

南京邮电大学

实验报告

(17/18 学年 第一学期)

题 目： Linux 下的多进程 / 线程网络通信

专 业 网络工程

学 生 姓 名 刘美含

班 级 学 号 B15070204

指 导 教 师 吴振宇

指 导 单 位 物联网学院

日 期 2017.12.18

Linux 下的多进程 / 线程网络通信

一、概述

1、课题目标

本课题要求在 Linux 下使用 C 语言实现一个抓取并分析网络数据的系统。

2、内容概述

该系统采用 C/S 开发模式。

在客户端，设计两个进程 P1 和 P2 同时读取网卡上的数据，P1 读取数据后，负责将 TCP 报文中的头部数据写入到 Packet 文件中，P2 读取数据后，负责将 UDP 报文的头部同样写入到 Packet 文件中。

另外，还需要将数据发送到服务器端存储。服务器端软件需要一个进程接收客户端发送的文件，并将接收到的数据保存成一份文件。程序运行之后，用户可以随时终止程序的运行，要求两个进程 P1 和 P2 在结束前将各自读取的报文数量和各自写入 Packet 文件中报文的数量分别写入 Report 文件中，并计算读取与写入的百分比写入到 Report 文件。

3、课题分析

经初步分析，本课题涉及到进程 / 线程创建与销毁、信号量、互斥量、用 socket 实现网络通信、TCP 与 UDP 协议、文件的读取和写入等知识点的灵活运用。

因此，我采取增量开发的方式实现本系统。实现的增量依次为：抓包、建立互斥多线程、添加信号量、文件的写入与读取、服务器端与客户端的通信与信息传输。

二、需求分析

随着因特网技术的发展，网络通信问题日益成为人们关注的焦点，网络数据包分析程序便是一种分析网络状况的有效方法，它从数据包流量分析角度出发，通过实时地收集和监视网络数据包信息并对数据包进行分析，来检查网络数据传输的状态，从而监控网络环境。当前，网络在人们生活中发挥着越来越大的作用，人们也对网络有了越来越高的需求，要适应时代的需要，提供完善便利的网络服务，进行对网络数据包的分析课题研究很有必要。分析网络流量,开展对网络数据包的监测活动,对网络的发展都具有极其重要的意义。

本系统开发的目标即采用捕获一段时间内以本机为源地址的 IP 数据包的方法，具体的数据包包括 TCP 数据包和 UDP 数据包，以此来统计 IP 数据包的信息。针对于 IP 包的分析可以得到以下内容：本地 IP 地址及与本机发生数据通信的 IP 地址，交换的数据包数量、数目，IP 包的版本和生存时间等。通过对这些内容的分析，网络管理员就可以了解计算机通信的情况，从而做出应对措施。

本系统共有三个模块，分别为客户端模块、服务器模块和抓包模块，下面对这三个模块分别进行需求分析。

1、 客户端模块

客户端模块主要实现数据发送到服务器的功能。如图 1 所示：

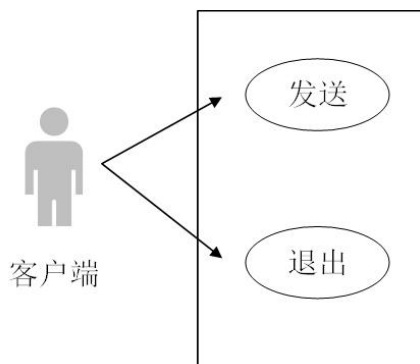


图 1 客户端模块

2、 服务器端模块

服务器端模块的用例图如图 2 所示，主要功能实现如下：

- (1) 接收客户端发送的文件；
- (2) 将接收到的数据保存成一份文件。
- (3) 程序运行之后，用户可以随时终止程序的运行。

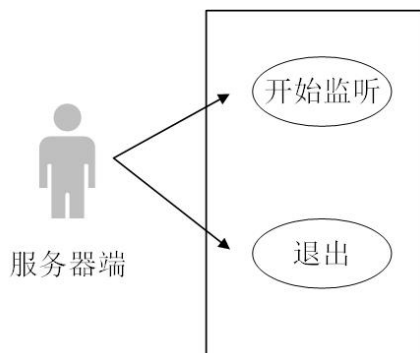


图 2 服务器模块

3、 抓包模块

抓包模块功能实现过程如下：

- (1) 读取网卡上的数据；
- (2) 将 TCP 报文和 UDP 报文中的头部数据写入到文件中；
- (3) 分别统计读取的报文数量和写入文件中报文的数量，计算读取与写入的百分比并写入到文件中。

