221220034-2

实验信息

1. 姓名： 王旭
2. 学号：221220034
3. 专业：计算机科学与技术系
4. 邮箱：2069625874@qq.com

5. 实验进度：已正确完成所有任务

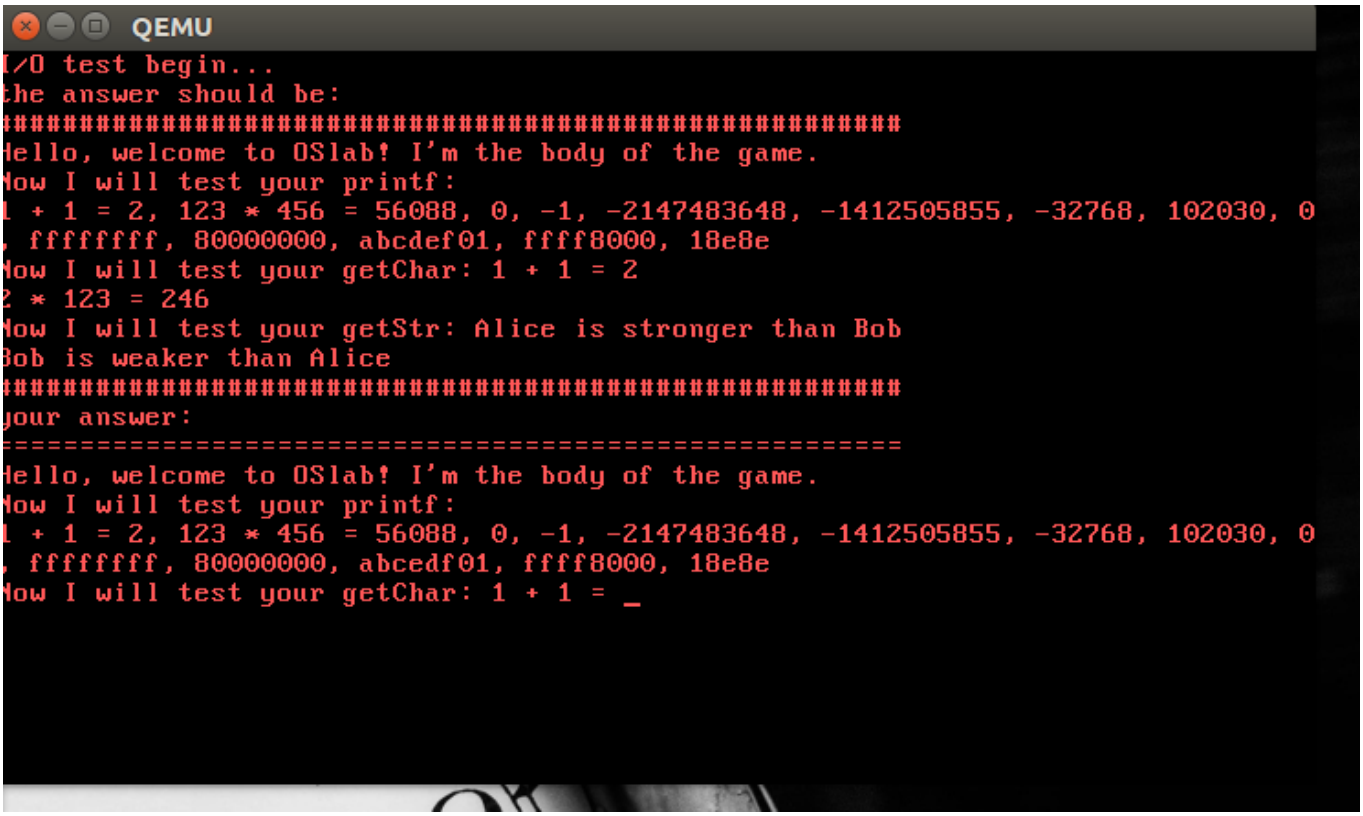
实验主体：

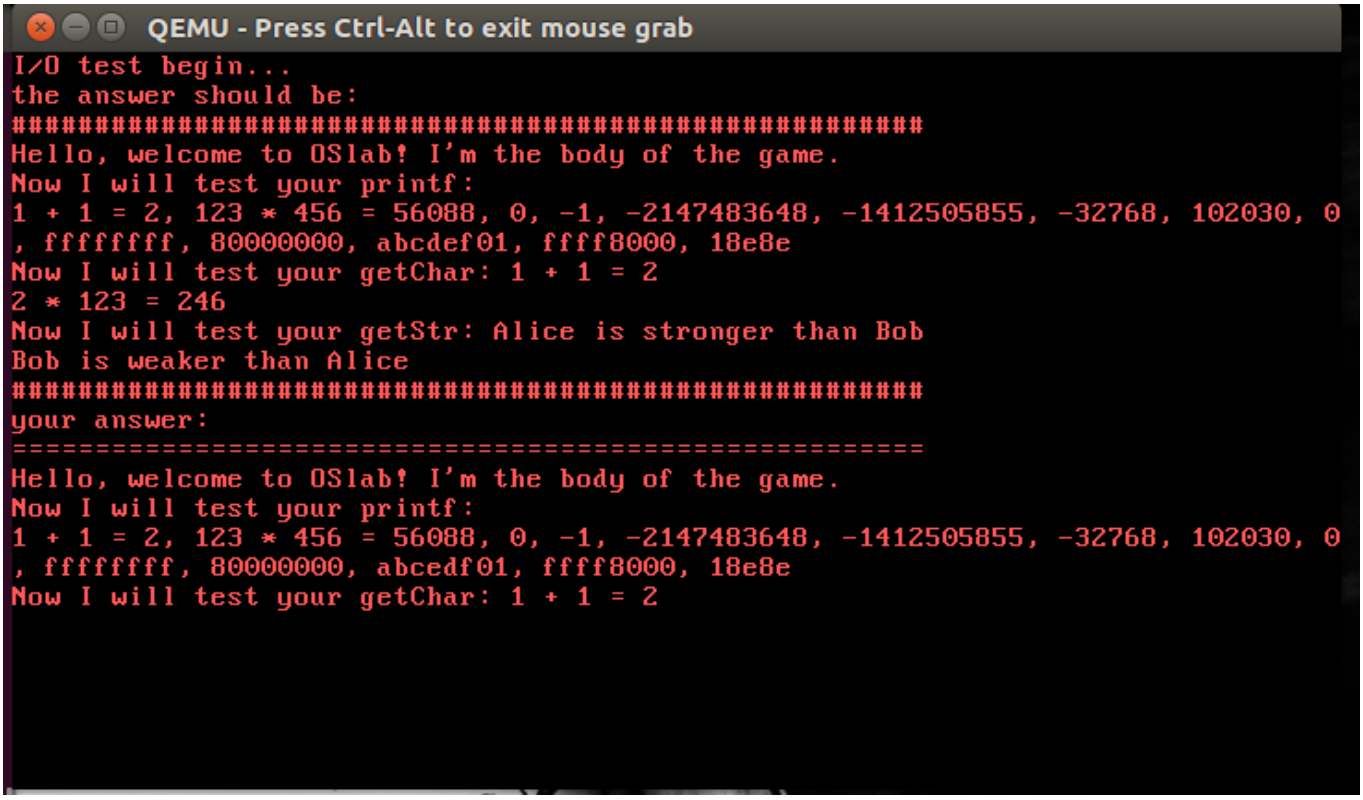
实验目的：

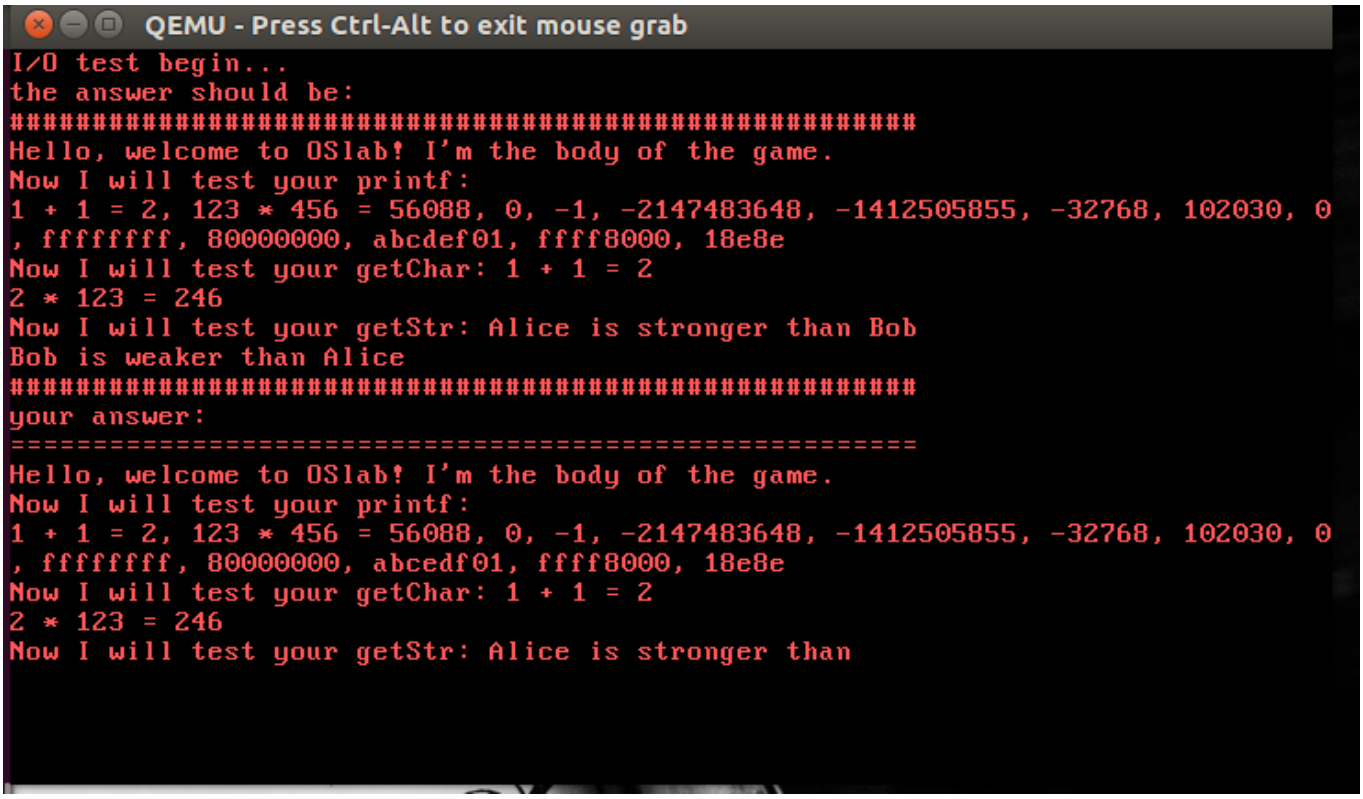
1. 了解中断机制的执行过程；
