

****

信息学院软件工程系

《计算机网络》实验报告

**题　　目 实验三　基于 PCAP 库侦听并分析网络流量**

**班　　级 软件工程2019级2班**

**姓　　名 李世豪**

**学　　号 22920192204229**

**实验时间 2021年4月03日**

**2021 年 6 月 03 日**

填写说明

1. 本文件为Word模板文件，建议使用Microsoft Word 2019打开，在可填写的区域中如实填写；
2. 填表时，勿破坏排版，勿修改字体字号，打印成PDF文件提交；
3. 文件总大小尽量控制在1MB以下，勿超过5MB；
4. 应将材料清单上传在代码托管平台上；
5. 在学期最后一节课前按要求打包发送至cni21@qq.com。

# 实验目的

通过完成实验，理解数据链路层、网络层、传输层和应用层的基本原理。掌 握用 Wireshark 观察网络流量并辅助网络侦听相关的编程；掌握用 Libpcap 或 WinPcap 库侦听并处理以太网帧和 IP 报文的方法；熟悉以太网帧、IP 报文、TCP 段和 FTP 命令的格式概念，掌握 TCP 协议的基本机制；熟悉帧头部或 IP 报文头 部各字段的含义。熟悉 TCP 段和 FTP 数据协议的概念，熟悉段头部各字段和 FTP 控制命令的指令和数据的含义。

实验任务：

1. 用侦听解析软件观察数据格式
2. 用侦听解析软件观察 TCP 机制
3. 用 Libpcap 或 WinPcap 库侦听网络数据
4. 解析侦听到的网络数据

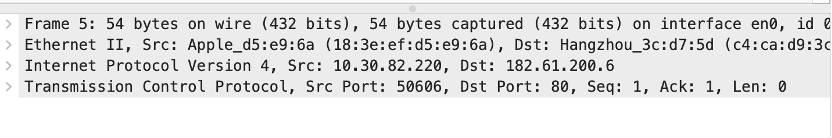
# 实验环境

操作系统：Mac Big Sur 11.3（类Unix系统）

编程语言：C/C++，libpcap库

# 实验结果

。(1) 网络协议层嵌套



如图，在wireshark中捕捉到的数据包packet中：

第1行描述的含义：

Frame 5: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface en0, id 0

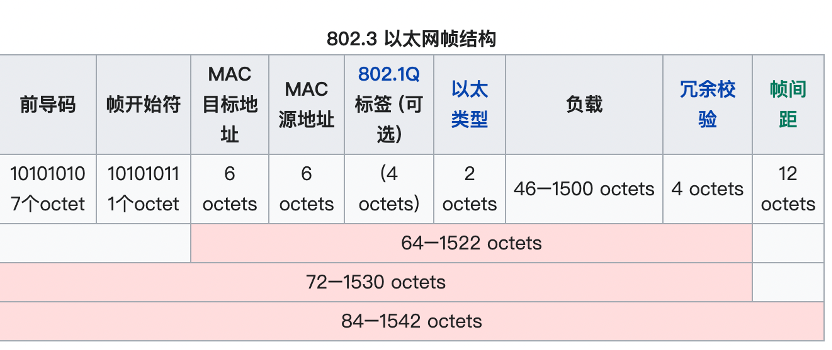
即数据包在本机pc网络接口处被捕获的记录，是网络层次中物理层的表现，表达含义即共54bytes在网络线路中传输，共54bytes在网络接口en0中被捕捉。