三、设计与实现

1、总体框架

系统主要由三个子模块组成，分别为：客户端模块、抓包模块和服务器模块。各模块具备不同的功能，其功能框图如图 3 所示。

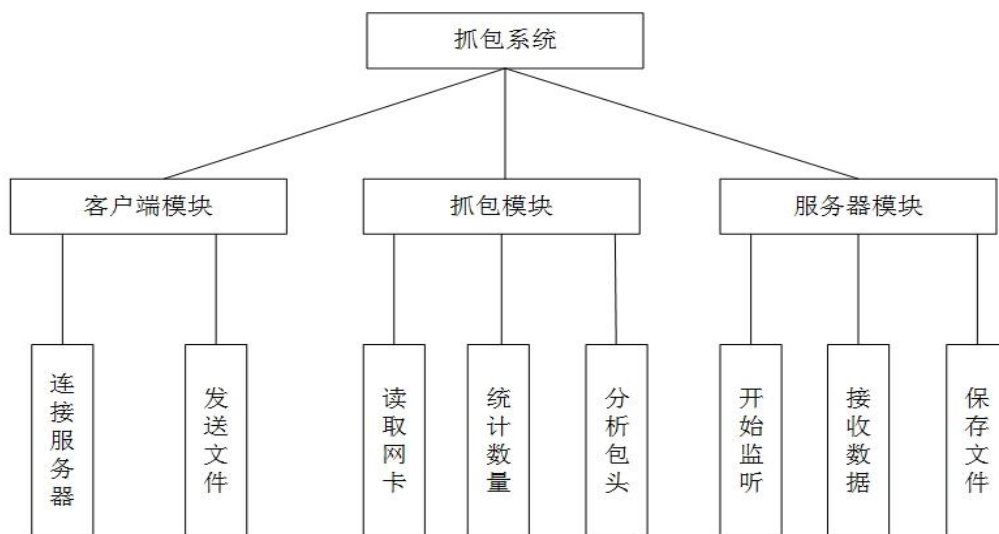


图 3 功能框图

由图 3 可知，客户端模块负责连接服务器和发送文件；抓包模块具有读取网卡、统计数量和分析包头的功能；服务器模块可实现监听、接收数据和保存文件的功能。

总体系统框架图如图 4 所示：

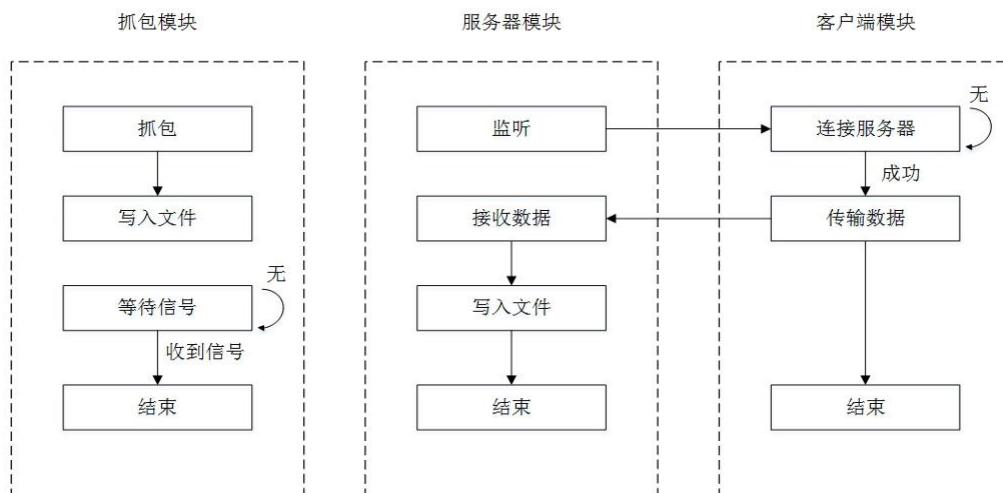


图 4 系统框架图

为了抓取网络数据包，用户运行抓包模块，抓包模块对 TCP 和 UDP 的包头数据进行捕获并等待用户输入信号，一旦收到信号，抓包模块立即结束，计算 TCP 和 UDP 的统计数据写入文件。为了将获取的数据上传到服务器，首选开启服务器，使

其处于监听状态，等待客户端模块连接服务器的信号；一旦客户端发起连接请求，则尝试与其连接；若连接成功，则客户端模块开始传输数据，服务器将接收到的数据写入文件，直到文件传输完毕。

