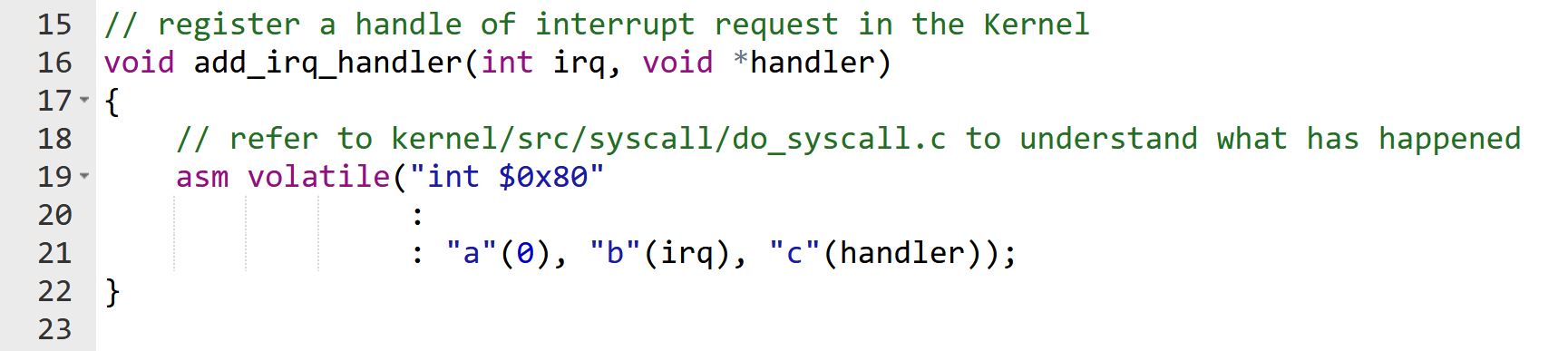
针对echo测试用例，在实验报告中，结合代码详细描述：

1. 注册监听键盘事件是怎么完成的？

注册监听键盘事件的测试样例是echo，即观察echo.c中代码：

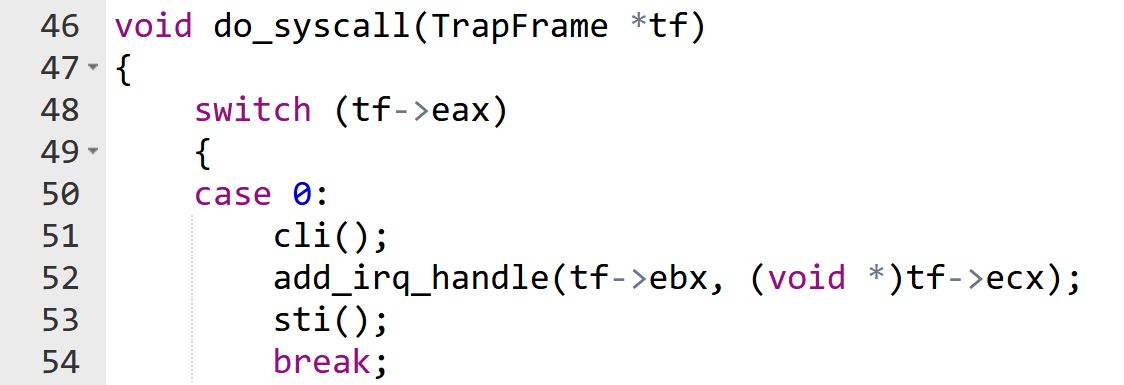


调用了add\_irq\_handler，其实现如下：