第2行到第4行描述：

以太网帧层次报文头，网络层ip报文头，传输层tcp报文头，层层嵌套，每层都包含下一层的所有信息作为其负载data

以上即计算机网络中的网络嵌套机制模型。

以太网帧格式：

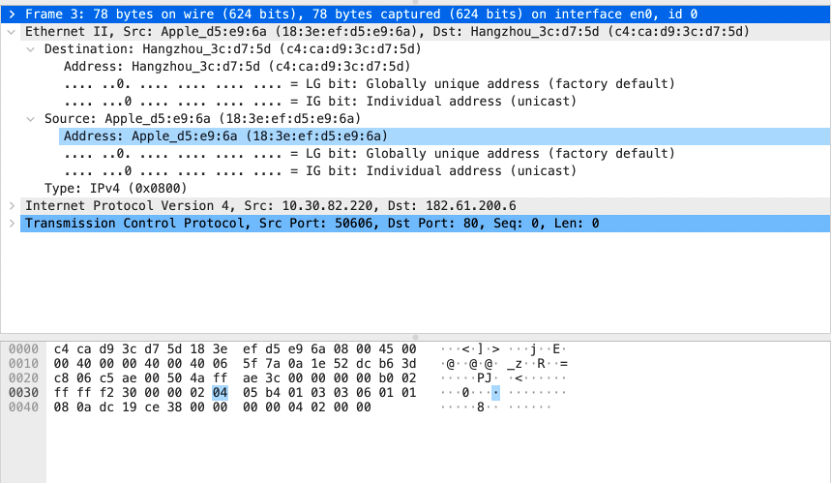


图源：维基百科-《以太网帧格式》

Wireshark捕捉packet的以太网帧格式见下图：

以太网帧的前导码和帧首定位符在链路传输中，在接口处被过滤，因此以太网帧以MAC地址开始；

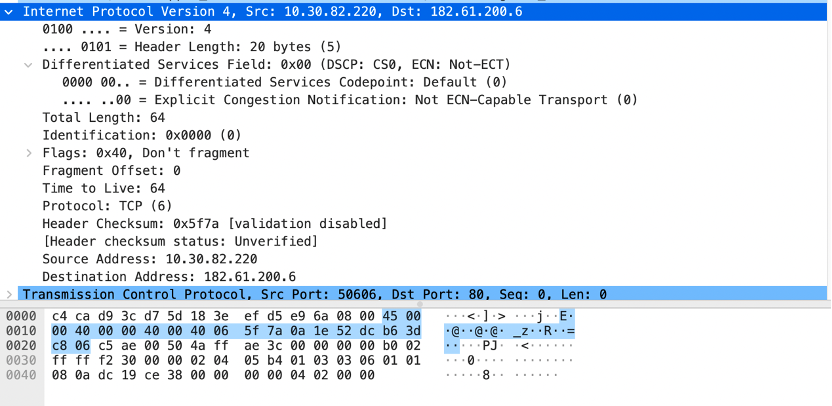
如图，前6bytes为MAC目的地址，紧接6bytes为MAC源地址，再接2bytes为下一层的协议类型，这里为0x0800，代表是IPv4协议。在这之后的数据即为负载。