2、客户端模块

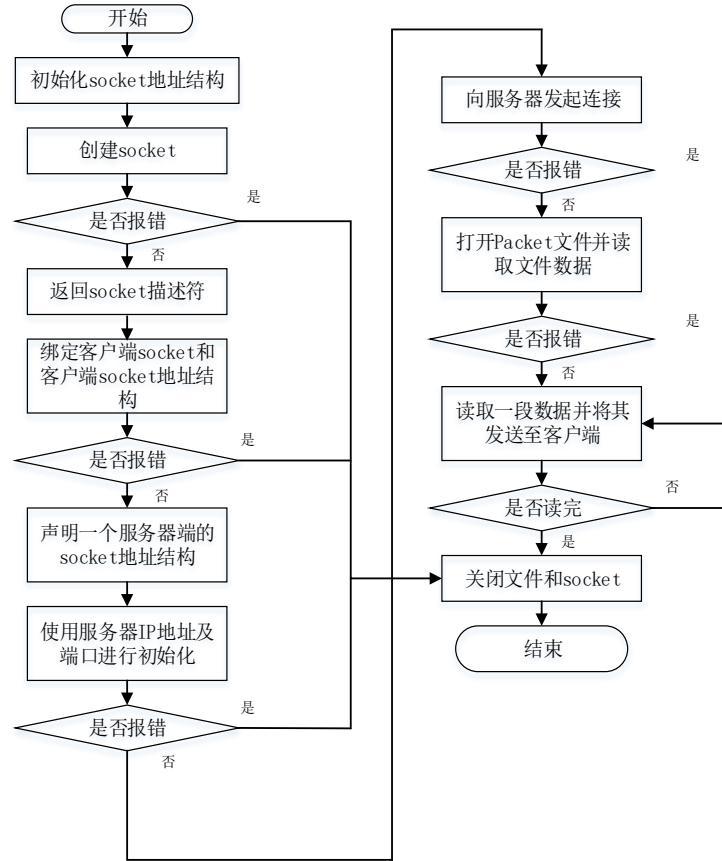


图 5 客户端模块执行流程图

客户端模块执行流程如图 5 所示，步骤如下：

步骤 1：初始化 socket 地址结构；

步骤 2：创建 socket；若成功，返回 socket 描述符；否则报错，转步骤 8；

步骤 3：绑定客户端的 socket 和客户端的 socket 地址结构；若成功，转步骤 4；若失败，转步骤 8；

步骤 4：声明一个服务器端的 socket 地址结构，并用服务器的 IP 地址及端口对其进行初始化；若成功，转步骤 5；若失败，转步骤 8；

步骤 5：向服务器发起连接；若成功，转步骤 6；若失败，转步骤 8；

步骤 6：打开 Packet 文件并读取文件数据；若成功，转步骤 7；若失败，转步骤 8；

步骤 7：每读取一段数据，便将其发送给客户端；若未读完，则继续读；若读完，则转步骤 8；

步骤 8：关闭文件和 socket，程序结束。

3、服务器端模块

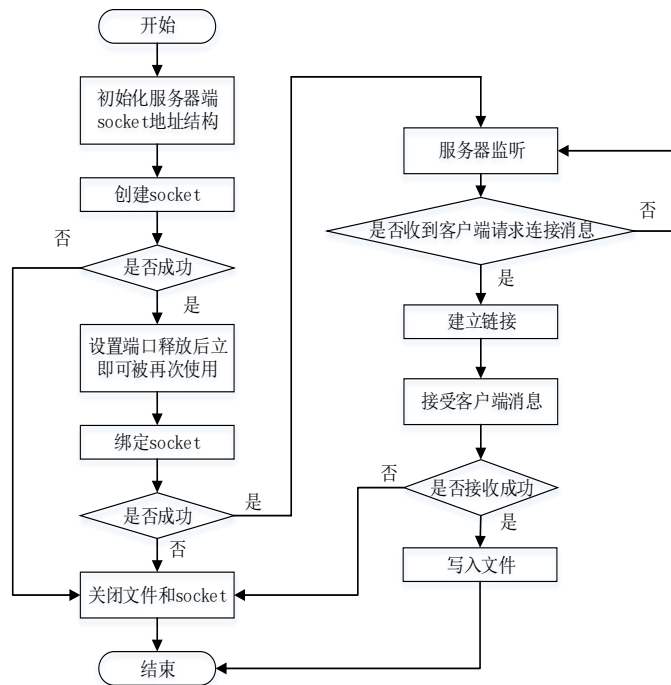


图 6 服务器端模块执行流程图

服务器端模块执行流程如图 6 所示，步骤如下：

步骤 1：初始化服务器端的 socket 地址结构；

步骤 2：创建 socket；若成功，设置端口释放后立即就可以被再次使用转步骤 3；若失败，转步骤 6；

步骤 3：绑定 socket；若成功，转步骤 4；若失败，转步骤 6；

步骤 4：服务器开始监听；若收到客户端请求连接消息，则建立链接，转步骤 5；否则，一直监听；

步骤 5：接收客户端消息；若接收成功，则写入文件；否则，转步骤 6；

步骤 6：关闭文件和 socket。

4、抓包模块

抓包模块的实现分四个步骤进行，首先进行初始化，包括绑定本机网卡、设置网卡为混杂模式并打开信号机制；其次进行抓包，捕获数据包并等待用户输入信号；然后写入文件，将抓取的数据包统计分析，若收到信号，则建立线程，第一个线程将 TCP 的相关数据写入文件，第二个线程将 UDP 的相关数据写入文件，同时打开线程互斥锁，最后程序结束。抓包模块层次图如图 7 所示。