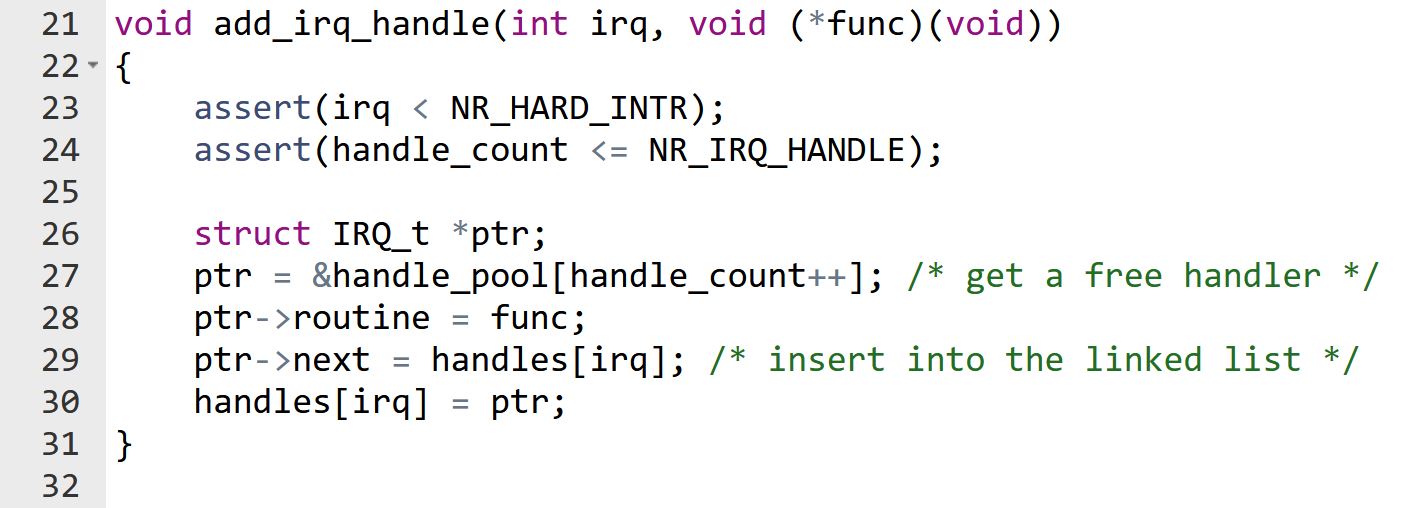


可见该函数的功能为注册监听事件，即将handler与中断异常处理程序绑定

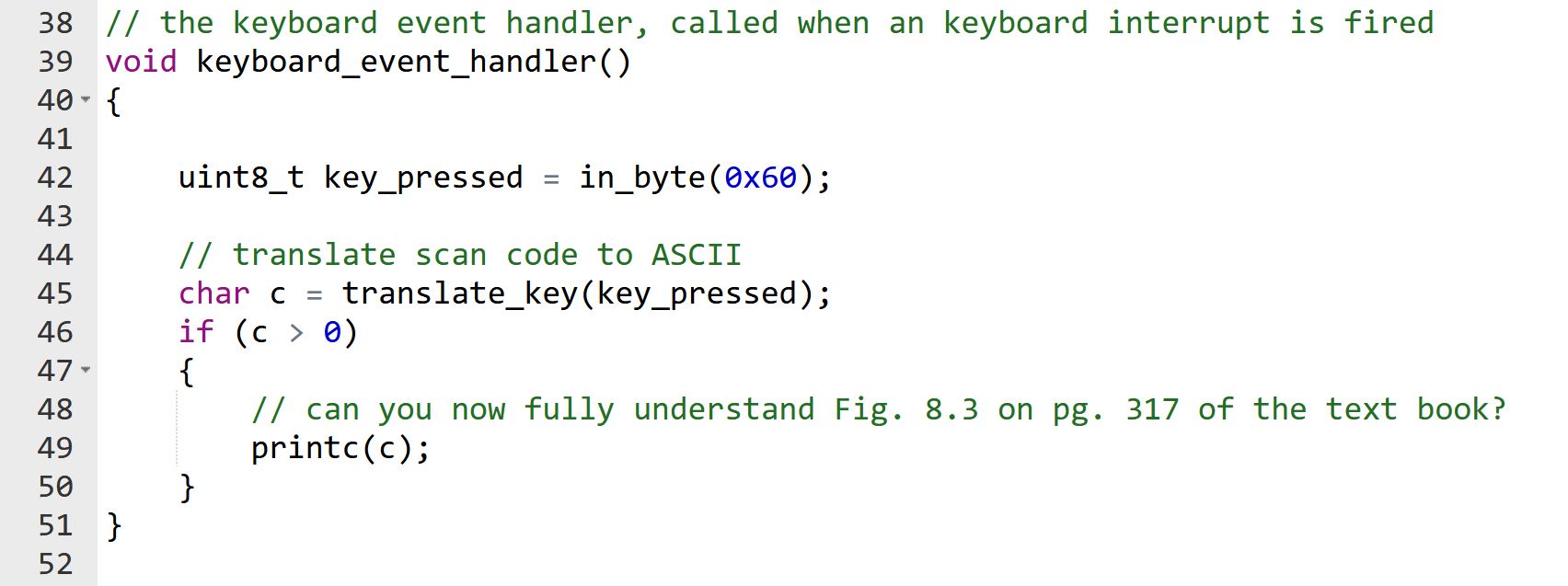
根据int $0x80可知为系统调用，又此时eax为0，找到do\_syscall代码：