IP报文格式：



图源：维基百科—《IPv4》

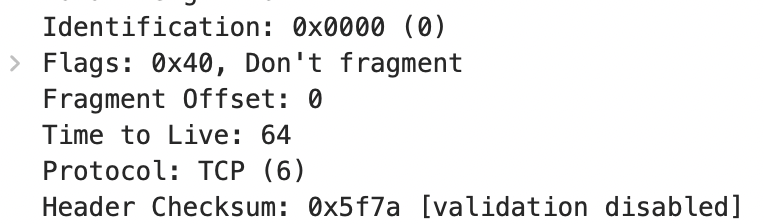


第1个bytes对应版本控制和IP报文长度，前4bits为版本，后4bits为长度。

第2个byte对应区分服务领域，包括区分服务和拥塞通告等

第3-4个bytes对应IP数据包的长度，即IP报文长度+数据长度

之后2bytes为标识符，4bits标志号，12bits分片偏移，1byte存活时间，2byte传输层协议类型，2bytes校验和checksum，如图



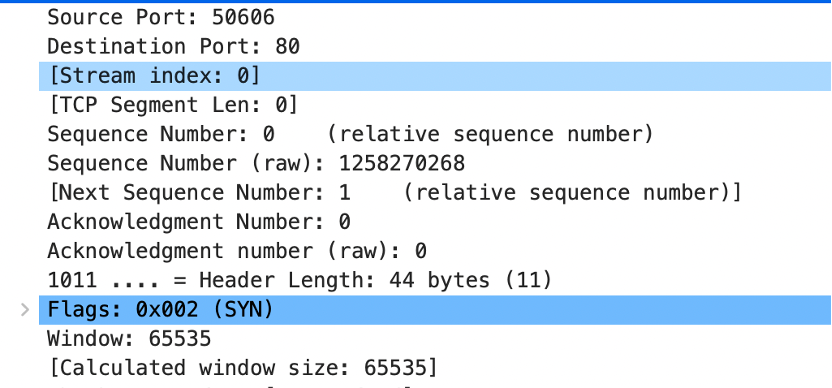
TCP段格式：



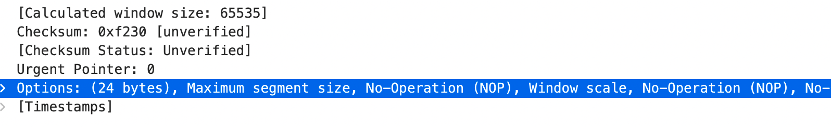
图源：维基百科-《传输控制协议》

前4bytes分别表示源端口和目的端口，之后4bytes序列号，4bytes ACKNumber，

后2bytes为标志等，包括段长度，保留符，标志符等，后2bytes为窗口大小，如下图：



紧接的4bytes分别为checksum和urgent pointer；最后为可选项：

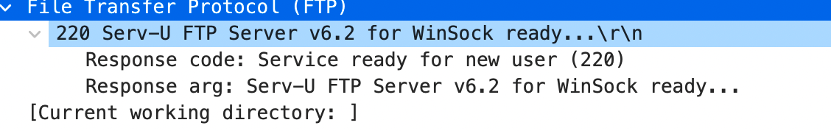


FTP命令和响应格式：

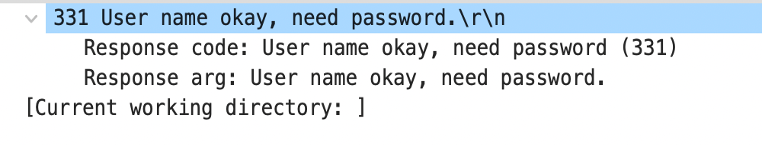
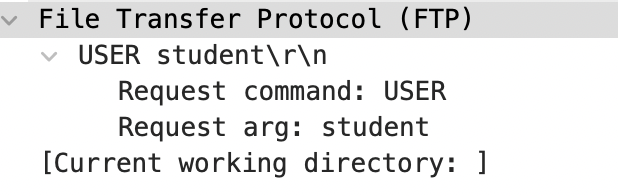
FTP中数据的传输分两种模式，一是主动模式，一是被动模式。两种模式的主要区别在于发起连接方的不同，主动模式中是由Client主动发送PORT 地址端口到Server，发出TCP连接请求，Server接收请求后进行确认后发起TCP连接；而被动模式中，有Server确认到Client发出的PASV命令请求后，Server发送地址端口到Client中，由Client发起TCP连接。但这里不对这两种数据流传输方式进行验证说明。

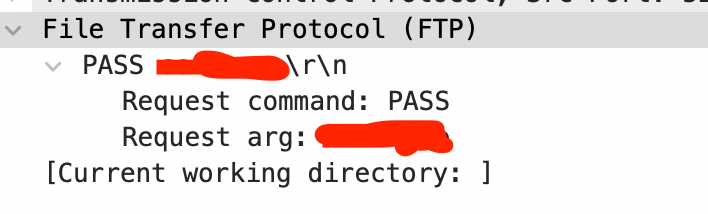
详细可见<https://blog.csdn.net/hhd1988/article/details/114885609>

