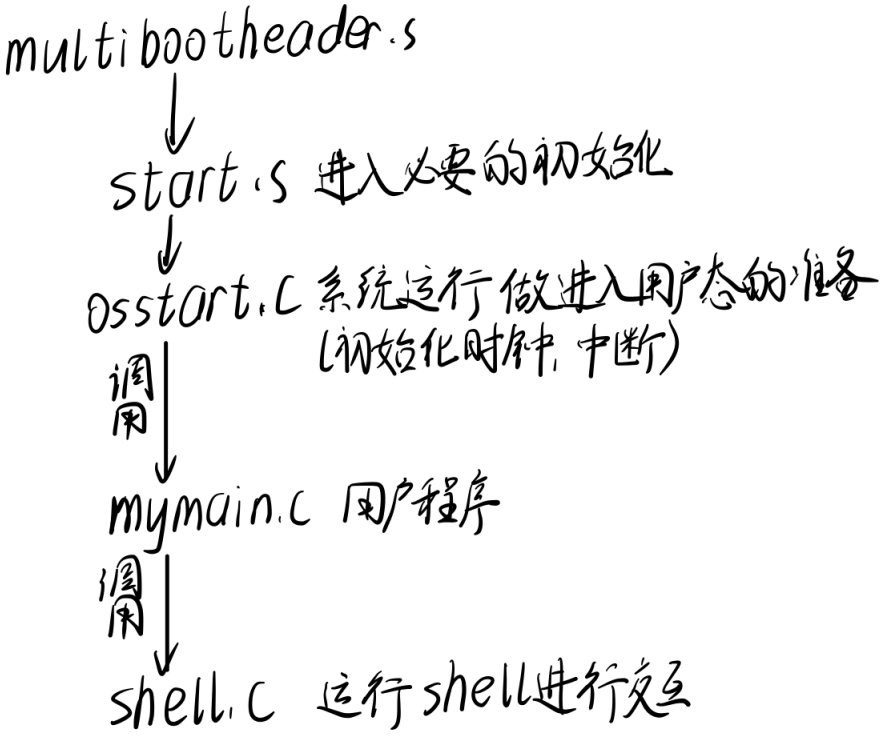
1. 软件框图



myos运行在在qemu构建的虚拟硬件环境，myos通过ostart调用用户程序mymain进入用户态，其中调用shell进入shell程序，myos通过提供myprintk/f为自身和用户程序提供格式化输出的功能。时钟和中断在进入用户程序前初始化完成。

2.系统启动流程

在qemu构建的虚拟环境中通过multibootheader启动，在start32.S中进行栈的构建和运行环境的初始化（包括中断向量表维护，清除bss段），然后系统由osStart.c接管系统开始运行初始化中断控制器，开中断，通过“调用”mymain来运行用户程序，最后main函数调用shell进入shell环境。



3.主功能模块及代码实现

- start32中的两种中断

> 调用终端服务程序前先通过pushf，pusha保护现场，服务程序执行完成后通过popa，，popf恢复现场。

```
time_interrupt:
# todo//done
    cld
    pushf
    pusha
    call tick
    popa
    popf
    iret

ignore_int1:
# todo//done
    cld
    pusha
    call ignoreIntBody
    popa
    iret
```

- 初始化8253和8259

> 按规定的顺序向端口写入数据

```
void init8253(void){
    //todo//done
```

```
    outb(0x43,0x34);
    outb(0x40,0x9B);//分频参数为0x2E9B
    outb(0x40,0x2E);
}

void init8259A(void){
    //todo//done
    outb(0x20,0x11);
    outb(0x21,0x20);
    outb(0x21,0x04);
    outb(0x21,0x3);
    outb(0xA0,0x11);
    outb(0xA1,0x28);
    outb(0xA1,0x02);
    outb(0xA1,0x01);
}
```

- 开关中断

> 通过汇编sti，cli实现

```
    .globl enable_interrupt
enable_interrupt:
# todo//done
    sti
    ret

    .globl disable_interrupt
disable_interrupt:
# todo//done
    cli
    ret
```

- 时钟中断的服务函数tick

> 每次被调用时使system_ticks++，同时由system_ticks求出时分秒，并调用oneTIckUpdateWallClock()设置显示时间或用户自定义的函数。

```
void tick(void){
    system_ticks++;
    //todo//done
    int MS;
    MS = (10*system_ticks)%1000;
    SS = ((10*system_ticks)/1000)%60;
    MM = (((10*system_ticks)/1000)/60)%60;
    HH = ((((10*system_ticks)/1000)/60)/60)%24;
    oneTickUpdateWallClock(HH, MM, SS);
    return;
}
```

