JOS-Lab-6 实验报告

熊伟伦
5120379076
azardf4yy@gmail.com

2014年12月12日-12月15日

Contents

1 前言

该报告描述了我在lab6实验的过程中遇到的问题与解决的方法，介绍了lab6的整体结构。指导中问题的解答参考上传的压缩包中的answers-lab6.txt文件

2 Part A: Initialization and transmitting packets

又要换一次QEMU版本，不忍吐槽，为何不所有lab直接都用MIT最新的JOS lab，这样就不用QEMU各个版本装来装去了。

在grade测试的input中，我也不知道为什么测试脚本会读我gcc的编译信息，因此这部分测试通不过，调了一段时间没搞定，后来我就我没管它。其他测试均能通过。

2.1 Exercise-1

这里比较简单，现在kern/trap.c的switch里的IRQ_OFFSET+IRQ_TIMER的块里调用time_tick()，然后在sys_call里分别注册一下SYS_time_msec和调用time_msec()。然后就通过了user/testtime.c里的sleep()操作。

打INIT_CFLAGS=-DTEST_NO_NS太麻烦，暂时先把init里的ns的ENV注释掉了。

2.2 Exercise-2

接下来查看Intel，在必要的时候参考一下。了解什么是PCI，接下来的exercise都和PCI的启动和初始化有关。

2.3 Exercise-3

先查看手册5.2的表5-1，82540EM-A桌面版Vender ID 8086h， Device ID 100E。

按照说明描述E1000的Array要放在0,0,0之前。

kern/pci.c

```
struct pci_driver pci_attach_vendor[] = {
    { E1000_VENDOR, E1000_DEVICE, &e1000_attach},
    { 0, 0, 0 },
};
```

然后再在e1000.h里加入对应的函数声明，在e1000.c中添加如下函数。

kern/e1000.c

```
int
e1000_attach(struct pci_func* pcif)
{
```

```
4     pci_func_enable(pcif);
5     return 0;
6 }
```

这里我make grade发现gradelib.py有问题，419行用的tab其他都用的空格，不能跑，所以我把这一行前面的缩进换成了tab。然后还改killall qemu-system-i386改成了killall qemu。

make grade的testtime和pci attach都OK了。

2.4 Exercise-4

这里我先输出了base[0]和size[0]，分别是0xfebc0000和20000。

需要把物理内存这部分映射到虚拟内存中的块，这里我就按资料说的直接映射到KSTACKTOP。status寄存器查看文档的13.4.2节说是00008h只读。

代码如下：

kern/e1000.h

```
1 #define E1000_ADDR KSTACKTOP
2 #define E1000_STATUS 0x00008/4
3 volatile uint32_t* e1000;
```

kern/e1000.c

```
1 boot_map_region(kern_pgdir, E1000_ADDR, pcif->reg_size[0],
2 pcif->reg_base[0], PTE_W | PTE_PCD | PTE_PWT);
3 e1000 = (uint32_t*)E1000_ADDR;
4 cprintf("E1000 status: %08x\n", e1000[E1000_STATUS]);
```

输出确实是0x80080783

2.5 Exercise-5

这里我遇到了迷之bug，后来发现是struct padding的问题，开始没仔细看材料。

按照材料逐步实现init，make E1000_DEBUG=TXERR,TX qemu有输出说明init正确。

2.6 Exercise-6和Exercise-7

添加syscall很麻烦，总是忘记某个地方。一共有inc/lib.h，inc/syscall.h，lib/syscall.c，kern/syscall.c，还有inc/error.h。

装了tcpdump，还需要pixman，最后pcap可以解析有输出。

2.7 Exercise-8

在output里call一下之前写的sys_net_transmit，能pass make grade的input。

3 Part B: Receiving packets and the web server

3.1 Exercise-9 12

这里要写的部分是input和receive，和之前的transmit和output有部分雷同，实现的功能不一样。

这里的make grade读了我gcc的信息会有问题，我不知道如何解决，就没管它了。

3.2 Exercise-13

这部分比较简单，在send_data和send_file里call一下给的接口就行。

3.3 Challenge

Challenge部分我实现了load MAC的challenge，因为比其他的challenge简单。

加了一个syscall，修改了inc/lib.h，inc/syscall.h，lib/syscall.c，kern/syscall.c。详细的代码其实就2行，在sysycall.h中，返回e1000结构中的2个内容就行。

kern/e1000.c

```
1 static int
2 sys_get_mac(uint32_t* low, uint32_t* high)
3 {
4     *low = e1000[E1000_RAL];
5     *high = e1000[E1000_RAH] & 0xffff;
6     return 0;
7 }
```

经测试，修改模拟的E1000硬编码写的MAC地址，输出会改变。