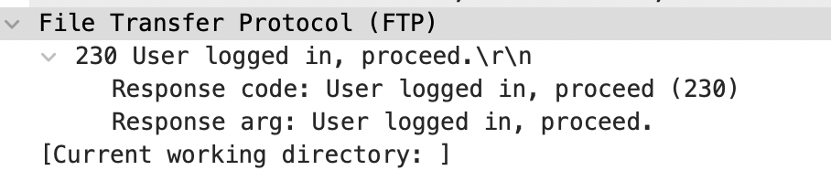
服务器准备响应： 状态符号（220，530）+响应参数



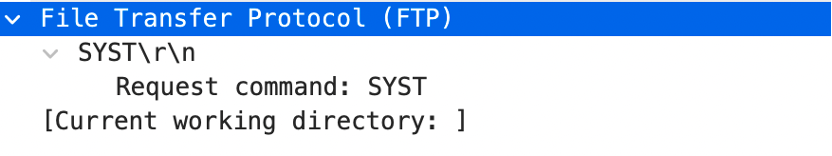
用户标识认证：







其他命令还有：更换目录，要求文件等等，这里不再展示



2、用侦听解析软件观察 TCP 机制

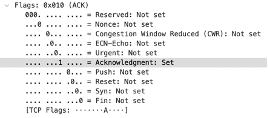
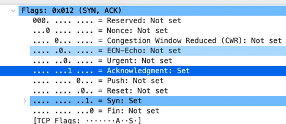
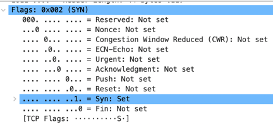
用 Wireshark 侦听并观察 TCP 数据段。观察其建立和撤除连接的过程，观察 段 ID、窗口机制和拥塞控制机制等。将该过程截图在报告中。

建立：

TCP三次握手：



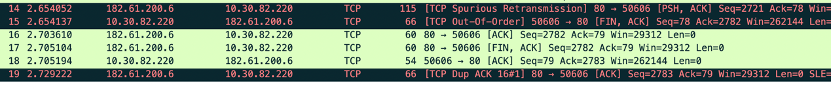
如图，首先原目的地址的client发送连接请求，标志符置SYN=1，序列号Seq=0；Server端收到连接请求后返回确认响应，标志符置SYN=1，ACK=1，置序列号Seq=0，Ack=1；Client端接收到确认响应后发送确认连接请求，建立连接，标志符置ACK=1，序列号置Seq=1，Ack=1。



撤除连接：

TCP三次握手撤除连接：

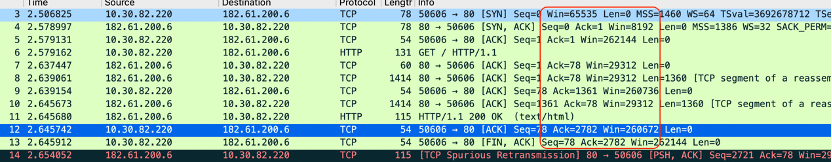
Client发送关闭Close请求，标志符置FIN，ACK；Server收到close请求，检查ACK序列号是否=client发送的Seq+1，若是则标志符置FIN，ACK，序列号Ack+1，返回响应信息；Client收到确认关闭信息，置ACK+1，注意此时Seq不变，发送确认关闭数据段包，关闭连接。Server端收到信息，检查出现DUP错误，说明连接结束，断开连接。



窗口机制：

即在TCP数据段中，Client使用2bytes 窗口大小字节来表达当前可接收数据的最大字节数，Server接收到信息后，不能响应发送大于client窗口大小的数据包；Client根据当前剩余的可接收量，不断修改窗口大小，当窗口大小为0时，说明Client无法接受字节，此时server端开始一个计时器，停止发送数据并进行一定时间的计时，计时结束时重新进行数据发送，并再次开启窗口机制来控制流量发送。注意：窗口流量控制机制时发送在发送端和接受端两侧的终端流量控制，而不是在链路中的流量控制。