- 设置与获取时钟

> **hook的思考**：系统在tick中执行的始终为oneTickUpdateWallClock()函数，但它调用的是一个函数指针(*wallClock_hook)(int, int, int)，在本系统中该指针被默认设置为指向系统提供的setWallClock，但系统也提供了setWallClockHook给用户，这就赋予了用户重新编写自己的“setWallClock”的能力实现了机制与策略相分离，这使得用户能够自定义完成一些特别的需求。

```
void (*wallClock_hook)(int, int, int) = 0;

void oneTickUpdateWallClock(int HH, int MM, int SS){//调用 wallClock_hook 函数
    if(wallClock_hook) wallClock_hook(HH,MM,SS);
}

void setWallClockHook(void (*func)(int, int, int)) {//设置 wallClock_hook 的值
    wallClock_hook = func;
}

void setWallClock(int HH,int MM,int SS){
    //todo//done
    int args[3] = { HH, MM, SS };
    char* format = "%02d:%02d:%02d";
    vsprintf(Buf, format, args);
    int len = strlen(Buf);
    for (int i = 0;Buf[i]!='\0';i++){//在右下角输出显示时间
        put_char2pos(Buf[i],0x07,80*25-len+i);
    }
}

void getWallClock(int *HH,int *MM,int *SS){//从显存中获取时间信息
    //todo//done
    unsigned short int* p = VGA_BASE+(80*25-8)*2;
    *HH = ((*p & 0x00ff)-48)*10+((*(p+1) & 0x00ff)-48);
    p+=3;
    *MM = ((*p & 0x00ff)-48)*10+((*(p+1) & 0x00ff)-48);
    p+=3;
    *SS = ((*p & 0x00ff)-48)*10+((*(p+1) & 0x00ff)-48);
}
```

- 交互的shell程序

> shell使用静态注册方式提供help，cmd，time三个指令使用方式分别为
>
> help [command] (可以接多个)
>
> cmd (无参数)
>
> time (无参数)

获取args和argc

> 找到非空白字符向后获取一个string，再向后找非空白字符，再获取string循环至遍历整个命令。

```
void cmd_get(const char* cmd_string, int cmd_len, int* argc, char (*argv)[8]){
    int index = 0;
    *argc = 0;
    for(int i = 0; i<cmd_len; i++) {
        if(cmd_string[i]>=33 && cmd_string[i]<=126) {//非控制字符
            (*argc)++;
```

```
            int j;
            for(j=i;j<cmd_len && cmd_string[j]>=33 && cmd_string[j]<=126 ; j++){//get
a string
                argv[*argc-1][index++] = cmd_string[j];
            }
            i = j;
            argv[*argc-1][index] = '\0';
            index = 0;
        }
    }
}
```

- 字符串比较

```
int strcmp(char* str1,char* str2){
    int i;
    for(i=0;str1[i]!='\0' && str2[i] !='\0';i++){
        if(str1[i]!=str2[i])return 0;
    }
    if(str1[i]!='\0' | str2[i]!='\0')return 0;
    else return 1;
}
```

- 三条命令本体

```
extern myCommand cmd;//ahead declration
extern myCommand time;
extern myCommand help;

int func_cmd(int argc, char (*argv)[8]){
    if(argc != 1) myPrintf(14,"notice: cmd needs no args\n");
    myPrintf(0x07,cmd.name);myPrintf(0x07," ");
    myPrintf(0x07,time.name);myPrintf(0x07," ");
    myPrintf(0x07,help.name);
    myPrintf(0x07,"\n");
}
myCommand cmd={"cmd\0","List all command\n\0",func_cmd};

int func_time(int argc, char (*argv)[8]){
    if(argc != 1) myPrintf(14,"notices: time needs no args\n");
    int h,m,s;
    h=0;m=0;s=0;
    getWallClock(&h,&m,&s);
    myPrintf(0x07,"%02d:%02d:%02d\n",h,m,s);
}
myCommand time={"time\0","show the current time\n",func_time};

int func_help(int argc, char (*argv)[8]){
    if(argc == 1) {
        myPrintf(0x02,help.help_content);
    }
    else{
        for (int i = 1; i < argc; i++)//help all command after the help
        {
```

```
            if(strcmp(argv[i],"help")) myPrintf(0x02,help.help_content);
            else if(strcmp(argv[i],"cmd")) myPrintf(0x02,cmd.help_content);
            else if(strcmp(argv[i],"time")) myPrintf(0x02,time.help_content);
        }

    }
}
myCommand help={"help\0","Usage: help [command]\nDisplay info about
[command]\n\0",func_help};
```