2. 完成基本I/O操作的实现。

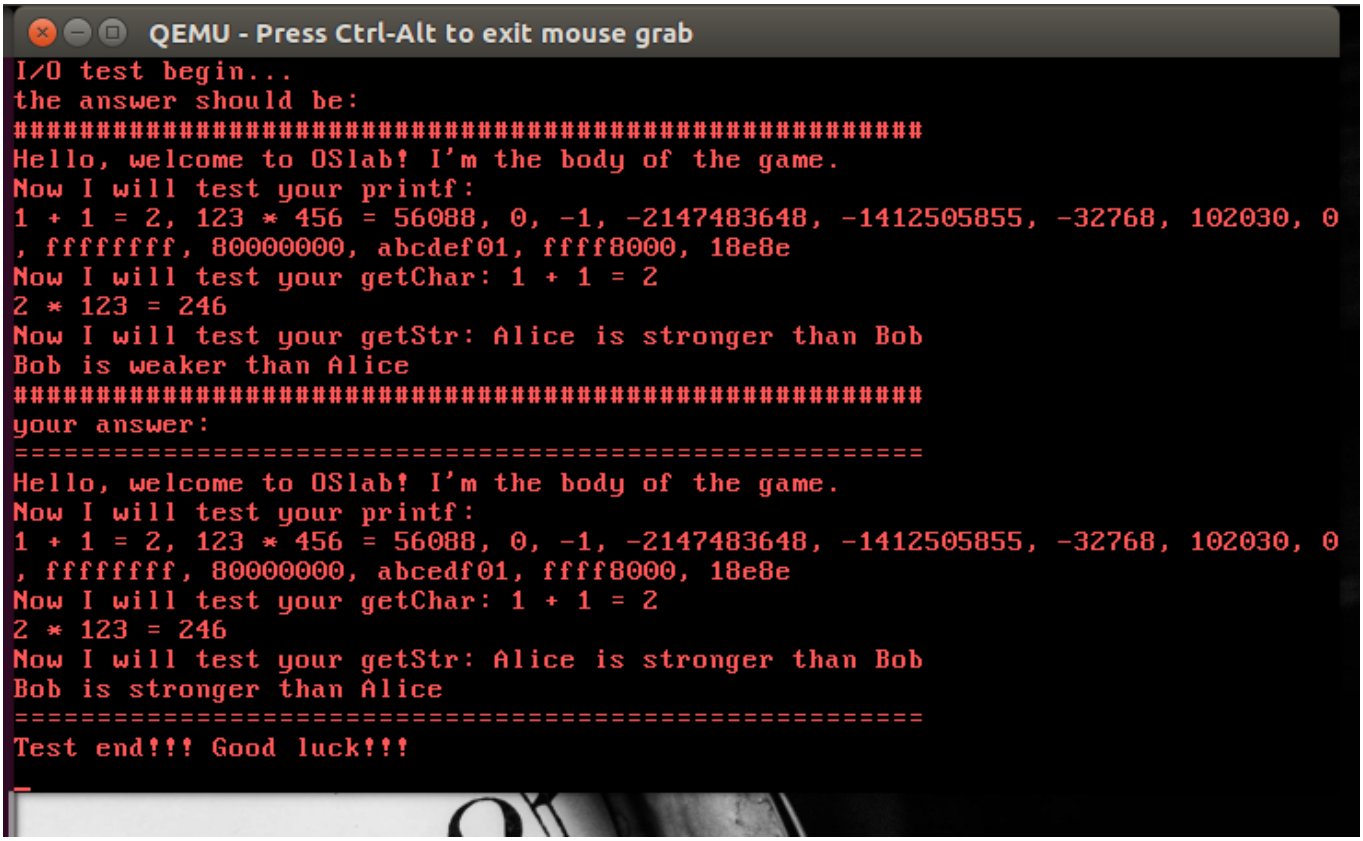
实验结果：

完成了两次装载、一系列的kernel的初始化以及printf、getChar、getStr函数的实现，下面是结果截图：


QEMU
I/O test begin...
the answer should be:
Hello, welcome to OSlab! I'm the body of the game.
Now I will test your printf:
1 + 1 = 2, 123 * 456 = 56088, 0, -1, -2147483648, -1412505855, -32768, 102030, 0
, ffffffff, 80000000, abcdef01, ffff8000, 18e8e
Now I will test your getChar: 1 + 1 = 2
2 * 123 = 246
Now I will test your getStr: Alice is stronger than Bob
Bob is weaker than Alice
your answer:
Hello, welcome to OSlab! I'm the body of the game.
Now I will test your printf:
1 + 1 = 2, 123 * 456 = 56088, 0, -1, -2147483648, -1412505855, -32768, 102030, 0
, ffffffff, 80000000, abcedf01, ffff8000, 18e8e
Now I will test your getChar: 1 + 1 =