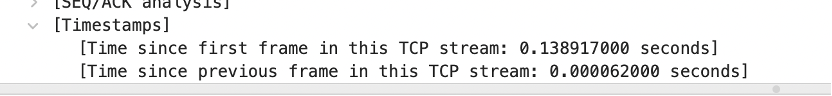
如图：接收端窗口大小在不断的变化，确认可接收数据包大小。



拥塞控制：

拥塞控制主要是处理链路层中数据传输发生的拥塞崩溃等，主要是使用RTT，来回通信时延，计时器，来确定网络链路的通信状况，若RTT高，即网络拥堵，则减小发送量，降低发送速率；若RTT小，则增大发送量，提高发送速率。

如图为TCP中RTT处理信息：



3、用 Libpcap 或 WinPcap 库侦听网络数据

用 Libpcap 或 WinPcap 库侦听网络上的数据流，解析发送方与接收方的 MAC 和 IP 地址，并作记录与统计。程序在文件上输出形如下列 CSV 格式的日志：

时间、源 MAC、源 IP、目标 MAC、目标 IP、帧长度（以逗号间隔）

2015-03-14 13:05:16,60-36-DD-7D-D5-21,192.168.33.1,60-36DD-7D-D5-72,192.168.33.2,1536

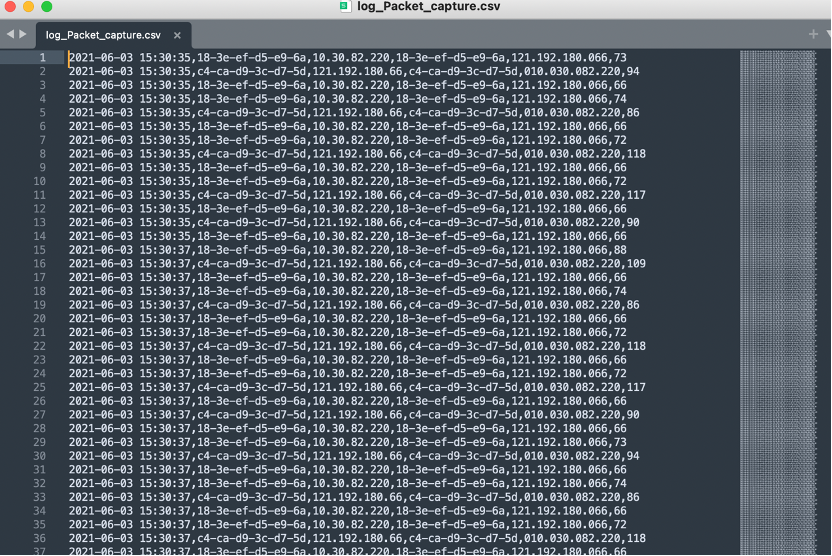
每隔一段时间（如 1 分钟），程序统计来自不同 MAC 和 IP 地址的通信数 据长度，统计发至不同 MAC 和 IP 地址的通信数据长度。

程序运行如下：

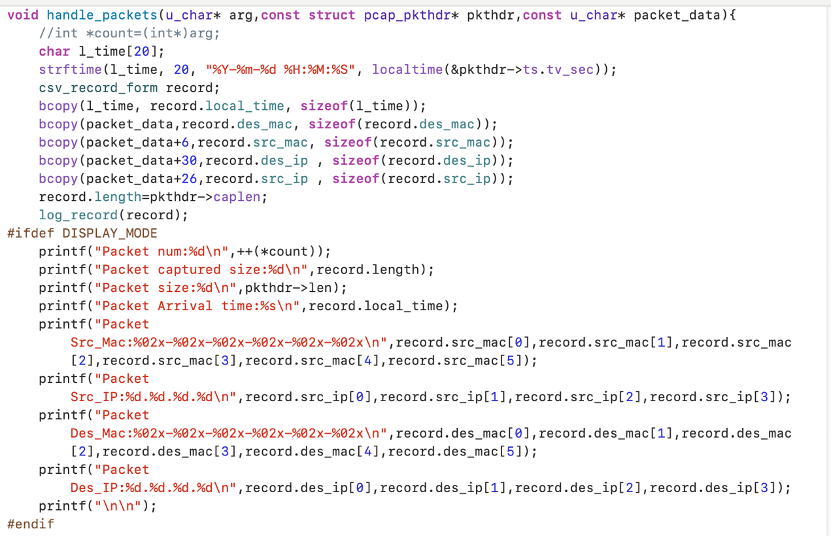


程序在运行开始时，对本地网络接口en0（一般WIFI网络接口），进行一个数据统计，每个5sec进行一次新的统计，统计来自不同MAC地址和IP地址的数据包总长度以及发送到不同MAC和IP的总数据包大小。

