PA_4-2 实验报告

计算机科学与技术系 张桓 191220156

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如何完成的？

答：执行 echo 测试用例时，观察其代码：

```
int main()
{
    // register for keyboard events
    add_irq_handler(1, keyboard_event_handler);
    while (1)
        asm volatile("hlt");
    return 0;
}
void add_irq_handler(int irq, void *handler)
{
    // refer to kernel/src/syscall/do_syscall.c to understand what has happened
    asm volatile("int $0x80"
                 :
                 : "a"(0), "b"(irq), "c"(handler));
}
```

通过 add_irq_handler()函数，这通过 int $0x80 指令不断调用一直到 do_syscall()函数，通过 switch 执行

```
case 0:
        cli();
        add_irq_handle(tf->ebx, (void *)tf->ecx);
        sti();
        break;
```

关中断，并且通过 add_irq_handle()函数为 1 号中断注册 keyboard_event_handler()处理函数。

在 nemu/src/device/sdl.c 中通过 SDL 开发包注册了对于键盘物理事件的监听，由 NEMU_SDL_Thread()线程捕获键盘按下和抬起两个事件。再调用 nemu/src/device/dev/keyboard.c 中的 do_keyboard()键盘模拟

函数，该函数判断物理事件的类型，即键盘按下或抬起分别调用 keyboard_down()函数和 keyboard_up()函数。其参数为按下物理按键的编码。代码如下：

```
else if (e.type == SDL_KEYDOWN)
{
    keyboard_down(e.key.keysym.sym);
}
else if (e.type == SDL_KEYUP)
{
    keyboard_up(e.key.keysym.sym);
}
```

函数定义如下：

```
// called by do_keyboard() on detecting a key down event
void keyboard_down(uint32_t sym)
{
    // put the scan code into the buffer
    scan_code_buf = sym2scancode[sym >> 8][sym & 0xff];
    // issue an iterrupt
    i8259_raise_intr(KEYBOARD_IRQ);
    // maybe the kernel will be interested and come to read on the data port
}

// called by do_keyboard() on detecting a key up event
void keyboard_up(uint32_t sym)
{
    // put the scan code into the buffer
    scan_code_buf = sym2scancode[sym >> 8][sym & 0xff] | 0x80;
    // issue an iterrupt
    i8259_raise_intr(KEYBOARD_IRQ);
    // maybe the kernel will be interested and come to read on the data port
}
```

可以看到函数先将按键的编码转换成扫描码并存入 scan_code_buf 中，然后调用 i8259_raise_intr()产生中断。

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？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：按下按键后 NEMU 检测到该物理事件，依次调用 do_keyboard()函数和 keyboard_down()函数，将按键编码转成标准扫描码缓存在本地变量 scan_code_buf 中，再调用 i8259_raise_intr(KEYBOARD_IRQ)，其中 KEYBOARD_IRQ=1.从而产生 1 号中断。(详见上一题)

在 nemu/src/device/i8259_pic.c 中找到对应代码如下：

```
void i8259_raise_intr(uint8_t irq_no)
{
    // this is the only place i8259_intr_no and cpu.intr is set
    while (SDL_LockMutex(i8259_mutex) != 0)
    {
        if (nemu_state == NEMU_STOP)
            return;
        SDL_Delay(1);
    }
    // the i8259_mutex is to protect the i8259_intr_no and cpu.intr
    if (i8259_intr_no == I8259_NO_INTR ||
        (i8259_intr_no != I8259_NO_INTR && irq_no > i8259_intr_no - IRQ_BASE))
    {
        // priority
        i8259_intr_no = irq_no + IRQ_BASE;
        //printf("i8259 raise %d, %d\n", irq_no, i8259_intr_no);
    }
#ifdef IA32_INTR
    cpu.intr = 1;
#endif
    SDL_UnlockMutex(i8259_mutex); // unlock
}
```

可以看到中断号 i8259_intr_no = irq_no + IRQ_BASE，其中 IRQ_BASE=32，而传入的参数 irq_no=1，因此中断号为 33。再将中断引脚置 1，这样在 nemu/src/cpu/cpu.c 中的 do_intr()函数中如果开中断的话就会进行中断执行。代码如下：

```
void do_intr()
{
    if (cpu.intr && cpu.eflags.IF)
    {
        is_nemu_hlt = false;
        // get interrupt number
        uint8_t intr_no = i8259_query_intr_no(); // get interrupt number
        assert(intr_no != I8259_NO_INTR);
        i8259_ack_intr();  // tell the PIC interrupt info received
        raise_intr(intr_no); // raise intrrupt to turn into kernel handler
    }
}
```

这里拿到的中断号 intr_no 就是 33，再 raise_intr(intr_no)，同样查询 IDT，在 kernel/src/irq/idt.c 中找到对应中断门描述符如下:

```
set_intr(idt + 32 + 1, SEG_KERNEL_CODE << 3, (uint32_t)irq1, DPL_KERNEL);
// keyboard
```

对应入口地址为 irq1，置 CS:EIP 为这个地址，从而开始执行该地址处的代码。在 kernel/src/irq/do_irq.S 中找到其代码如下:

```
.globl irq1;     irq1:  pushl $0;  pushl $1001; jmp asm_do_irq
```

将 error_code 和 irq 号(1001)压入内核栈中，再跳转到 asm_do_irq 处，代码如下:

```
asm_do_irq:
    pushal
    pushl %esp      # ???
    call irq_handle
    addl $4, %esp
    popal
    addl $8, %esp
    iret
```

先 pusha 将所有通用寄存器的值(GPRS)压栈，再压入 ESP，然后调用 irq_handle 中断/异常处理程序。找到 irq_handle 函数对应代码:

```
void irq_handle(TrapFrame *tf)
{
    int irq = tf->irq;
    /***/
    else if (irq >= 1000)
```

```
    {
        int irq_id = irq - 1000;
        assert(irq_id < NR_HARD_INTR);
        struct IRQ_t *f = handles[irq_id];
        while (f != NULL)
        { /* call handlers one by one */
            f->routine();
            f = f->next;
        }
    }
}
```

irq 参数为 1001>=1000，得到 irq_id = 1，即键盘中断号。然后拿到 handlers[]数组中的 1 号元素存入 f，这是个指向一个 IRQ_t 类型链表的指针，而这个链表上只有一个结点其 routine()为 keyboard_event_handler()函数，这是在 echo 测试用例中增加的(详见上一题)。

而 keyboard_event_handler()函数代码：

```
// the keyboard event handler, called when an keyboard interrupt is fired
void keyboard_event_handler()
{

    uint8_t key_pressed = in_byte(0x60);

    // translate scan code to ASCII
    char c = translate_key(key_pressed);
    if (c > 0)
    {
        // can you now fully understand Fig. 8.3 on pg. 317 of the text book?
        printc(c);
    }
}
```

通过 in 指令传入端口号并从该端口读一个字节出来放入 key_pressed，即之前存入的 scan_code_buf 键盘扫描码。端口号为 0x60 即键盘数据端口，从而实现和 echo 用户程序的数据交换。

再将扫描码翻译成 ASCII 码放入 c 中，再用 printfc()通过 int 指令过

串口将 c 打印到屏幕上，整个过程就结束了。