其中调用了add\_irq\_handle，找到这个函数的定义：

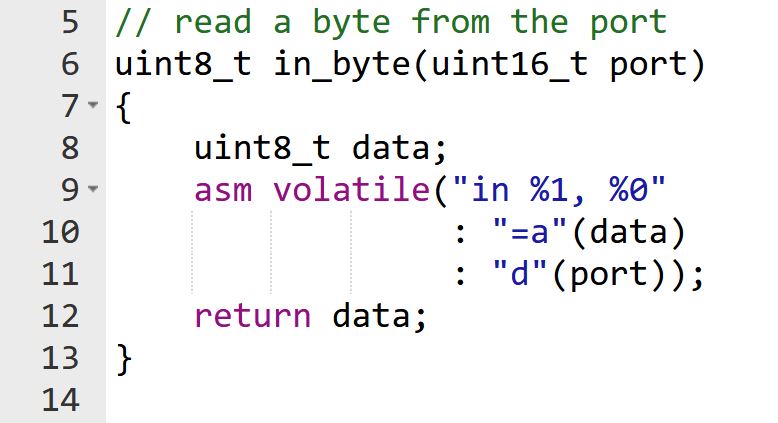


在handle\_pool里找到空的位置插入，此处func为keyboard\_event\_handler：

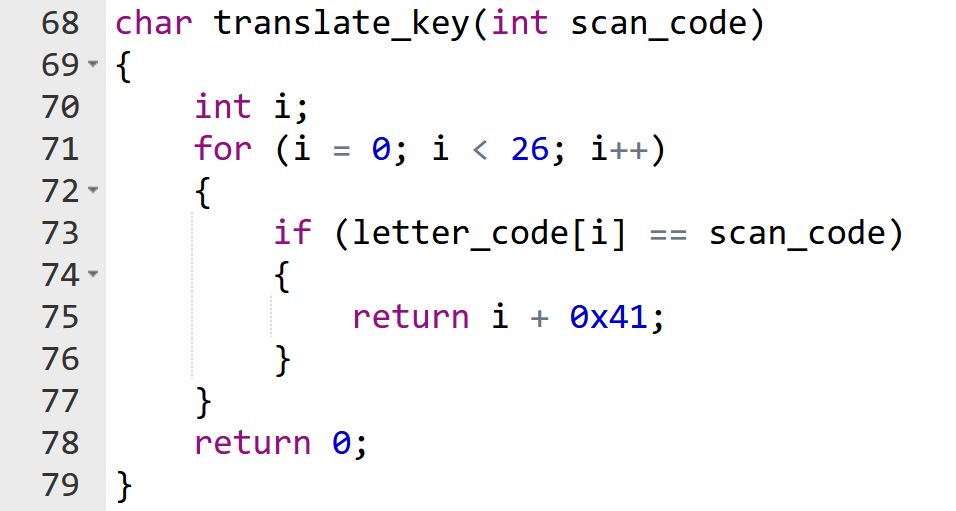


可见keyboard\_event\_handler调用in\_byte来获取按键输入的键码

然后通过translate\_key来将输入的键码“翻译”为ascii码：



观察in\_byte可见它的功能是从端口读入一个字节的数据

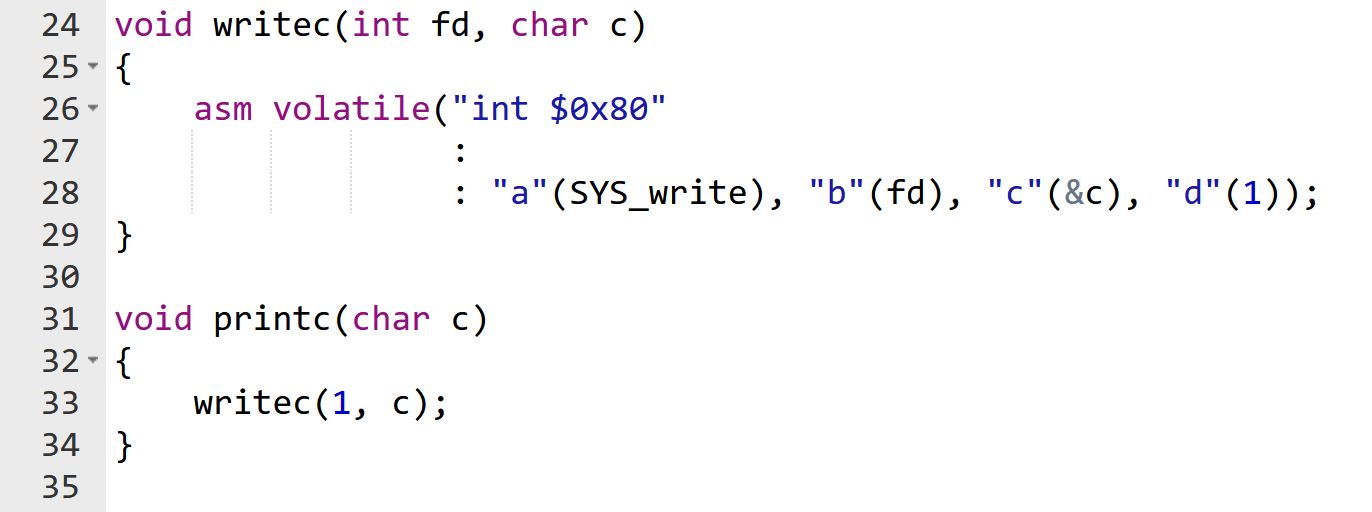