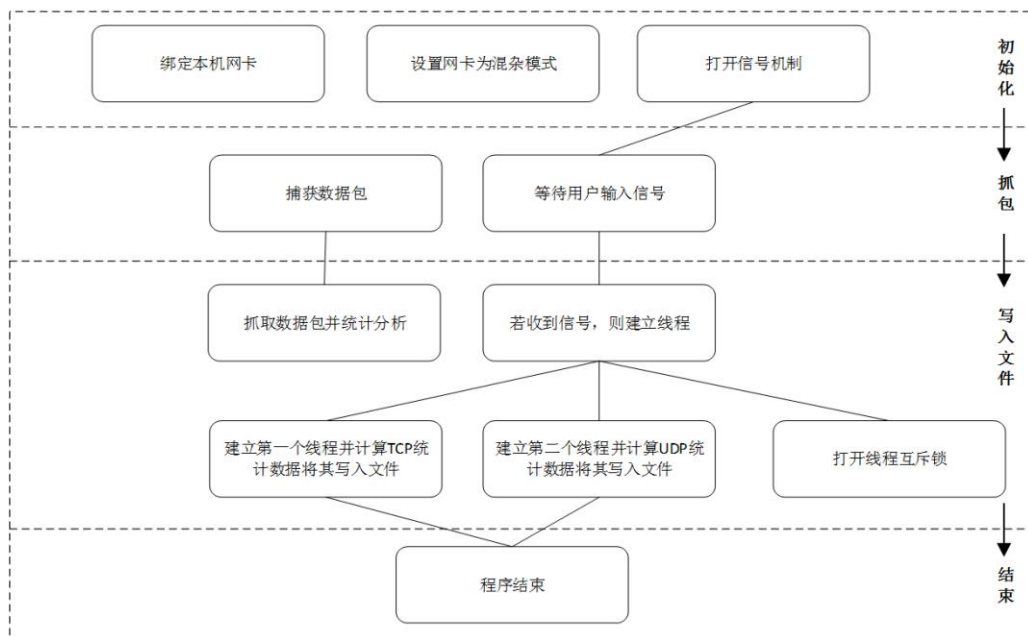


图 7 抓包模块层次图

5、主要数据结构：

主要数据结构如表 1 所示：

表 1 主要数据结构

数据结构名	解释
sniff_ethernet	以太网头
sniff_ip	IP 头
sniff_udp	UDP 头
sniff_tcp	TCP 头

6、主要函数：

主要函数如表 2 所示：

表 2 主要函数

函数名	解释
gotPacket	抓取 TCP 和 UDP 包
signalHandler	信号处理函数
thread1	将 UDP 统计数据写入文件
thread2	将 TCP 统计数据写入文件
threadCreate	建立线程
threadWait	等待线程结束

四、测试与分析

测试步骤如下：

1、首先运行 write.c 文件进行抓包，对 TCP 和 UDP 数据进行捕获，捕获的数据在命令行中显示，运行结果如图 8 所示：

```
root@ubuntu: /home/parallels/Desktop
parallels@ubuntu:~$ sudo su
[sudo] password for parallels:
root@ubuntu: /home/parallels# cd Desktop/
root@ubuntu: /home/parallels/Desktop# ls
c
chat1
client.c
Parallels Shared Folders
qt-all-opensource-src-4.3.0
qt-all-opensource-src-4.3.0.tar.gz
qt-creator-linux-x86_64-opensource-2.5.2.bin
qt-x11-opensource-src-4.3.0
qt-x11-opensource-src-4.3.0.tar.gz
server.c
Untitled Document
write.c
root@ubuntu: /home/parallels/Desktop# gcc -o write write.c -lpcap -lpthread
root@ubuntu: /home/parallels/Desktop# ./write
1[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:126 udp->udp_sum:9059
2[UDP]udp->sport:35798 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
3[UDP]udp->sport:123 udp->dport:35798 udp->udp_length:56 udp->udp_sum:11085
4[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:126 udp->udp_sum:9059
5[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:53 udp->udp_sum:8986
6[UDP]udp->sport:44564 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
7[UDP]udp->sport:123 udp->dport:44564 udp->udp_length:56 udp->udp_sum:5670
8[UDP]udp->sport:41035 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
