南京大学本科生实验报告

课程名称： **操作系统**

学院	计算机科学与技术		
学号	191220029	姓名	傅小龙
Email	1830970417@qq.com		

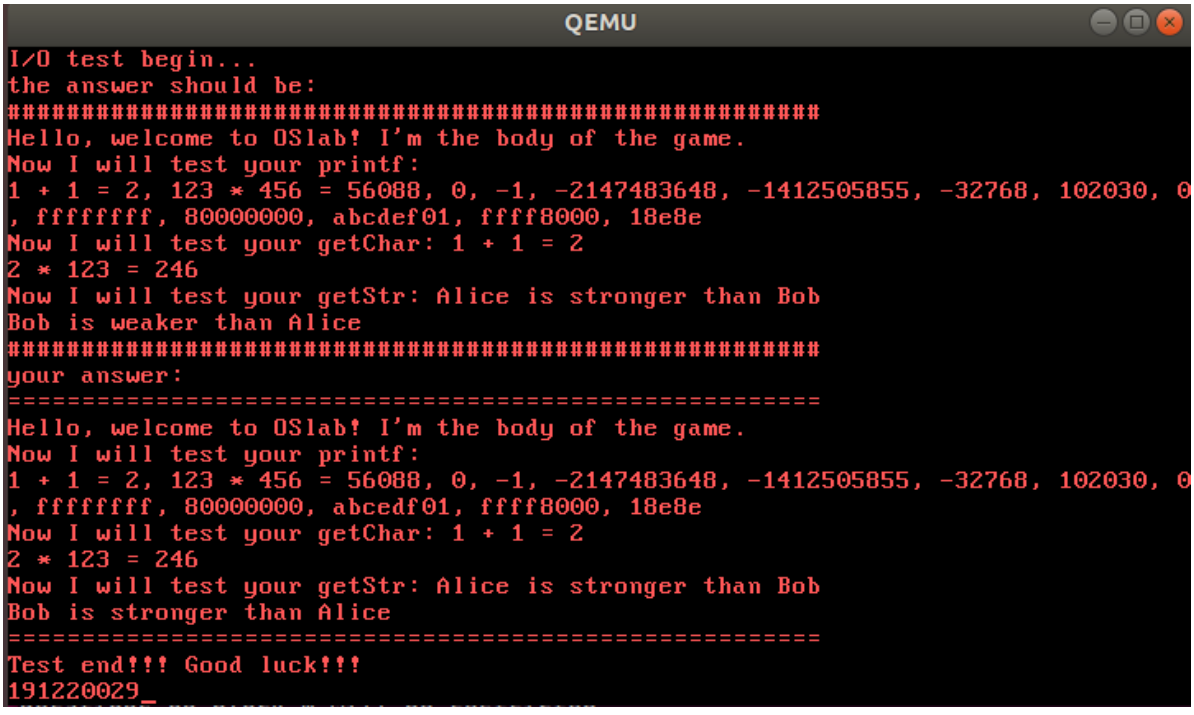
1.实验名称

Lab2: 系统调用

2.实验进度

我完成了所有内容。

3. 实验结果



4.代码修改处

`bootloader/start.S` Line 31 setting esp

`bootloader/boot.c` 填写kMainEntry、phoff、offset

`bootloader/Makefile` 缩小启动盘大小至<510字节

`kernel/main.c` `void kEntry(void)` 做一系列初始化

`kernel/kernel/kvm.c` 参照bootloader加载内核的方式加载用户程序

`kernel/kernel/irqHandle.c``irqHandle(struct TrapFrame *tf)` 填好中断处理程序的调用

`kernel/kernel/irqHandle.c` `void KeyboardHandle(struct TrapFrame *tf)` 处理键盘输入及输入的显示

`kernel/kernel/irqHandle.c` `void syscallPrint(struct TrapFrame *tf)` ①获取用户段选择子;②完成光标的维护和打印到显存

`kernel/kernel/irqHandle.c` `void syscallGetChar(struct TrapFrame *tf)` 通过键盘中断的服务例程将输入的字符返回给调用者

`kernel/kernel/irqHandle.c` `void syscallGetStr(struct TrapFrame *tf)` 通过键盘中断的服务例程将输入的字符串返回给调用者

`kernel/kernel/idt.c` `static void setIntr(struct GateDescriptor *ptr, uint32_t selector, uint32_t offset, uint32_t dpl)` 初始化中断门

`kernel/kernel/idt.c` `static void setTrap(struct GateDescriptor *ptr, uint32_t selector, uint32_t offset, uint32_t dpl)` 初始化陷阱门

`kernel/kernel/idt.c` `void initIdt()` 填入剩下的中断/陷阱表项

`lib/sycall.c``char getChar()` 调用系统调用`SYS_READ`从`STD_IN`读入一个`char`型变量

`lib/syscall.c``void getStr(char *str, int size)` 调用系统调用`SYS_READ`从`STD_IN`读入一个字符串

`lib/syscall.c``void printf(const char *format,...)` 调用系统调用从`SYS_WRITE`从`STD_OUT`写入格式串

`kernel/kernel/doIrq.S` 将`irqKeyboard`的中断向量号压入栈

5. 实验中遇到的问题及思考

5.1 小键盘输入

在测试为`getChar()`提供的服务例程时，按下小键盘数字`2`没有在终端上回显，键盘设备提供的`getChar(uint32_t code)`返回的ascii码为0.

原因：框架提供的键盘设备的码表并没有初始化小键盘上按键对应的ascii码值，而是初值`0`.

