LAB2 实验报告

张运吉 (211300063、211300063@ smail.nju.edu.cn)

(南京大学人工智能学院, 南京 210093)

1 实验进度

我已经完成 **LAB2 全部内容。**

2 实验过程

2.1 加载内核到内存某地址并跳转运行

首先使用框架代码提供的 readSect 函数从磁盘中读取内核的数据和代码到内存中以 0x100000 起始的位置，接着根据 elf 文件的格式解析并填写 kMainEntry、phoff、offset。

加载用户程序和上述步骤几乎一样。

2.2 设置esp

同 lab1，设置 esp 为 0x1fffff。

2.3 初始化IDT

首先根据讲义上给的 Interrupt Gate 和 Trap Gate 的结构以及框架代码的数据结构 GateDescriptor 完善 setIntr 和 setTrap 函数，并利用这两个函数来实现 initIdt 函数（初始化 IDT 表, 为中断设置中断处理函数），这里需要注意的就是除了系统调用的 DPL 是用户级别之外，其他的都是内核级别。

2.4 完善键盘中断服务例程

首先，在 IDT 表中加上键盘中断对应的门描述符，然后在 doIrq.S 的 irqKeyboard 函数中将 irqKeyboard 的中断向量号 0x21 压入栈，接着在 irqHandle.c 中填好键盘中断处理程序的调用，最后就是实现核心的键盘中断处理函数 KeyboardHandle，我的实现思路是每次键盘中断到来时，我都将用户按下的键存到一个缓冲区 keyBuffer 中（框架代码提供的，本质是一个数组），然后根据框架代码中 TODO 的要求实现一些特殊按键的处理。

2.5 实现printf的处理例程

同键盘终端一样，先在 IDT 表中加上键盘中断对应的门描述符，因为 printf 属于系统调用，所以填写的是系统调用的门描述符，接着在 irqHandle.c 中填好系统调用中断处理程序的调用，后就是实现核心的写显存相关内容 syscallPrint 函数，这里主要是完成光标的维护和字符输出功能。

2.6 完善printf的格式化输出

我的实现思路是：遍历格式字符串，遇到百分号的时候就根据对应的格式把传入的参数进行转化，结果存在一个字符数组 buffer 里，最后调用 syscall: syscall(SYS_WRITE, STD_OUT, (uint32_t)buffer, (uint32_t)MAX_BUFFER_SIZE, 0, 0)来完成把字符串写入显存