9[UDP]udp->sport:123 udp->dport:41035 udp->udp_length:56 udp->udp_sum:27076
10[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:185 udp->udp_sum:52854
11[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:126 udp->udp_sum:9059
12[UDP]udp->sport:59711 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
13[UDP]udp->sport:59711 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
14[UDP]udp->sport:123 udp->dport:59711 udp->udp_length:56 udp->udp_sum:39738
15[UDP]udp->sport:33798 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
16[UDP]udp->sport:123 udp->dport:33798 udp->udp_length:56 udp->udp_sum:55861
17[UDP]udp->sport:42743 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
18[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:126 udp->udp_sum:9059
19[UDP]udp->sport:42743 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
20[UDP]udp->sport:45373 udp->dport:53 udp->udp_length:42 udp->udp_sum:33767
21[UDP]udp->sport:45373 udp->dport:53 udp->udp_length:42 udp->udp_sum:33767
22[UDP]udp->sport:53 udp->dport:45373 udp->udp_length:103 udp->udp_sum:59823
23[UDP]udp->sport:53 udp->dport:45373 udp->udp_length:74 udp->udp_sum:54843
24[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:53 udp->udp_sum:8986
25[UDP]udp->sport:45373 udp->dport:53 udp->udp_length:50 udp->udp_sum:33775
26[UDP]udp->sport:45373 udp->dport:53 udp->udp_length:50 udp->udp_sum:33775
27[UDP]udp->sport:53 udp->dport:45373 udp->udp_length:163 udp->udp_sum:48469
28[UDP]udp->sport:53 udp->dport:45373 udp->udp_length:189 udp->udp_sum:24834
29[TCP]tcp->th_sport:57698 tcp->th_dport:80
30[TCP]tcp->th_sport:57700 tcp->th_dport:80
```