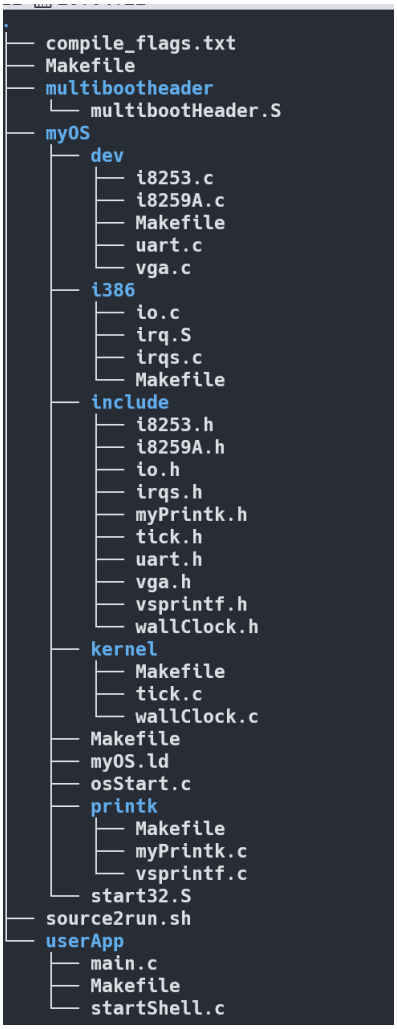
- shell程序

```
int argc;
    char argv[8][8];
    int h,m,s = 0;
    getWallClock(&h,&m,&s);
    myPrintf(0x07,"Welcome to myOS! time now is %02d:%02d:%02d\n",h,m,s);
    do{
        BUF_len=0;
        myPrintf(0x12,"PB20020586");
        myPrintf(0x07,">>");
        while((BUF[BUF_len]=uart_get_char())!='\r'){
            uart_put_char(BUF[BUF_len]);//将串口输入的数存入BUF数组中
            BUF_len++;  //BUF数组的长度加
        }
        BUF[BUF_len]='\0';
        uart_put_chars(" -pseudo_terminal\0");
        uart_put_char('\n');
        append2screen(BUF,0x07);
        append2screen("\n",0x07);
        cmd_get(BUF,BUF_len,&argc,argv);
        if(strcmp(argv[0],"help")) {//根据不同的命令执行对应的func
            help.func(argc,argv);
        }
        else if(strcmp(argv[0],"cmd")) {
            cmd.func(argc,argv);
        }
        else if(strcmp(argv[0],"time")) {
            time.func(argc,argv);
        }
        else myPrintf(12,"undeclared command\n");

    }while(1);
```

4.源代码组织结构说明

- 目录组织结构



- makefile组织结构

```
.
|--MULTI_BOOT_HEADER
|    |__output/multibootheader/multibootHeader.o
|--OS_OBJS
     |__MYOS_OBJS
     |    |__output/myOS/start32.o
     |    |__output/myOS/osStart.o
     |    |__DEV_OBJS
     |    |    |__output/myOS/dev/uart.o
     |    |    |__output/myOS/dev/vga.o
     |    |    |__output/myOS/dev/i8259A.o
     |    |    |__output/myOS/dev/i8253.o
     |    |__I386_OBJS
     |    |    |__output/myOS/i386/io.o
     |    |    |__output/myOS/i386/irqs.o
     |    |    |__output/myOS/i386/irq.o
     |    |__PRINTK_OBJS
     |    |    |__output/myOS/printk/myPrintk.o
     |    |    |__output/myOS/lib/vsprintf.o
     |    |__KERNEL_OBJS
     |         |__output/myOS/kernel/tick.o
     |         |__output/myOS/kernel/wallClock.o
     |__USER_APP_OBJS
          |__output/userApp/main.o
```

5.代码布局说明

Section	Offset	说明
.multibootheader	0	开头
.text	16	multibootheader占4×3=12,ALIGN(8)对齐后到16
.data	8016	通过myprint打印出
.bss	11008	通过myprint打印出
_end	12272	通过myprint打印出

