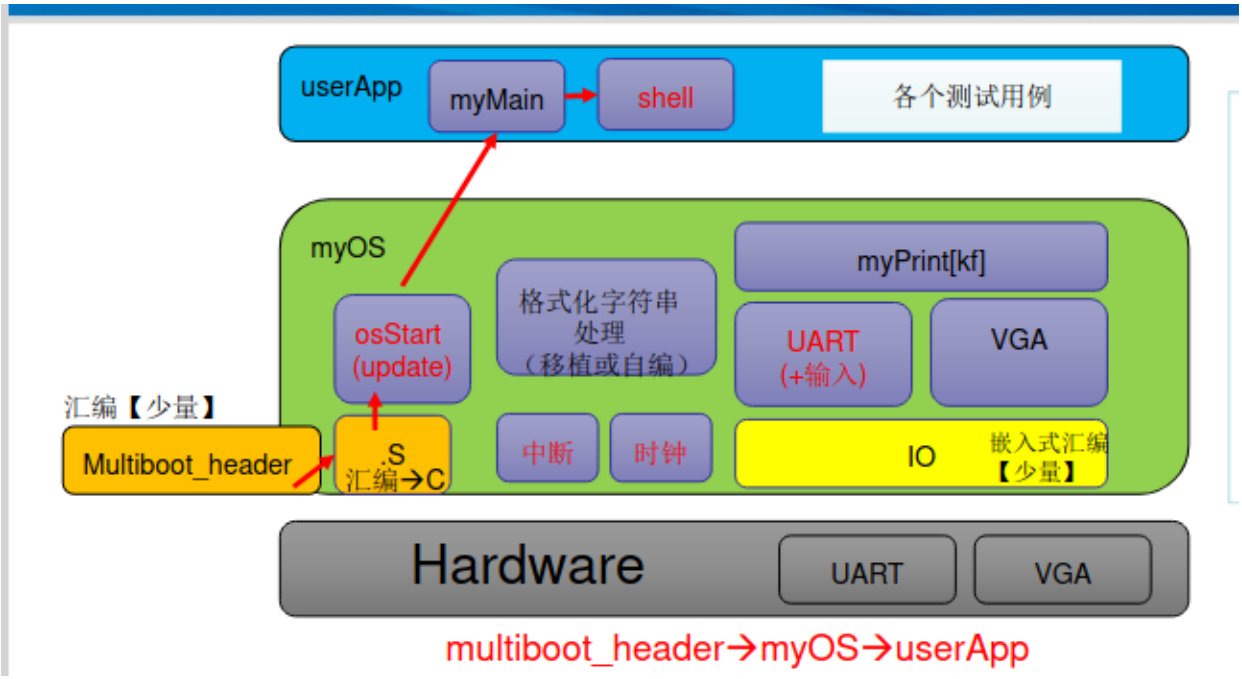
lab3实验报告

毛陈诚 PB20111694

详细说明主要功能模块及其实现，画出流程图



1. `multibootheader.S`是一段操作系统启动代码，在设置好启动头标的代码后，将程序引导执行`start.s`
2. `start.s`设置堆和栈的大小，然后将BSS段清零，将程序引导执行`osstart.c`
3. `osstart`调用`mymain`函数，用于将`myOS`层与`userApp`层连接。

   本实验新增了：中断处理、时钟
4. `myprint[kf]`对输入字符串进行格式化处理，是用户可以调用的函数
5. `vga/uga`实现vga的相关功能，清屏和屏幕输出,调用`IO`的`inb`,`outb`
6. `IO`实现端口输出，采用嵌入式汇编
7. `userApp` 是最外层的接口来使操作者调用`myprintk`或`myprintf`函数实现输出功能
8. `shell`

详细说明主流程及其实现，画出流程图

1. `Multiboot header`中为进入C程序准备好上下文，然后调用`_start`入口
2. 在操作系统初始化完毕后，`OSstart` 调用 `mymain` 函数，转入用户程序运行
3. 用户程序只能调用操作系统定义的接口和用户自定义的函数，这里`mymain`函数调用`myprintk`函数解析字符串
4. `myprintk`函数调用`myOS`中的函数包括:`IO`端口输出;串口`uart`输出,VGA输出

源代码说明（目录组织、Makefile组织）

1. Makefile　　提供，可修改
2. Multibootheader子目录
3. 内核子目录　子目录下可以进一步按功能划分子目录
    - dev: `vga`, `uart`
    - i386: `IO`
    - printk: `myprintk`
4. userApp子目录
5. main.c
6. output子目录（所有编译链接生成的文件在此）
7. source2run.sh　　脚本文件，提供，不要修改

代码布局说明

start32.s

```
time_interrupt:
    cld
    pushf
    pusha
    call tick
    popa
    popf
    iret

    .p2align 4
ignore_int1:
    cld
    pusha
    call ignoreIntBody
    popa
    iret
```