记录每一条记录的CSV文件如图：



代码核心部分：主要是对libpcap捕捉到的数据包的以太网报文，IP报文，TCP报文头进行解析，抓取其中的源MAC，目标MAC，源IP，目标IP等等字节：如下示：



4、解析侦听到的网络数据

用 Wireshark 侦听并观察 FTP 数据，分析其用户名密码所在报文的上下文特 征，再总结出提取用户名密码的有效方法。解析协议内容，并作记录与统计。对 用户登录行为进行记录。程序在文件上输出形如下列 CSV 格式的日志：

时间、源 MAC、源 IP、目标 MAC、目标 IP、登录名、口令、成功与否

2015-03-14 13:05:16,60-36-DD-7D-D5-21,192.168.33.1,60-36DD-7D-D5-72,192.168.33.2,student,software,SUCCEED

2015-03-14 13:05:16,60-36-DD-7D-D5-21,192.168.33.1,60-36DD-7D-D5-72,192.168.33.2,student,software1,FAILED

题目简析：使用wireshark对FTP的命令通信进行学习，并解析FTP的用户登陆机制，获取用户的账号，密码，并获取成功失败状态。

FTP数据包抓包分析：

首先要过滤是否是FTP协议，

//判断方法一：对是否包含命令字段进行判断，如客户端请求一般包含字段（SYST，USER，PASS，CWD，PCWD，FEAT，PASV，PORT，MKD，EPRT，DELET，EPSV，LIST等等），服务器端返回码3字节（200：成功，202:命令未执行，220:服务准备就绪，227:进入被动模式，331，332:账号密码，421:服务器不可用，450:文件不可用，500：无效命令，等等）

//判断方法二：对端口进行一个过滤选择，命令控制端口为21，数据传输端口为20

//这里选择使用方法二，设置pcap过滤器对端口进行过滤，这里只保留端口21，进行一个命令控制学习

//FTP命令报文格式，客户端：CMD 参数 服务器：状态码 参数

//因为FTP是基于TCP协议，是TCP协议的一部分，而一般包含FTP的TCP协议的报文头长度为32bytes，新增了12bytes可选项描述tcp传输中的状态

//因此可以认为TCP报文开始后32bytes之后的数据即为FTP命令请求报文

//判断FTP报文结束，即FTP固定在CMD模式中以\t\n结束，hex分别为0x0d,0x0a;

核心代码如下：

解析FTP命令数据包中的用户密码信息：

**bool** user\_ready=**false**;

u\_char USER[20];

bzero(USER, 20);

**if** (memcmp((u\_char\*)"USER", FTP\_stream,4)==0) {

pcap\_pkthdr pk;

**const** u\_char\* data=pcap\_next(device\_fp, &pk);

**if**(data==**NULL**){

printf("not capture.");

exit(-1);

}

**if** (memcmp(data+14+20+32, (u\_char\*)"331", 3)==0) {

user\_ready=**true**;

**int** i=5;

**while** (FTP\_stream[i]!=0x0d&&FTP\_stream[i+1]!=0x0a) {

USER[i-5]=FTP\_stream[i];

i++;

}

printf("USER:%s\n",USER);

}

}

**bool** pass\_ready=**false**;

u\_char PASS[20];

pcap\_pkthdr pass\_pk;//过滤每次server返回状态后client发送确认tcp协议数据包

pcap\_next(device\_fp, &pass\_pk);

**const** u\_char\* send\_pass=pcap\_next(device\_fp, &pass\_pk);

**if** (memcmp(send\_pass+14+20+32, (u\_char\*)"PASS", 4)==0) {

pcap\_pkthdr ret\_pk;