并输出的功能。

2.7 实现getChar, getStr的处理例程

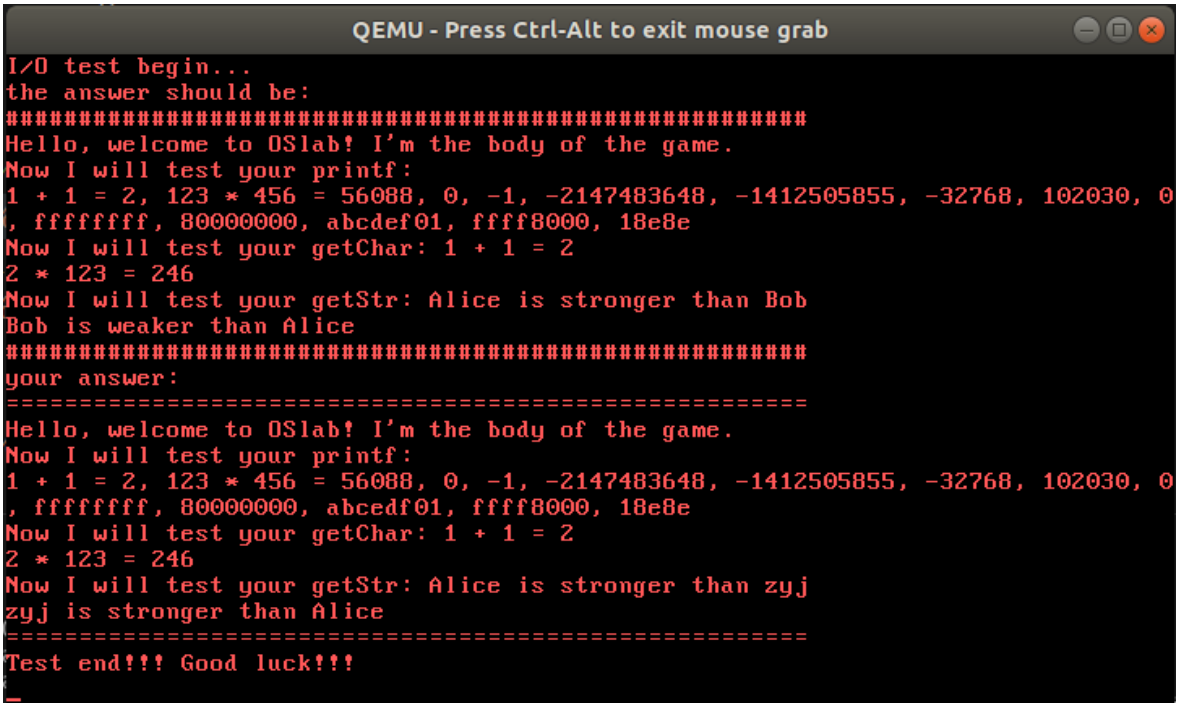
实现 getChar 比较简单，就如讲义上说的：等按键输入完成（遇到回车符）的时候，将末尾字符通过 eax 寄存器传递回来。

实现 getStr 则涉及到内核段和用户段不一致的问题，因为涉及到字符串的传递即使采用了通用寄存器做为中间桥梁，字符串地址作为参数仍然涉及到了不同的数据段。这里我采取的解决方案是先将数据段寄存器指向用户数据段选择子所描述的数据段，这样就可以在内核对用户数据段进行访问。是通过以下代码实现的：

```
int usr_data = USEL(SEG_UDATA);
asm volatile("movw %0, %%es"::"m"(usr_data));
```

3 实验结果

正确通过样例:



4 思考题

4.1 Ex1: 计算机系统的中断机制在内核处理硬件外设的 I/O 这一过程中发挥了什么作用？

I/O 设备需要进行数据交互的时候，它会向 CPU 发送中断信号，CPU 在收到中断信号之后会暂停当前任务的处理进入中断处理程序。中断机制的运用使得计算机避免了在运行过程当中不断轮询 I/O 设备检查是否有数据传输，减少了 CPU 时间的浪费， 提高了计算机执行的效率。

4.2 Ex2: 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?

当堆栈发生切换时，新的 ss 和 esp 从 TSS 取得的，比如，我们当前所在的是 ring3，当转移至 ring1 时，堆栈将被自动切换到由 ss1 和 esp1 指定的位置。由于只是在由外层转移到内层（低特权级到高特权级）切换时新堆栈才会从 TSS 中取得，所以 TSS 中没有位于最外层的 ring3 的堆栈信息。

4.3 Ex3: 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?

去掉保存和恢复的步骤在返回后可能会产生不可恢复的错误，因为在用户程序可能使用这些寄存器存储了一些关键信息，如果在进入内核前不压栈保存，在内核处理的过程中可能会改变了这些寄存器的值导致数据丢失，从而使用户程序运行发生错误。

4.4 Ex4: 查阅相关资料，简要说明一下%d , %x , %s , %c 四种格式转换说明符的含义。

%d:十进制整数，%x:十六进制整数，%s:字符 串，%c:一个字符。

5 收获与感想

通过本次实验，我加深了基于中断实现系统调用全过程的理解，计算机系统的中断机制在内核处理硬件外设的 I/O 发挥的重要作用，本次实验相比 lab1 代码量有了很大提升，我花费了很长时间才了解了框架代码的大致结构，自己动手写代码也是 bug 不断。

总之，本次实验加深了我对系统调用和中断机制的了解，也锻炼了我的代码能力。