中断处理唤起`tick`

tick.c

更新时钟值

```
void tick(void){
    system_ticks++;
    //你需要填写它
    ms += 10;
    if (ms >= 1000) {
        ms -= 1000;
        ++SS;
```

```
    }
    if (SS >= 60) {
        SS -= 60;
        ++MM;
    }
    if (MM>= 60) {
        MM -= 60;
        ++HH;
    }
    if (HH >= 24) {
        HH -= 24;
    }
    setWallClock(HH,MM,SS);
    return;
}
```

wallclock.c

用于设置已被更新的时钟值，并设置钩子函数`update_tick_hook`,随时间变化而进行维护时间的更新

```
typedef void update_tick_hook(void);
update_tick_hook *tick_hook = 0;
int H=0,M=0,S=0;
void setWallClock(int HH,int MM,int SS){
    H = HH;
    M = MM;
    S = SS;
    //你需要填写它
    if(tick_hook){
        tick_hook();
    }
}

void getWallClock(int *h,int *m,int *s){
    //你需要填写它
        *h = H;
        *m = M;
        *s = S;
}

void setClockHook(update_tick_hook *newhook){
    tick_hook = newhook;
}
```

startshell

变量

设计用于维护命令的数组`commandlist`，与统计命令个数的变量`numcmd`

```
myCommand commandlist[20];
int numcmd = 2;
```

func_cmd

```
    for (int i = 0; i < numcmd; i++)
        myPrintk(0x07, "%s\n", commandlist[i].name);
    return 0;
```

func_help

- 如果`help`只有一个变量，则则调用`help`的help函数
- 如果`help`有二个变量调用指定命令的help函数，
- 否则报错

```
int func_help(int argc, char (*argv)[8]){
    if (argc == 1){
        myPrintk(0x07, "%s\n", commandlist[1].help_content);
    }
    else if(argc == 2){
        for(int i = 0; i < numcmd; i++){
            if (!strcmp(argv[1], commandlist[i].name)) {
                myPrintk(0x07, "%s\n", commandlist[i].help_content);
                return 0;
            }
        }
    }
    else{
        myPrintk(0x07, "%s\n", "ERROR! At most two arguments are allowed!");
    }void scroll_up(void){
    int backline = COLUMNNUMBER*SIZEOFWORD;
    pos = get_pos();
    for (int i = backline; i < pos; i++){
        vga_init_p[i - backline] = vga_init_p[i];
    }
    for(int i = pos -backline; i < pos; i = i + 2){
        vga_init_p[i] = 0x00;
        vga_init_p[i+1] = 0x07;
    }
}
```

initcmdlist

初始化`commandlist`

```
void initcmdlist(void){
    commandlist[0].func = cmd.func;
    strcpy(commandlist[0].help_content ,cmd.help_content);
    strcpy(commandlist[0].name,cmd.name);
    commandlist[1] = help;
}
```

startShell

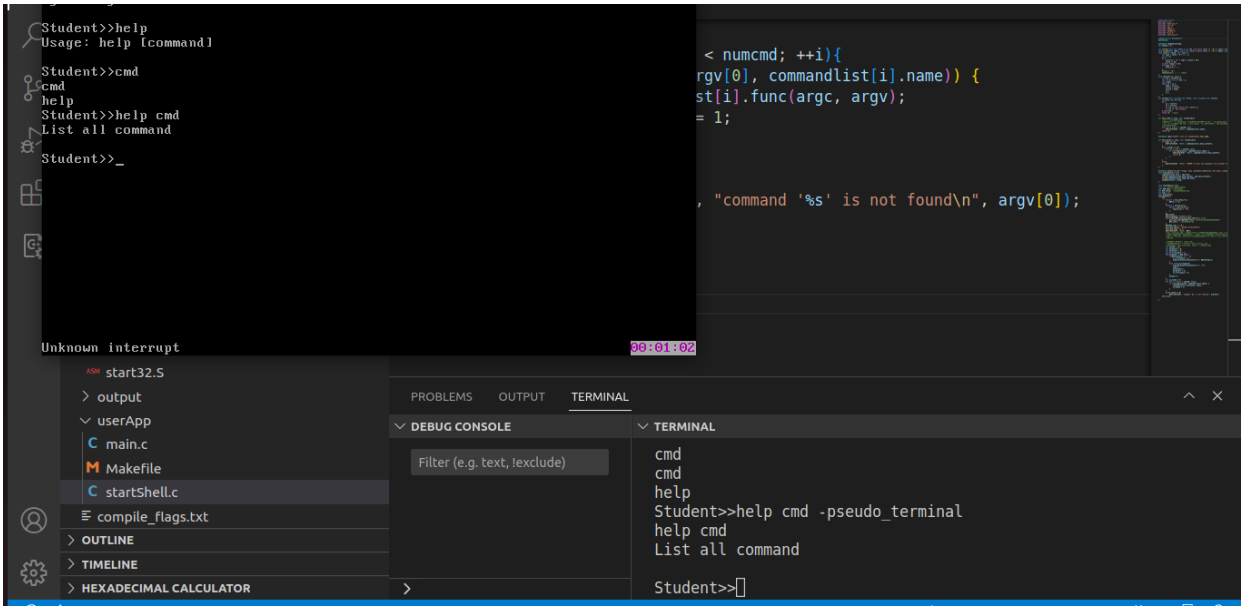
首先分析每个字符并存入`argv`中

```
int nextptr = 0;
        int wordcount = 0;
        int wordstart = 0;
        int is_firstspace = 1;
        while(nextptr < BUF_len){
            if(BUF[nextptr] != ' '){
                is_firstspace = 1;
                argv[wordcount][wordstart++] = BUF[nextptr];
            }
            else if(is_firstspace){
                argv[wordcount][wordstart++] = '\0';
                argc++;
                wordcount++;
                wordstart = 0;
                is_firstspace = 0;
            }
            nextptr++;
        }
```

判断输入命令是否在命令集合中，是则执行对应函数，否则输出错误信息

```
        int is_found = 0;
        for (int i = 0; i < numcmd; ++i){
            if (!strcmp(argv[0], commandlist[i].name)) {
                commandlist[i].func(argc, argv);
                is_found = 1;
            }
        }
        if(is_found == 0)
            myPrintk(0x07, "command '%s' is not found\n", argv[0]);
```

运行和运行结果说明



遇到的问题和解决方案说明

1. 编写`vsprintf`不能调用库函数，比如说`strcpy`等。

   解决方案：自己实现，详细在代码布局中已经说明

2. `hook`函数理解与实现