QEMU - Press Ctrl-Alt to exit mouse grab
I/O test begin...
the answer should be:
Hello, welcome to OSlab! I'm the body of the game.
Now I will test your printf:
1 + 1 = 2, 123 * 456 = 56088, 0, -1, -2147483648, -1412505855, -32768, 102030, 0
, ffffffff, 80000000, abcdef01, ffff8000, 18e8e
Now I will test your getChar: 1 + 1 = 2
2 * 123 = 246
Now I will test your getStr: Alice is stronger than Bob
Bob is weaker than Alice
your answer:
Hello, welcome to OSlab! I'm the body of the game.
Now I will test your printf:
1 + 1 = 2, 123 * 456 = 56088, 0, -1, -2147483648, -1412505855, -32768, 102030, 0
, ffffffff, 80000000, abcedf01, ffff8000, 18e8e
Now I will test your getChar: 1 + 1 = 2


QEMU - Press Ctrl-Alt to exit mouse grab
I/O test begin...
the answer should be:
##
Hello, welcome to OSlab! I'm the body of the game.
Now I will test your printf:
1 + 1 = 2, 123 * 456 = 56088, 0, -1, -2147483648, -1412505855, -32768, 102030, 0
, ffffffff, 80000000, abcdef01, ffff8000, 18e8e
Now I will test your getChar: 1 + 1 = 2
2 * 123 = 246
Now I will test your getStr: Alice is stronger than Bob
Bob is weaker than Alice
##
your answer:
==
Hello, welcome to OSlab! I'm the body of the game.
Now I will test your printf:
1 + 1 = 2, 123 * 456 = 56088, 0, -1, -2147483648, -1412505855, -32768, 102030, 0
, ffffffff, 80000000, abcedf01, ffff8000, 18e8e
Now I will test your getChar: 1 + 1 = 2
2 * 123 = 246
Now I will test your getStr: Alice is stronger than


QEMU - Press Ctrl-Alt to exit mouse grab
I/O test begin...
the answer should be:
##
Hello, welcome to OSlab! I'm the body of the game.
Now I will test your printf:
1 + 1 = 2, 123 * 456 = 56088, 0, -1, -2147483648, -1412505855, -32768, 102030, 0
, ffffffff, 80000000, abcdef01, ffff8000, 18e8e
Now I will test your getChar: 1 + 1 = 2
2 * 123 = 246
Now I will test your getStr: Alice is stronger than Bob
Bob is weaker than Alice
##
your answer:
==
Hello, welcome to OSlab! I'm the body of the game.
Now I will test your printf:
1 + 1 = 2, 123 * 456 = 56088, 0, -1, -2147483648, -1412505855, -32768, 102030, 0
, ffffffff, 80000000, abcedf01, ffff8000, 18e8e
Now I will test your getChar: 1 + 1 = 2
2 * 123 = 246
Now I will test your getStr: Alice is stronger than Bob
Bob is stronger than Alice
==
Test end!!! Good luck!!!

实验过程:

boot.c:

```
kMainEntry = (void(*)(void))((struct ELFHeader*)elf)->entry;

//phoff = ((struct ELFHeader*)elf)->phoff;

//offset = ((struct ProgramHeader*)(elf + phoff))->off;
```

由于编译过程中删除了elf头文件，所以phoff、offset就不能再按照手册来了。

kvm.c:

代码逻辑与bootload一致，仅仅修改了elf位置与磁盘编号，

idt.c：

此处填写了setIntr和setTrap函数，具体详见代码，按照手册填写。
编写了InitIdt（）函数：

```
setTrap(idt + 0x8, SEG_KCODE, (uint32_t)irqDoubleFault,
DPL_KERN);

    setTrap(idt + 0xa, SEG_KCODE, (uint32_t)irqInvalidTSS,
DPL_KERN);

    setTrap(idt + 0xb, SEG_KCODE, (uint32_t)irqSegNotPresent,
DPL_KERN);

    setTrap(idt + 0xc, SEG_KCODE, (uint32_t)irqStackSegFault,
DPL_KERN);

    setTrap(idt + 0xd, SEG_KCODE, (uint32_t)irqGProtectFault,
DPL_KERN);

    setTrap(idt + 0xe, SEG_KCODE, (uint32_t)irqPageFault,
DPL_KERN);

    setTrap(idt + 0x11, SEG_KCODE, (uint32_t)irqAlignCheck,
```

```
DPL_KERN);

    setTrap(idt + 0x1e, SEG_KCODE, (uint32_t)irqSecException,
DPL_KERN);

    setTrap(idt + 0x21, SEG_KCODE, (uint32_t)irqKeyboard,
DPL_KERN);



    setIntr(idt + 0x80, SEG_KCODE, (uint32_t)irqSyscall,
DPL_USER); //注意此处是DPL_USER
```

doIrq.s：

将irqKeyboard的中断向量号0x21压入栈，即pushl $0x21

irqHandle.c：

1. irqHandle(struct TrapFrame *tf)添加了根据中断向量号选择不同中断服务的代码；
2. KeyboardHandle(struct TrapFrame *tf)函数：增添了处理了正常输入的字符的代码， 将字符打印在显示器上。
3. syscallPrint(struct TrapFrame *tf)函数，则是增添了显示正常字符以及输入\n时的光标移动，以便让printf调用。
4. syscallGetChar函数：在syscall.c文件中是循环调用该函数，直至eax返回结果是1，也就是用户输入完成时。输入字符以及回车后，将keyBuffer[bufferHead++]输出，这里有一个注意点，框架代码中处理键盘输入中的“退格符”时，并未对bufferTail修改，我认为这是不妥的，我增添了bufferTail--的操作，真正实现删除的作用。
5. syscallStr函数：大体部分与getchar相同，但是最先在用户区的堆栈开辟了一定的空间，把字符传至该区域，并且该区域对应str的地址（物理地址）。

kernel/kernel/main.c：

该部分就是把相应初始化函数调用。

```
void kEntry(void) {

    // Interruption is disabled in bootloader

    initSerial();// initialize serial port

    // TODO: 做一系列初始化

    // initialize idt

    initIdt();

    // initialize 8259a

    initIntr();

    // initialize gdt, tss

    initSeg();

    // initialize vga device

    initVga();

    // initialize keyboard device

    initKeyTable();




    loadUMain(); // load user program, enter user space
```

```
    while(1);
    assert(0);
}
```

lib/syscall.c：

1. 该部分getChar和getStr都是循环调用相应的中断处理函数，直至用户输入完成。
2. printf函数，则是根据format[i]是否为“%",再根据format[i+1]对应什么格式符，对应相应的处理函数，最终调用syscallPrint函数。

思考与想法：

1. 框架代码中

```
void KeyboardHandle(struct TrapFrame *tf){
    uint32_t code = getKeyCode();
    if(code == 0xe){ // 退格符
        //要求只能退格用户键盘输入的字符串，且最多退到当行行首
        if(displayCol>0&&displayCol>tail){
            displayCol--;
            bufferTail--;
            uint16_t data = 0 | (0x0c << 8);
            int pos = (80*displayRow+displayCol)*2;
            asm volatile("movw %0, (%1)"::"r"(data),"r"(pos+0xb8000));
        }
```

   输入退格符时，我认为是应该bufferTail--的，不然无法起到删除的作用。
2. 这次实验非常感谢我的室友，我用的ubantu16.04没有对应的VSCode，用vim对于这次实验实在太困难，我的室友花费了一晚上帮我克服了很多问题成功实现主机VSCode与虚拟机ssh连接，提高了生产效率。