实验十

Networking

目的：实现网卡，并支持UDPsocket编程。

相关内容在xv6第四章和第七章（7.13）。E1000是xv6自带的虚拟网络设备，连接在局域网中，xv6本机（guest）IP为10.0.2.15，而另一台虚拟出来的机器（host）的IP为10.0.2.2。因为不需要管理员权限，我们将运行在QEMU的用户模式网络栈。收发的网络包都将存储在packets.pcap 中，用命令tcpdump -XXnr packets.pcap 可以得到对应的包。Xv6已经创建了net.c/h代码文件。

**具体操作：**

网络设备驱动：实现E1000网络驱动，给定的代码只能初始化设备以及处理中断，但是不能收发包。

为了实现E1000的收发包，需要阅读intel 以太网控制器软件编程说明书。EMU模拟的是82540EM。阅读说明书的时候，大概看一下第二章了解整体，然后主要阅读第三章和第四章以及4.1（子目录不用看了），第十三章可以作为参考。其它的部分用处不大，这个实验只需要用到一小部分知识。



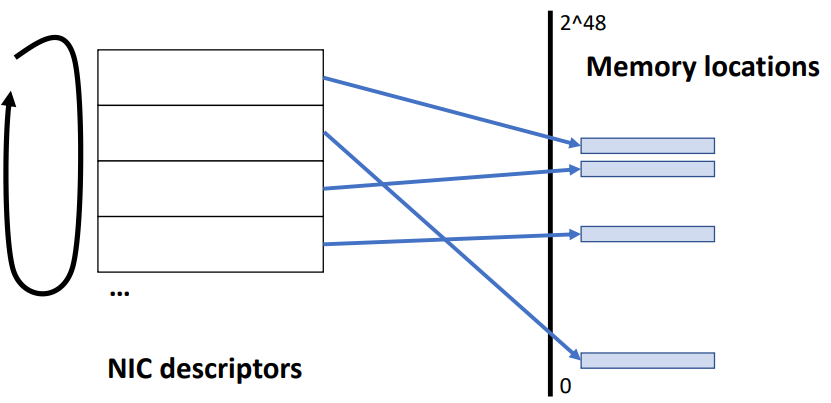
网络套接字：支持用户态程序使用网络功能。

套接字的访问是通过文件描述符来实现的，读文件就是获取数据包，写文件就是发送数据包。如果读的时候没有收到数据包，就会进入阻塞状态，等待接受下一个数据包（一般此时进程会被切换）。本实验要求实现简单版的套接字以支持UDP网络协议。

Connect系统调用提供创建和绑定端口和地址的套接字功能，并返回文件描述符，该系统调用在sysfile.c中实现。Sockalloc()和相关文件放在sysnet.c。编程的时候注意一下给定的数据结构，struct sock 是socket的数据结构，sockets是所有正在运行的socket的单向链表（找到对应的socket以传送数据包非常重要），不仅如此，每一个socket都保持一个等待接收数据的mbufs队列，收到包就放在里面等待用户read()取出。

步骤：

1. E1000收发包：xv6与e1000在内存中共享描述符队列，这些队列提供了内存中数据包的指针。他们以环形数组实现，意味着如果读取到数组末尾将转而从头开始。一般来说将收到的数据简写为RX，发送的数据简写为TX。E1000收到新的数据包时，会触发中断，这个时候应该要扫描RX队列查看哪些新的包到达了，并且调用net\_rx()将其送到协议层。Rx\_desc结构体描述了描述符的框架。



Usertrap对应的解决方案：

* 首先，如何实现lazy？分配与释放的时候都按照一个块分配。释放的时候要注意，是按照页表进行地址的查找，找到了addr再释放，并设置这段空间已被释放。
* 第二，如何知道所有的map都已经被unmap？将每个VMA以一个PGSIZE为单位来存放，而非一整个size，这样避免了切分问题。但是要注意，这样会产生额外参数offset，在读取文件的时候，以及写入文件的时候要注意。还有，最终由于无法调节的锁bug原因，转而调用filewrite实现（用f->offset）控制offset。
* 第三，不是所有的map都被允许，如果传入的flag和文件的标识符不是被包含的关系，那么就不能够正常map。但是，如果是MAP\_PRIVATE模式，写就无所谓了。
* 第四：小心，mmaptest测试样例只有1.5个PAGE的’a’，但是却要读出两个PAGE的数据，并且test默认最后0.5PAGE为0，如果我们直接从file中读，是读不出的，只能够将内存超过部分置零。
* 第五（后来发现实际上要先考虑fork再考虑exit）：修改exit实际上就是munmap，那么把munmap单独放出来包装一层作为独立的函数就好啦。修改fork相反，就是mmap一下。还有一些些要注意，exit不会释放对应进程的资源，wait会，释放资源的时候，因为lazy，在unmap的时候会发现一些没有映射的页表，这个时候和下面的uvmcopy一样，略过就好。
* 第六：fork需要解决更多的问题，迎面而来的问题就是页表复制问题。页表中含有需要我们自身进行管理的vma部分。这里，我们采取的办法就是，卸下vma相关的页表（要写回file），但是不卸载vma（也不closefile），之后对新的进程进行vma的mmap（要进行filedup）。卸载之后，页表copy的时候会因为卸载和lazy策略的原因，会出现部分虚拟地址未映射的情况，因此需要uvmcopy略过这种情况。

