操作系统课程设计

Xv6实验报告

Lab7 networking

|  |  |
| --- | --- |
| Name | 朱从周 |
| ID | 2351893 |
| Number | 42028701 |
| jor | 软件工程 |



同济大学

Tongji University

1. **环境搭建**
2. 下载VMware虚拟机，并且下载 Ubuntu 镜像并安装 Ubuntu 系统。
3. 启动Ubuntu，安装本项目所需的所有软件，运行：

$ sudo apt-get update && sudo apt-get upgrade

$ sudo apt-get install git build-essential gdb-multiarch qemu-system-misc gcc-riscv64-linux-gnu binutils-riscv64-linux-gnu

1. 下载xv6内核源码

$ git clone git://github.com/mit-pdos/xv6-riscv.git

1. 更新镜像源

$ sudo nano /etc/apt/sources.list

$ sudo apt-get update

1. 获取源码

$ git clone git://g.csail.mit.edu/xv6-labs-2021

$ cd xv6-labs-2021

$ git checkout net

**1. 实验目的**

编写一个在xv6操作系统中用于网络接口卡（network interface card, NIC）的设备驱动程序。通过这个实验，你将学习如何初始化并操作一个虚拟的网络设备，以及如何处理网络通信，从而深入理解操作系统中设备驱动程序的工作原理。

实现 kernel/e1000.c 中的 e1000\_transmit() 和 e1000\_recv() 函数, 使得网卡驱动可以发送和接收数据包.

**2. 实验内容**

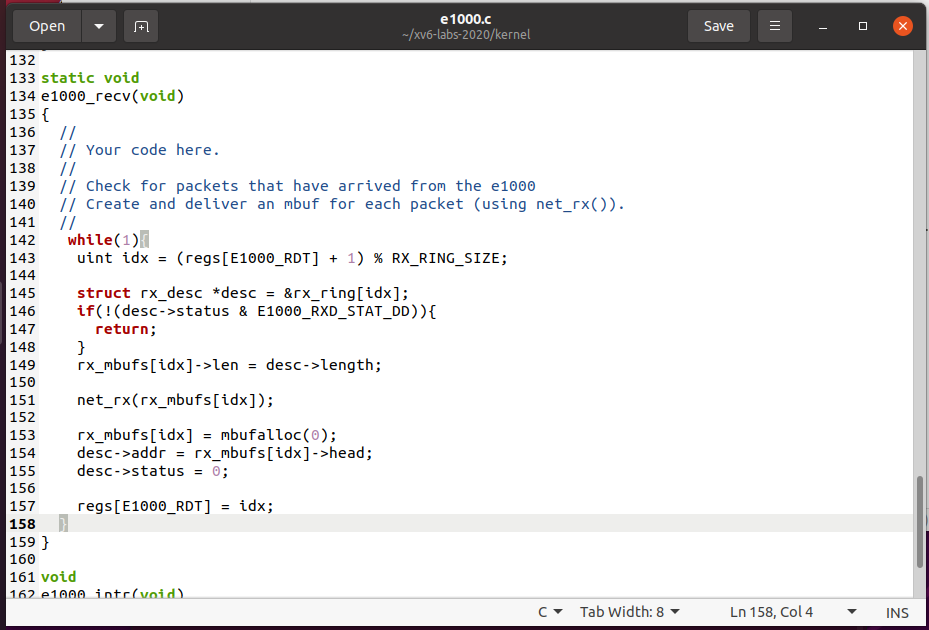
**发送函数 e1000\_transmit**

1. 获取当前环形队列的tail位置。
2. 检查tail位置的描述符状态，如果没有DD（Descriptor Done）标志位，表示队列已满。
3. 清理已发送的描述符缓存，并设置新的描述符地址和长度。
4. 设置发送命令，并更新tail位置。