解决办法：将框架代码中的码表补上小键盘的部分按键对应的ascii码，如下所示：

```
/*in kernel/kernel/keyboard.c (void) initKeyTable()*/
keyTableL[KKP0_P] = keyTableU[KKP0_P] = '0';
keyTableL[KKP1_P] = keyTableU[KKP1_P] = '1';
keyTableL[KKP2_P] = keyTableU[KKP2_P] = '2';
keyTableL[KKP3_P] = keyTableU[KKP3_P] = '3';
keyTableL[KKP4_P] = keyTableU[KKP4_P] = '4';
keyTableL[KKP5_P] = keyTableU[KKP5_P] = '5';
keyTableL[KKP6_P] = keyTableU[KKP6_P] = '6';
keyTableL[KKP7_P] = keyTableU[KKP7_P] = '7';
keyTableL[KKP8_P] = keyTableU[KKP8_P] = '8';
keyTableL[KKP9_P] = keyTableU[KKP9_P] = '9';
keyTableL[KKPD_P] = keyTableU[KKPD_P] = '.';
keyTableL[KKPM_P] = keyTableU[KKPM_P] = '-';
```

```
keyTableL[KKPS_P] = keyTableU[KKPS_P] = '*';
```

5.2 显存中字符的属性

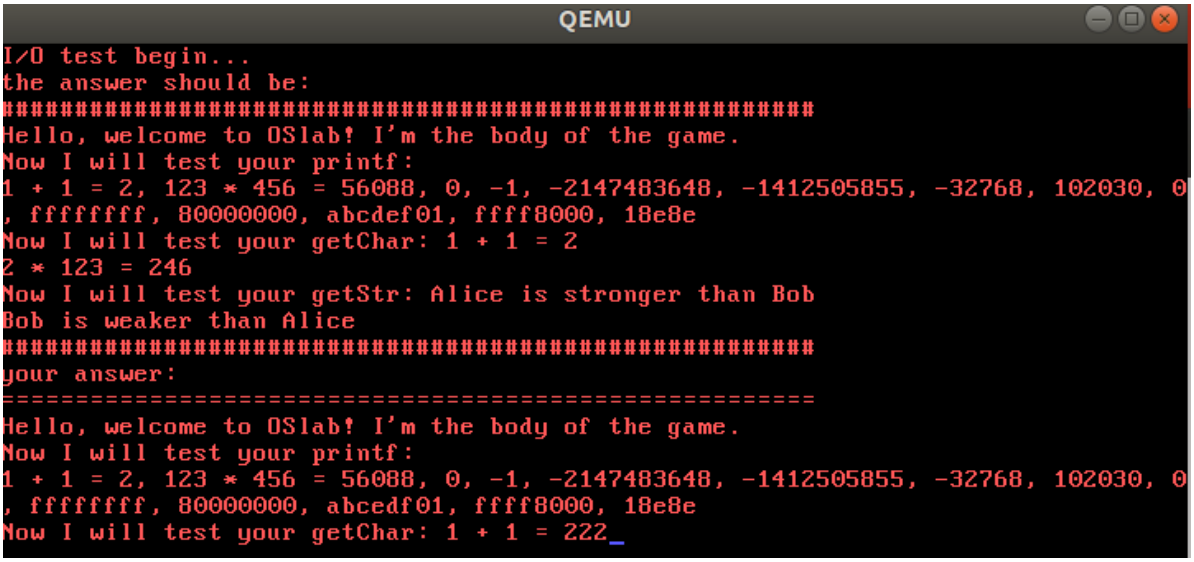
在`Qemu`终端删去输入的字符(键入退格)时，光标不回显。

原因：在键盘服务例程中对退格符的处理中，忘记将写入显存的字符设置属性`0x0c`，即应有`data = getChar(code) | 0x09 << 8;`

通过https://blog.csdn.net/gooding300/article/details/90127513了解到，`data`的高8位为显存中字符的属性值，且各位具有以下含义：

位	7	6	5	4	3	2	1	0
属性	闪烁	背景R	背景G	背景B	高亮	R	G	B

所以上述问题中的光标不回显其实是因为光标为黑色，与黑色的背景颜色相同，所以看不到。尝试把退格后移入显存字符的属性改为`0x09`(黑色背景蓝色字体)，可以看到在键入退格后光标又变成了蓝色：



6.3 Alice & Bob

在测试`getStr()`调用时，发现最后输出的是`Bob is stronger than Alice`，而答案应为`Bob is weaker than Alice`

原因：`app/main.c`提供的测试的打印内容确实是`%s is stronger than Alice`，和答案输出不符。

6. 思考题

6.1 保存寄存器的旧值

> 我们在使用eax，ecx，edx，ebx，esi，edi前将寄存器的值保存到了栈中，如果去掉保存和恢复的步骤，从内核返回之后会不会产生不可恢复的错误？

答：会产生不可恢复的错误。若在内核运行时修改了上述寄存器的值，那么从内核返回后将无法恢复之前用户程序的运行状态。

6.2 ring3的堆栈在哪里?

> IA-32提供了4个特权级，但TSS中只有3个堆栈位置信息，分别用于ring0，ring1，ring2的堆栈切换.为什么TSS中没有ring3的堆栈信息?

答：TSS段用于从低特权级到高特权级空间的切换。从高特权级返回低特权级空间时，低特权级空间的状态已经暂存在了高特权级空间的堆栈中，不需要使用TSS来转换。故TSS不需要保存`ring3`堆栈信息。