图 8 运行 write.c 文件结果

2、当用户按下 Ctrl+C 键，代码立即结束运行，如图 9 所示：

```
455[TCP]tcp->th_sport:59330 tcp->th_dport:443
456[TCP]tcp->th_sport:443 tcp->th_dport:59330
457[TCP]tcp->th_sport:443 tcp->th_dport:59330
458[TCP]tcp->th_sport:59334 tcp->th_dport:443
459[TCP]tcp->th_sport:59334 tcp->th_dport:443
460[TCP]tcp->th_sport:443 tcp->th_dport:59334
461[TCP]tcp->th_sport:443 tcp->th_dport:59334
462[TCP]tcp->th_sport:443 tcp->th_dport:59330
463[TCP]tcp->th_sport:59330 tcp->th_dport:443
464[TCP]tcp->th_sport:443 tcp->th_dport:59332
465[TCP]tcp->th_sport:59332 tcp->th_dport:443
466[TCP]tcp->th_sport:443 tcp->th_dport:59334
467[TCP]tcp->th_sport:59334 tcp->th_dport:443
468[TCP]tcp->th_sport:57698 tcp->th_dport:80
469[TCP]tcp->th_sport:80 tcp->th_dport:57698
470[TCP]tcp->th_sport:47484 tcp->th_dport:80
471[TCP]tcp->th_sport:80 tcp->th_dport:47484
472[UDP]udp->sport:45373 udp->dport:53 udp->udp_length:46 udp->udp_sum:33771
473[UDP]udp->sport:53 udp->dport:45373 udp->udp_length:94 udp->udp_sum:50057
474[UDP]udp->sport:45373 udp->dport:53 udp->udp_length:46 udp->udp_sum:33771
475[TCP]tcp->th_sport:46244 tcp->th_dport:443
476[TCP]tcp->th_sport:443 tcp->th_dport:46244
477[TCP]tcp->th_sport:46260 tcp->th_dport:443
478[TCP]tcp->th_sport:443 tcp->th_dport:46260
479[TCP]tcp->th_sport:47504 tcp->th_dport:80
480[TCP]tcp->th_sport:80 tcp->th_dport:47504
481[TCP]tcp->th_sport:48896 tcp->th_dport:443
482[TCP]tcp->th_sport:443 tcp->th_dport:48896
483[TCP]tcp->th_sport:44632 tcp->th_dport:443
484[TCP]tcp->th_sport:443 tcp->th_dport:44632
485[TCP]tcp->th_sport:57698 tcp->th_dport:80
486[TCP]tcp->th_sport:80 tcp->th_dport:57698
^CThread1 created
Thread2 created
N_rev_P1:359
N_wrt_P1:359
N_rev_P2:127
N_wrt_P2:127
Capture complete.
root@ubuntu:/home/parallels/Desktop#
```