可见translate\_key的作用是依次检查键盘输入是否在26个字母之间

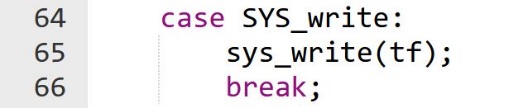
若在26个字母中，则返回对应字母的ascii码，否则返回值为0

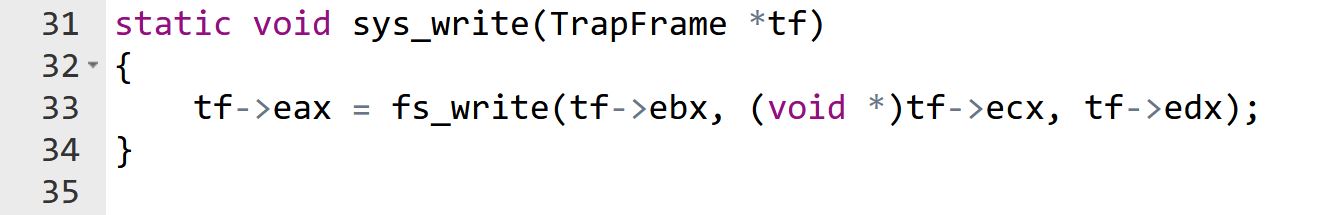
2. 从键盘按下一个键到控制台输出对应的字符，系统的执行过程是什么？

按下键盘出现异常，系统调用函数与keyboard\_event\_handler绑定，其中调用in\_byte，该函数调用汇编指令in，in指令调用了pio\_read，读取的结果返回值储存在data中，继续调用translate\_key转化成ascii码，通过printc调用调用writec输出字符：

