

一、实验内容

1.1 e1000网卡驱动

1.1.1 实验步骤

- 对于e1000_transmit
 - 首先获取尾指针，判断尾指针对应的描述符的状态是否为E1000_TXD_STAT_DD，如果该位置未被设置说明该部分数据仍未被传输完成，缓冲区已满，无法继续写入
 - 若可以继续写入，则在缓冲区队列中对应的数据帧未释放时进行释放
 - 将描述符对应的地址指向数据帧头部，并设置相应的length、cmd等字段，将数据帧记录到对应的缓冲区队列位置
 - 最后更新尾指针寄存器对应的值

```
int
e1000_transmit(struct mbuf *m)
{
    //
    // Your code here.
    //
    // the mbuf contains an ethernet frame; program it into
    // the TX descriptor ring so that the e1000 sends it. Stash
    // a pointer so that it can be freed after sending.
    //
    uint32 tail;
    struct tx_desc *desc;

    acquire(&e1000_lock);
    tail = regs[E1000_TDT];
    desc = &tx_ring[tail];

    if (!(desc->status & E1000_TXD_STAT_DD)) {
        release(&e1000_lock);
        return -1;
    }

    if(tx_mbufs[tail]){
        mbuf_free(tx_mbufs[tail]);
    }

    desc->addr = (uint64) m->head;
    desc->length = m->len;
```

```

desc->cmd = E1000_TXD_CMD_RS | E1000_TXD_CMD_EOP;
tx_mbufs[tail] = m;

regs[E1000_TDT] = (tail + 1) % TX_RING_SIZE;
release(&e1000_lock);

return 0;
}

```

- 对于e1000_recv

- 由于网卡硬件处理完毕一个数据帧后，头指针会先前移，并将数据帧放置到头指针对应的位置，所以在该部分，要处理最早接收到的数据帧应该从尾指针的下一个位置对应的缓冲区开始处理
- 在获取到最早接收的数据帧位置对应的描述符后，通过状态是否为E1000_RXD_STAT_DD，若否则说明网卡硬件还未处理完毕
- 若是则获取对应的数据帧，设置对应的length字段后交付给net_rx进行下一步处理，并分配一个新的缓冲区，并设置对应的描述符的相关字段
- 更新尾指针到下一个位置，尝试该位置是否能进行处理，若能则重新回到第2步，否则将尾指针退回一个位置

```

static void
e1000_recv(void)
{
    //
    // Your code here.
    //
    // Check for packets that have arrived from the e1000
    // Create and deliver an mbuf for each packet (using net_rx()).
    //
    uint32 tail = (regs[E1000_RDT] + 1) % RX_RING_SIZE;
    struct rx_desc *desc = &rx_ring[tail];

    while((desc->status & E1000_RXD_STAT_DD)){
        struct mbuf *m = rx_mbufs[tail];
        m->len = desc->length;
        net_rx(m);

        rx_mbufs[tail] = mbufalloc(0);

        desc->status = 0;
        desc->addr = (uint64) rx_mbufs[tail]->head;

        tail = (tail + 1) % RX_RING_SIZE;
        desc = &rx_ring[tail];
    }

    regs[E1000_RDT] = (tail + RX_RING_SIZE - 1) % RX_RING_SIZE;
}

```

1.1.2 实验测试结果

```
== Test running nettests ==  
$ make qemu-gdb  
(3.8s)  
== Test    nettest: ping ==  
    nettest: ping: OK  
== Test    nettest: single process ==  
    nettest: single process: OK  
== Test    nettest: multi-process ==  
    nettest: multi-process: OK  
== Test    nettest: DNS ==  
    nettest: DNS: OK  
== Test time ==  
time: OK  
Score: 100/100
```