**Lab 10 networking(hard)**

* 1. **实验目的**

为 xv6 操作系统编写一个网络接口卡（NIC）的设备驱动程序。

即完成e1000网络设备的驱动，使得操作系统能够通过这个驱动与网络进行通信。

* 1. **实验步骤**
     1. 查看kernel/e1000.c、net.c等文件,了解E1000设备等

通过环形队列（Ring Buffers）管理要发送和接收的数据包。环形队列分为TX与RX（发送与接收）

环形队列描述符与指针数组定义如下：

define TX\_RING\_SIZE 16

static struct tx\_desc tx\_ring[TX\_RING\_SIZE] \_\_attribute\_\_((aligned(16)));

static struct mbuf \*tx\_mbufs[TX\_RING\_SIZE];

#define RX\_RING\_SIZE 16

static struct rx\_desc rx\_ring[RX\_RING\_SIZE] \_\_attribute\_\_((aligned(16)));

static struct mbuf \*rx\_mbufs[RX\_RING\_SIZE];

E1000 能够直接在内存中读取和写入数据，即直接内存访问（DMA）

接收数据包时，E1000 设备将数据包的内容写入一个环形队列的描述符中。

* + 1. 实现上传传输包

kernel/e1000.c

int

e1000\_transmit(struct mbuf \*m)

{

// 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.

//

// mbuf 包含一个以太网帧；将其编程到 TX 描述符环中，以便 e1000 发送它。保存一个指针，以便在发送后可以释放它。

acquire(&e1000\_lock);

// 查询下一个 packet 的下标

int idx = regs[E1000\_TDT];

检查前一个传输是否完成，若未完成则释放当前锁并返回错误。

// 检查前一个传输是否完成

if (!(tx\_ring[idx].status & E1000\_TXD\_STAT\_DD)) {

release(&e1000\_lock);

return -1; // 若未完成,返回错误

}

如果上一个包成功完毕，则释放mbufs内存等待新的包填充进来。

// 释放上一个包的内存

if (tx\_mbufs[idx])

mbuffree(tx\_mbufs[idx]);

新包的填充。

// 填充新的网络包信息到 ring 的下标位置

tx\_mbufs[idx] = m;

tx\_ring[idx].length = m->len;

tx\_ring[idx].addr = (uint64) m->head;

tx\_ring[idx].cmd = E1000\_TXD\_CMD\_RS | E1000\_TXD\_CMD\_EOP;

regs[E1000\_TDT] = (idx + 1) % TX\_RING\_SIZE;

release(&e1000\_lock);

return 0;

}

* + 1. 实现接收包

kernel/e1000.c

static void

e1000\_recv(void)

{

// Check for packets that have arrived from the e1000

// Create and deliver an mbuf for each packet (using net\_rx()).

//

// 检查从 e1000 接收到的数据包

// 为每个数据包创建并传递一个 mbuf（使用 net\_rx()）。

循环获取环中下一个待处理描述符索引。

while (1) {

// 获取描述符环中下一个待处理的描述符的索引

int idx = (regs[E1000\_RDT] + 1) % RX\_RING\_SIZE;

在每次循环开始时检查描述符状态确定是否继续。

// 检查描述符的状态

if (!(rx\_ring[idx].status & E1000\_RXD\_STAT\_DD))

return; // 没有新包了，结束函数

接收传输，接收完毕后更新环形队列及当前索引

// 将数据包的长度设置到 mbuf 中

rx\_mbufs[idx]->len = rx\_ring[idx].length;

// 向上层网络栈传输

// net\_rx定义在net.c中，作用：将从网络接收到的数据包交给网络协议栈进行处理。在操作系统中，数据包从网络接口卡（如网卡）接收后，需要进行一系列的处理，包括解析协议头、路由选择、处理传输层协议（如 TCP、UDP）等操作。

net\_rx(rx\_mbufs[idx]);

if((rx\_mbufs[idx] = mbufalloc(0)) == 0)

panic("e1000");

rx\_ring[idx].status = 0;

rx\_ring[idx].addr = (uint64) rx\_mbufs[idx]->head;

// 更新待处理描述符的索引

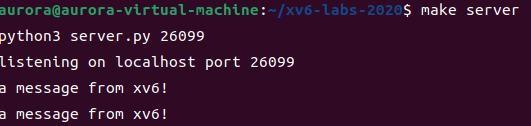
regs[E1000\_RDT] = idx;

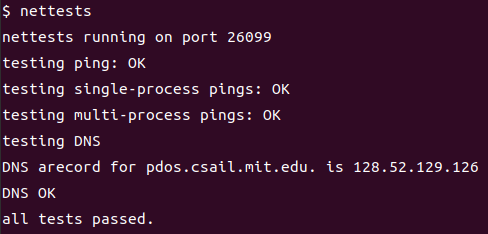
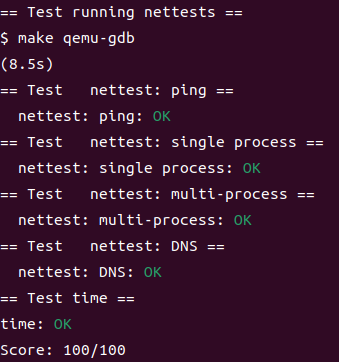
}

}

}

* + 1. 测试，分数生成及提交

与其他实验略有不同，在实验目录下先要在一个终端中使用make server构建。

然后在另一个终端中使用make qemu或make grade进行测试 

nettests与grade运行结果。

* 1. **实验中遇到的问题和解决方法**

问题：对e1000设备及net相关操作不熟悉，对部分函数或结构体内容无法较好地理解。

解决方式：快速浏览e1000软件开发手册，学习e1000.c等代码文件及相关他人笔记。

* 1. **实验心得**

本实验代码量较少，但是接触了较新的网络操作，有一定学习成本。完成后发现，对于该实验而言，实现功能并不需要全部了解其内核net操作的过程或是e1000的设备特征等，只需要知道如何使用部分信息并在自己的代码部分中实现即可。