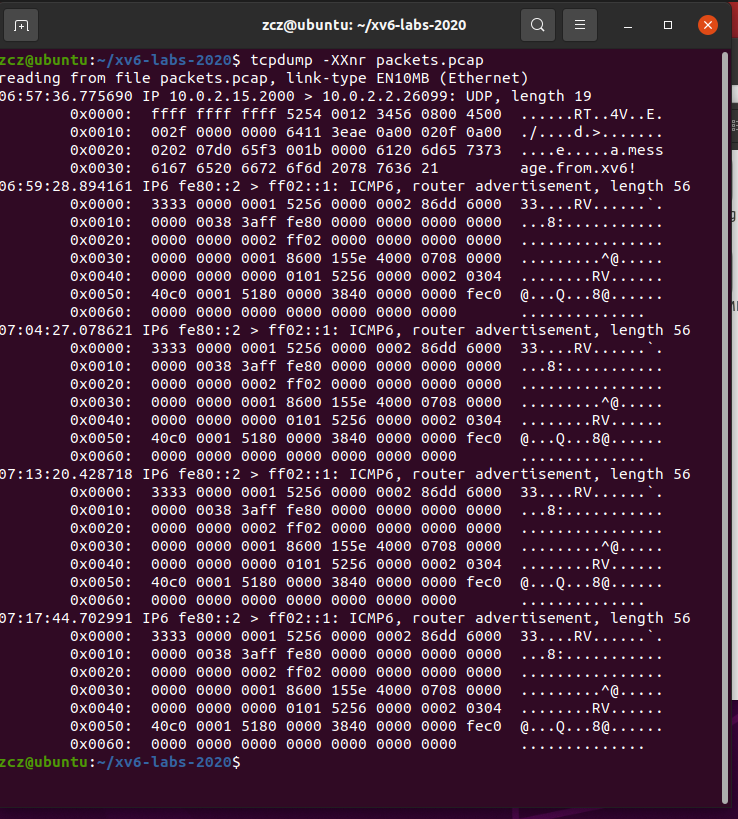


**接收函数 e1000\_recv**

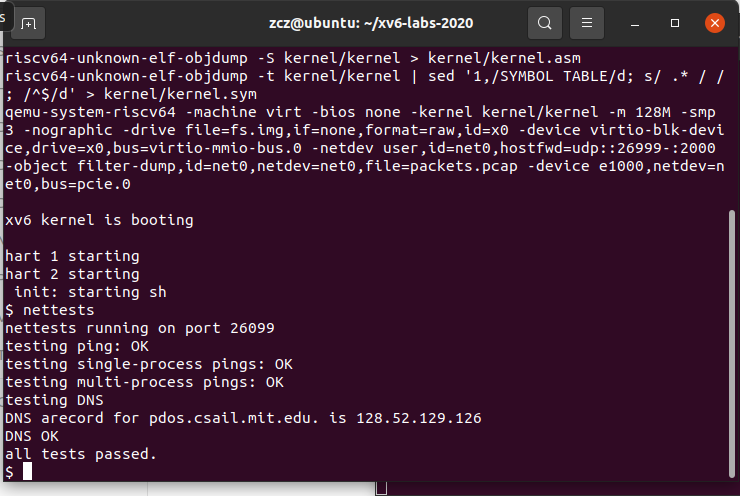
1. 读取环形队列的tail位置。
2. 检查tail位置的描述符状态，如果有DD标志位，表示数据包已接收。
3. 调用net\_rx函数处理接收到的数据包，并分配新的mbuf。
4. 更新描述符地址和状态，更新tail位置。