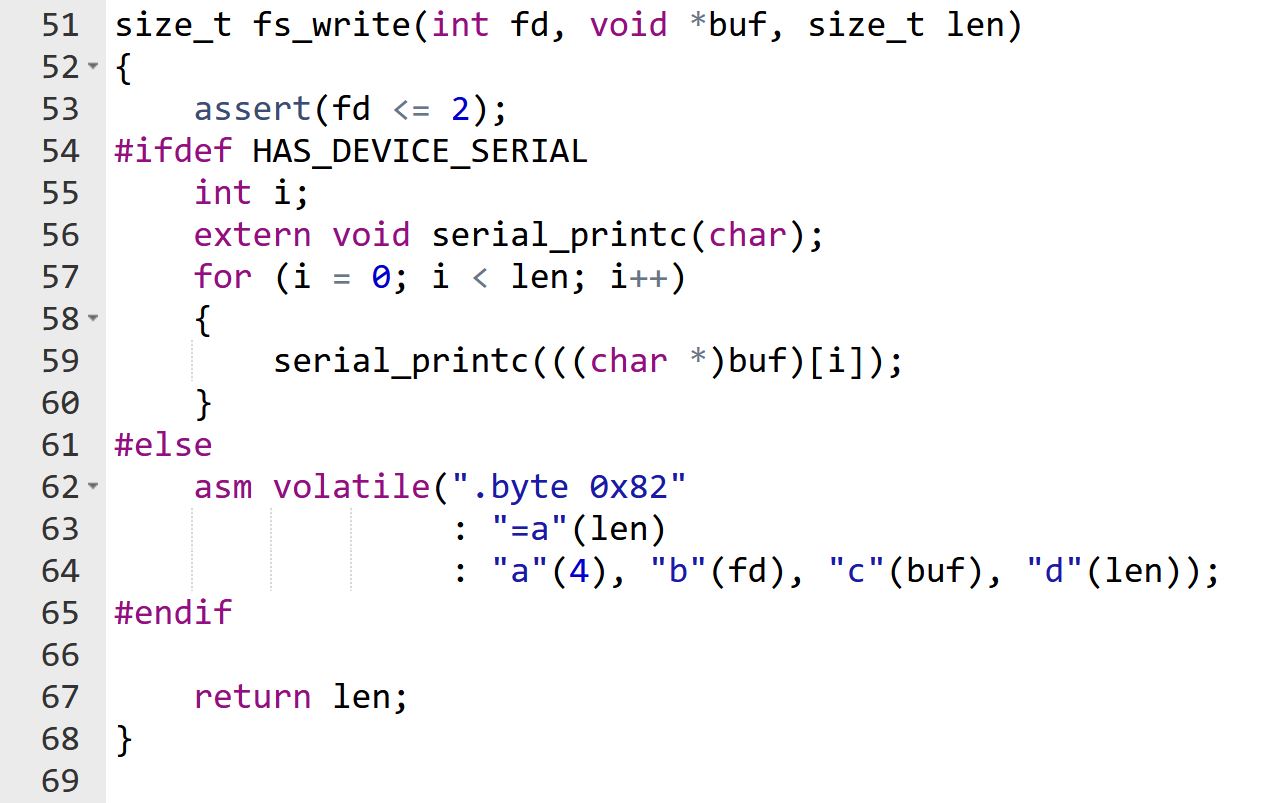


可见这里又有一个系统调用，经do\_syscall调用sys\_write：

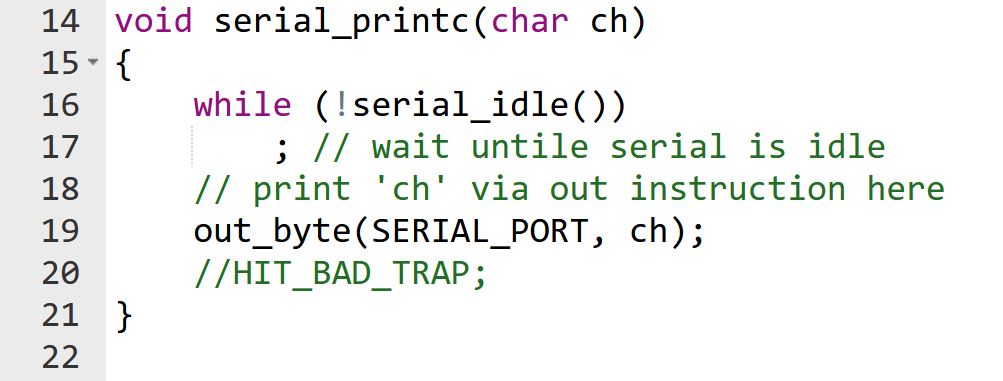


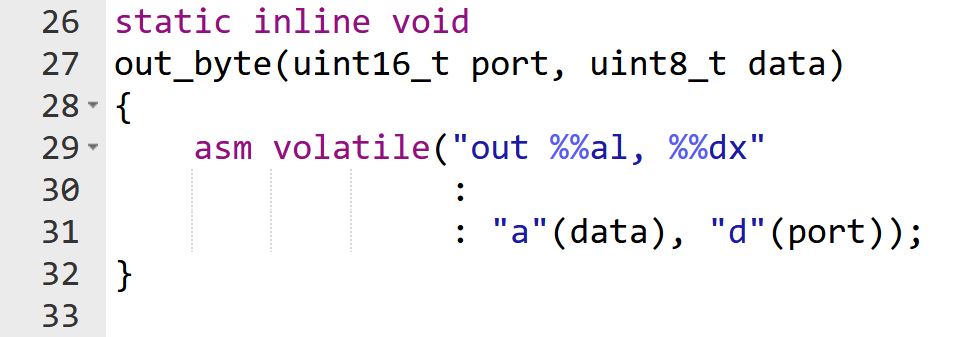


然后sys\_write调用fs\_write，传入输入的字符指针和长度：



可见调用serial\_printc，每次输出一个字符：





其中又调用了out\_byte，调用汇编指令out，利用pio\_write实现输出