6.编译过程说明

- 编译所用指令

  > make

- 编译过程

  1. 根据makefile找到并编译各个.c,.s文件到.o文件。
  2. 根据目标文件连接规则按照.ld文件描述进行连接生成myOS.elf

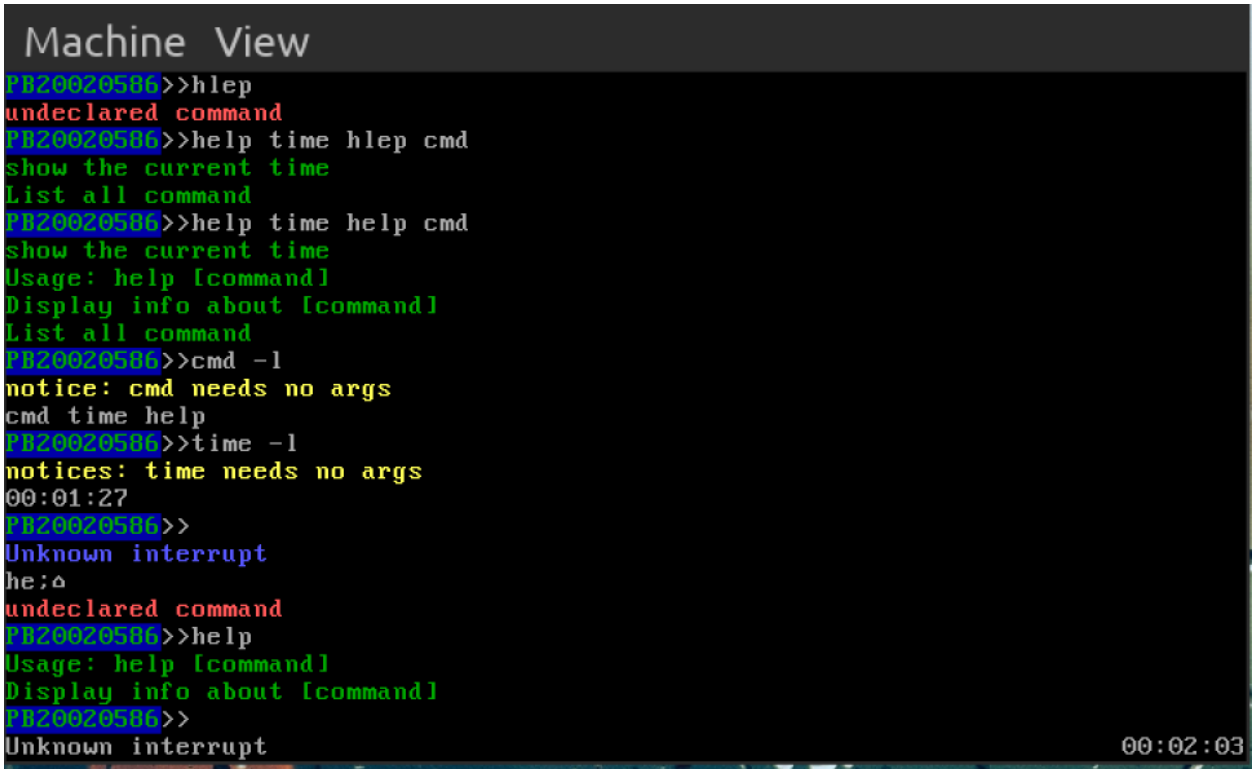
7.运行和运行结果说明

- 运行方式

  > zsh source2run.sh

- 运行结果

  qemu界面


Machine View
PB20020586>>hlep
undeclared command
PB20020586>>help time hlep cmd
show the current time
List all command
PB20020586>>help time help cmd
show the current time
Usage: help [command]
Display info about [command]
List all command
PB20020586>>cmd -l
notice: cmd needs no args
cmd time help
PB20020586>>time -l
notices: time needs no args
00:01:27
PB20020586>>
Unknown interrupt
he;o
undeclared command
PB20020586>>help
Usage: help [command]
Display info about [command]
PB20020586>>
Unknown interrupt
00:02:03

串口伪终端界面

```
help -pseudo_terminal
Usage: help [command]
Display info about [command]
PB20020586>>cmd -pseudo_terminal
cmd time help
PB20020586>>time -pseudo_terminal
00:00:31
PB20020586>>hlep  -pseudo_terminal
undeclared command
PB20020586>>help time hlep cmd -pseudo_terminal
show the current time
List all command
PB20020586>>help time help cmd -pseudo_terminal
show the current time
Usage: help [command]
Display info about [command]
List all command
PB20020586>>cmd -l -pseudo_terminal
notice: cmd needs no args
cmd time help
PB20020586>>time -l -pseudo_terminal
notices: time needs no args
00:01:27
PB20020586>>
Unknown interrupt
he; -pseudo_terminal
undeclared command
PB20020586>>help -pseudo_terminal
Usage: help [command]
Display info about [command]
PB20020586>>
```

8.遇到的问题及解决

- 开始将Vga显存中的数据误认为int型导致getwallclock获取后在输出的时间完全不对。后认识到写入的包括时间ascii码和颜色后经过处理数据解决。

总结与思考

- 通过本次实验，我进一步熟悉了OS的启动方式和流程，能够再已有的代码框架下，理解代码并补全确实功能。
- 实现了简单的shell，熟悉了进入用户态后实现简单交互的方法。
- 了解了中断在底层的原理和实现以及如何初始化中断控制器使其能发出预定的时钟信号，能接受中断并通知系统。
- 学习实践了hook机制，通过机制与策略相分离，使用户能够自定义setwallclock供系统使用。