PA4实验报告

**161220017 陈翔**

PA4.1

§4-1.3.1 通过自陷实现系统调用

1. 详细描述从测试用例中的 int $0x80 开始一直到 HIT\_GOOD\_TRAP 为止的详细的系统行为（完整描述控制的转移过程，即相关函数的调用和关键参数传递过程）， 可以通过文字或画图的方式来完成。

答：首先，测试用例中出现int指令，系统通过cpu轮询找到并调用相应的int函数，并将读到的8位立即数“0x80”传给raise\_sw\_intr函数并调用之。接着，raise\_sw\_intr函数先将pc修改为下条指令的位置，再将刚刚传入的0x80传给raise\_intr函数并调用之。然后，在raise\_intr函数中，一系列参数进栈，并将eip值修改，进入内核态，调用vecsys()系统函数，接着该函数调用asm\_do\_irq函数（为了执行pusha指令），asm\_do\_irq接着调用irq\_handle函数,接收的形参是Trapframe结构的首地址（指针），通过将esp压栈得以保存，然后通过该结构调用sys\_call函数，sys\_call函数又通过该结构中eax的值调用sys\_write函数，sys\_write函数调用fs\_write函数，fs\_wirte函数调用0x82指令，输出“Hello, world！”字样，然后返回用户态，输出“HIT\_GOOD\_TRAP”。

Kernel调用相关函数

用户进程

陷阱指令

通过eax调用sys\_write函数

返回到陷阱指令下一条执行

int $0x80

1. 在描述过程中， 回答 kernel/src/irq/do\_irq.S 中的 push %esp 起什么作用， 画出在 call irq\_handle 之前，系统栈的内容和 esp 的位置， 指出 TrapFrame 对应系统栈的哪一段内容。

答：push %esp的作用为把Trapframe结构的首地址压栈，为了方便irq\_handle函数调用结构的指针参数（即首地址）。

|  |  |
| --- | --- |
| 返回地址 |  |
| %ebp旧值 | 🡨%ebp |
| eflags |  |
| cs |  |
| eip（返回地址值） |  |
| error\_code |  |
| irq |  |
| %eax |  |
| %ecx | TrapFrame |
| %edx |  |
| %ebx |  |
| esp的旧值（无用） |  |
| %ebp |  |
| %esi |  |
| %edi |  |
| push%edi后的esp值 | 🡨%esp |

§4-1.3.2 响应时钟中断通过自陷实现系统调用

1. 详细描述 NEMU 和 Kernel 响应时钟中断的过程和先前的系统调用过程不同之处在哪里？相同的地方又在哪里？可以通过文字或画图的方式来完成。

答：NEMU：在开启时钟中断之后，NEMU会初始化cpu的中断引脚，并且会在每次执行指令后根据中断引脚和IF的情况判断要不要进行时钟中断处理，这个过程是通过将时钟中断号传给上一个任务中的raise\_intr()函数实现的。其余部分相同。Kernel：在开启时钟中断后，Kernel每10ms就会调用一次irq\_handle()函数，用来处理相应的时钟中断事件，而irq\_handle()函数的触发过程和上一个任务中通过自陷指令触发的类似。故两者结合起来的不同之处为：1.时钟中断需要开启线程以便对调用号和中断允许进行初始化，后者直接在指令中进行；2.系统调用号不同，前者是0x80，后者是0x20(0+32)；3.引发条件不一样，前者是自陷指令引起的系统调用（不可屏蔽），后者是每十毫秒调用一次的时钟中断函数引起的系统调用。相同之处：1.两者都需要进行用户态到内核态的准备工作，即把用户进程的相关数据放入用户栈中；2.两者都会调用irq\_handle函数。

PA4.2

1. 注册监听键盘事件是怎么完成的？

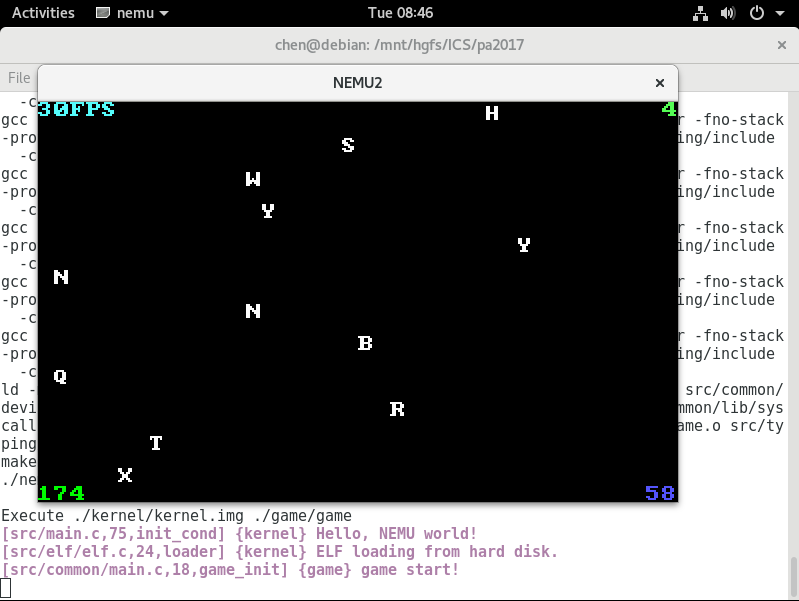
答：echo中的main函数首先调用了一条增加请求中断句柄的函数，该函数执行了陷阱指令，并给各个寄存器赋值，eax=0代表调用增加请求中断句柄，ebx为调用句柄号，ecx为相应的处理函数指针，这里为键盘事件处理函数。该处理函数会在相应的时刻读取键盘扫描码，然后交给扫描码翻译程序翻译，再将翻译出来的字符打印在屏幕上。main函数中一直循环执行的hlt指令是为了让cpu处于休眠，直到下一个键盘中断信号的到来。

1. 从键盘按下一个键到控制台输出对应的字符，系统的执行过程是什么？如果涉及与之前报告重复的内容， 简单引用之前的内容即可。

答：键盘按下一个键后，hlt指令监测到新的键盘中断信号（由cpu的中断引脚发出），引发硬件中断，并将该键盘扫描码交给键盘事件处理函数处理，然后键盘事件处理函数通过扫描码翻译程序翻译出扫描码对应的大写英文字符，并通过封装过的printc函数打印到屏幕上去。

PA4.3

1. 打字小游戏：



1. 仙剑奇侠传：