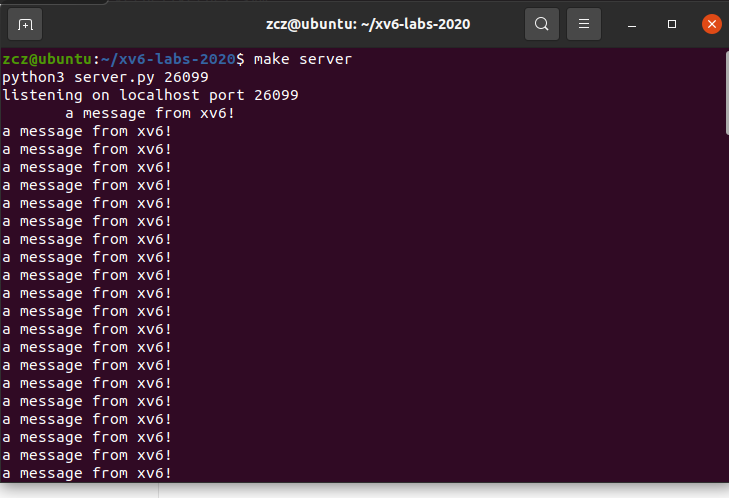
**测试结果：**

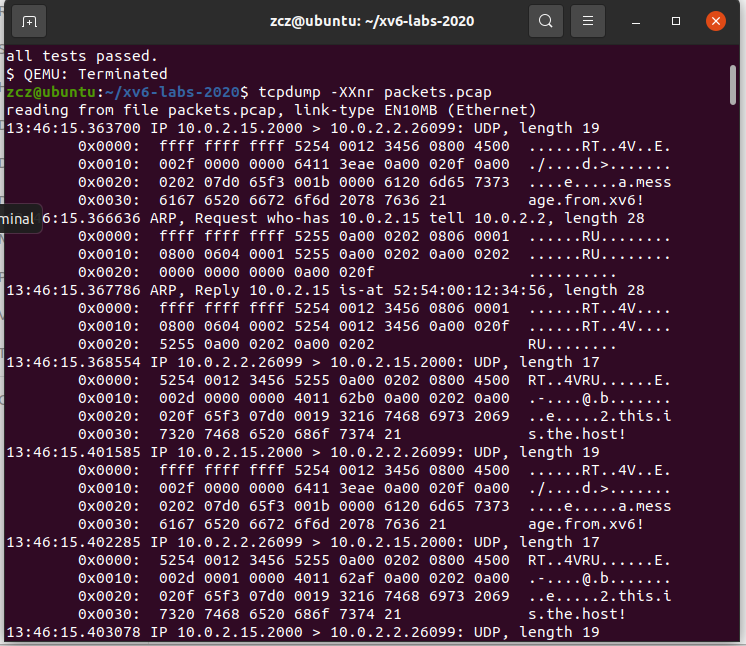


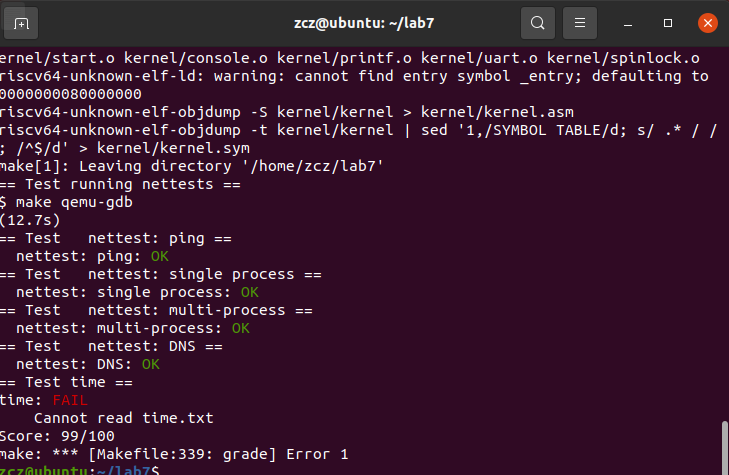
然后在另一个终端执行 make qemu 启动 xv6, 然后执行 nettests 命令进行测试:



在 xv6 目录下执行 make server 启动服务端.







**3.实验心得**

通过这次实验，我深入了解了设备驱动程序的编写和调试过程，特别是在网络设备上的应用。我学会了如何初始化和操作虚拟网络设备，以及如何处理数据包的发送和接收。同时，我还学会了如何使用互斥锁和同步机制来保证多个进程或线程对设备的正确访问。但对于锁与性能之间更多的知识，希望可以在下一个关于lock的实验中有更深的理解，说不定可以解决我在这个实验中遇到的困惑。