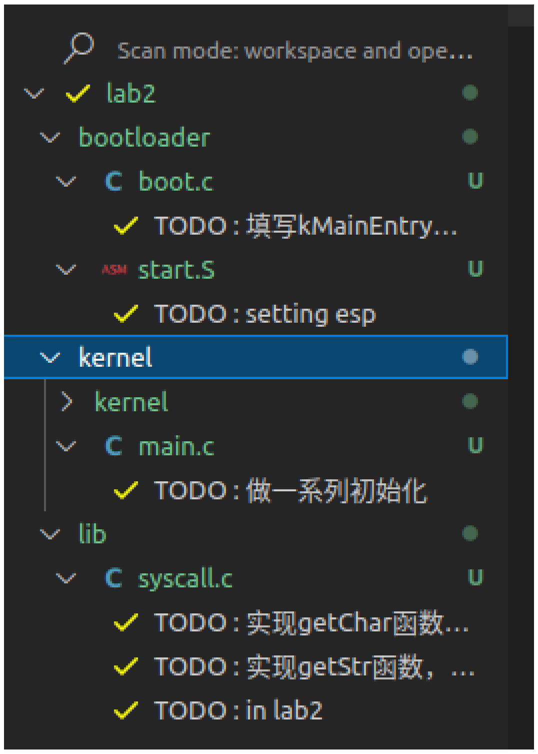
lab2 系统调用

181180050 孔孟荀

1571589383@qq.com

1.TODO 概览

（不完全）



```
Scan mode: workspace and ope...
v ✓ lab2
  v bootloader
    v C boot.c                                U
        ✓ TODO : 填写kMainEntry...
    v ASM start.S                             U
        ✓ TODO : setting esp
  v kernel
    > kernel
    v C main.c                                U
        ✓ TODO : 做一系列初始化
  v lib
    v C syscall.c                             U
        ✓ TODO : 实现getChar函数...
        ✓ TODO : 实现getStr函数，...
        ✓ TODO : in lab2
```

2.preparations

1. boot.c
   完成bootmain.c如下，其功能为： bootmain() loads an ELF kernel image from the disk starting at sector 1 and then jumps to the kernel entry routine.

```
void bootMain(void) {
        int i = 0;
        int phoff = 0x34;
        int offset = 0x1000;
        unsigned int elf = 0x100000;
        void (*kMainEntry)(void);
        kMainEntry = (void(*)(void))0x100000;

        for (i = 0; i < 200; i++) {
                readSect((void*)(elf + i*512), 1+i);
        }

         //TODO: 填写kMainEntry、phoff、offset
        kMainEntry = (void(*)(void))((struct ELFHeader *)elf)->entry;// // Call the entry point from the ELF header
        phoff = ((struct ELFHeader *)elf)->phoff;
        offset = ((struct ProgramHeader *)(elf + phoff))->off;

        for (i = 0; i < 200 * 512; i++) {
                *(unsigned char *)(elf + i) = *(unsigned char *)(elf + i + offset);
        }

        kMainEntry();
}
```

2. start.s
   set esp如下

```
        movl $0x1fffff, %eax # setting esp
        movl %eax, %esp
```

3. doIrq.s
   将irqKeyboard的中断向量号压入栈
4. idt.c
   完成初始化陷阱门、中断门的函数并且填好剩下的表项
5. irqHandle.c

- 补全中断处理程序
- 补全KeyboardHandle，对几种特殊情况的处理，使用到了助教给的提示

```
//(将字符character显示在屏幕的displayRow行displayCol列)
data = character | (0x0c << 8);
pos = (80*displayRow+displayCol)*2;
asm volatile("movw %0, (%1)"::"r"(data),"r"(pos+0xb8000));
```

6. kvm.c
   参照bootloader加载内核的方式即可，需要注意的是用户程序加载到了 0x200000
7. main.c
   做一系列初始化，对应的是教程3.3中的一系列初始化：

- 初始化串口输出
- 初始化中断向量表（initIdt）
- 初始化8259a中断控制器（initIntr）
- 初始化 GDT 表、配置 TSS 段（initSeg）
- 初始化VGA设备（initVga）
- 配置好键盘映射表（initKeyTable）
- 从磁盘加载用户程序到内存相应地址（loadUMain）

8. 实现syscallGetChar、syscallGetStr

3.实现printf()

```
void printf(const char *format,...){
        int i=0; // format index
        char buffer[MAX_BUFFER_SIZE];
        int count=0; // buffer index
        int index=0; // parameter index
        void *paraList=(void*)&format; // address of format in stack
        int state=0; // 0: legal character; 1: '%'; 2: illegal format
        int decimal=0;
        uint32_t hexadecimal=0;
        char *string=0;
        char character=0;
        while(format[i]!=0){


                //  in lab2
                if(format[i]=='%'){
                        i++;
                        switch(format[i]){
                                case 'd':
                                        decimal = *((int *)(paraList + 4 * index));
                                        index++;
                                        count = dec2Str(decimal, buffer, MAX_BUFFER_SIZE, count);
                                        break;
                                case 'x':
                                        hexadecimal = *((uint32_t *)(paraList + 4 * index));
                                        index++;
                                        count = hex2Str(hexadecimal, buffer, MAX_BUFFER_SIZE, count);
                                        break;
                                case 's':
                                        string = *((char **)(paraList + 4 * index));
                                        index++;
                                        count = str2Str(string, buffer, MAX_BUFFER_SIZE, count);
                                        break;
                                case 'c':
                                        character = *((char *)(paraList + 4 * index));
                                        index++;
                                        buffer[count++] = character;
                                        break;
                        }
                }
                else{
                        buffer[count++] = format[i];
                }
                if(count==MAX_BUFFER_SIZE){
                        syscall(SYS_WRITE, STD_OUT, (uint32_t)buffer, (uint32_t)MAX_BUFFER_SIZE, 0, 0);
                }
                i++;
        }
        if(count!=0)
                syscall(SYS_WRITE, STD_OUT, (uint32_t)buffer, (uint32_t)count, 0, 0);
}
```

4.实现getChar, getStr

直接用系统调用

```
char getChar(){ // 对应SYS_READ STD_IN
        //  实现getChar函数，方式不限
        return syscall(SYS_READ,STD_IN,0,0,0,0);
}

void getStr(char *str, int size){ // 对应SYS_READ STD_STR
        //  实现getStr函数，方式不限
        syscall(SYS_READ, STD_STR, (uint32_t)str, size, 0, 0);
}
```

完