**int** pass\_len=pass\_pk.caplen-2-5-14-20-32;

memcpy(PASS, send\_pass+14+20+32+5, pass\_len);

printf("PASS:%s\n",PASS);

**const** u\_char\* ret=pcap\_next(device\_fp, &ret\_pk);

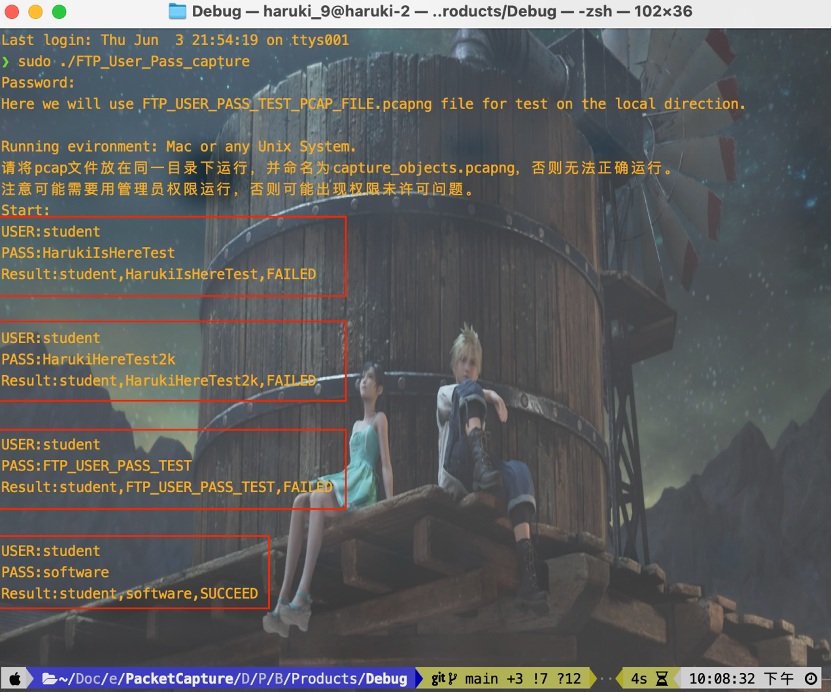
**if** (memcmp("230", ret+14+20+32, 3)==0) {

pass\_ready=**true**;

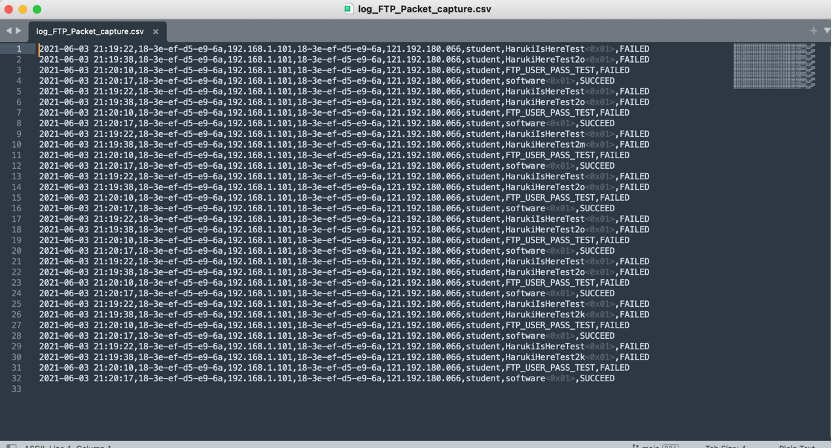
}

}

程序运行效果如图：



CSV文件记录如下图示：



# 实验代码

本次实验的代码已上传于以下代码仓库：代码放置于Github，地址如下：

Github：<https://github.com/Haruki9/Computer-Network_Labs/tree/main>

Gitee：<https://gitee.com/haruki9/computer-network_-labs>

# 实验总结

对TCP，IP，以太网帧格式有了进一步了解，并加深记忆，对于FTP的工作方式也有了更深的了解，对于FTP的两个工作模式也终于明白了。