图 9 结束运行 write.c 文件结果

3、抓包模块向 Report.txt 文件写入数据，如图 10 所示：

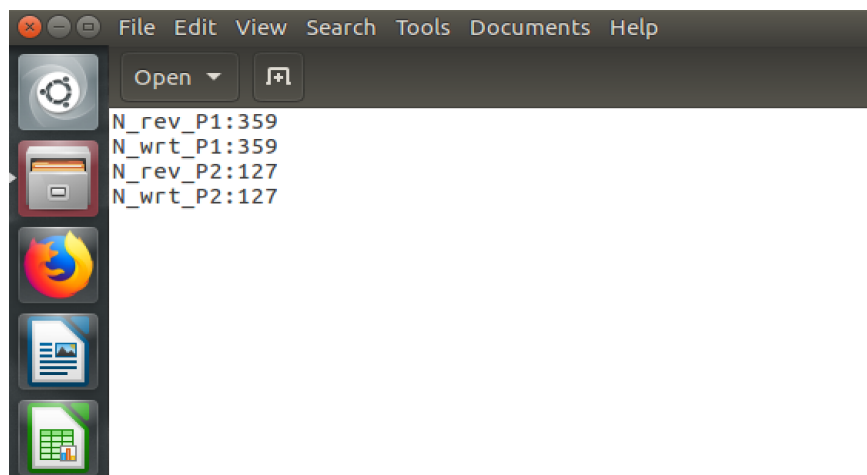


图 10 Report.txt 文件内容

4、同时抓包模块也向 Packet 文件写入数据信息，如图 11 所示：

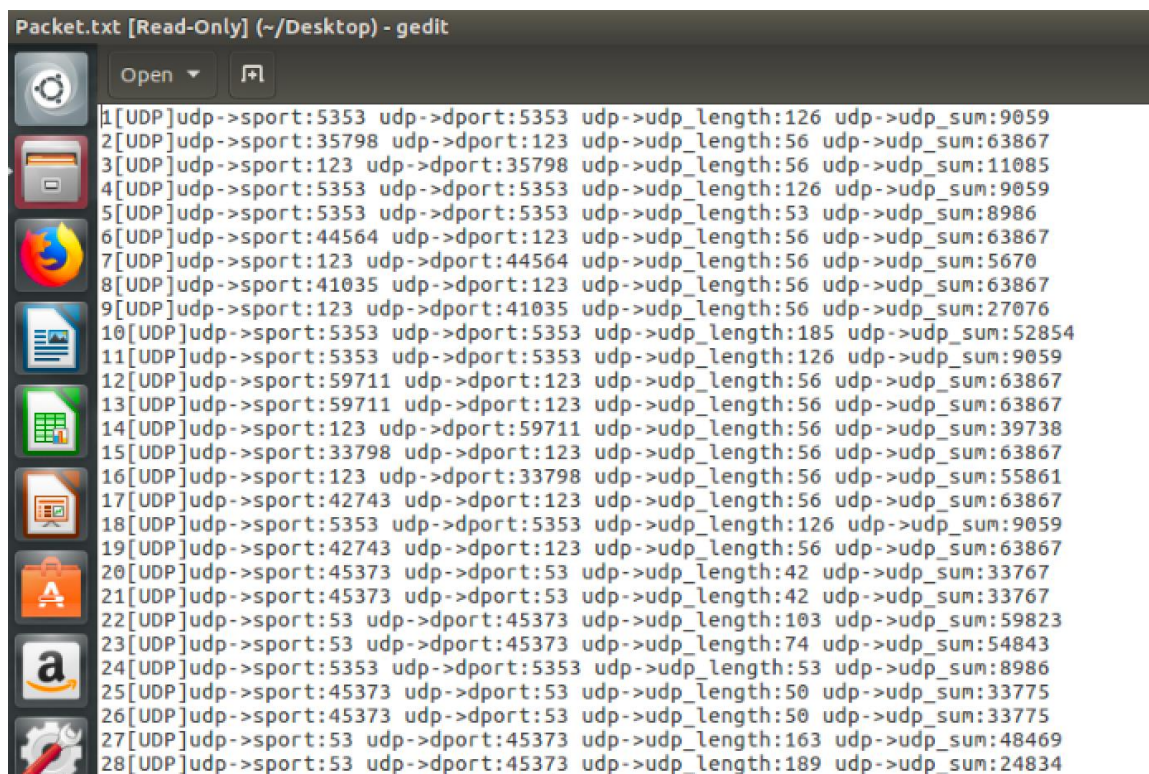


图 11 Packet.txt 文件内容

5、打开服务器，运行 server.c 文件，弹出控制对话框界面，如图 12 所示：

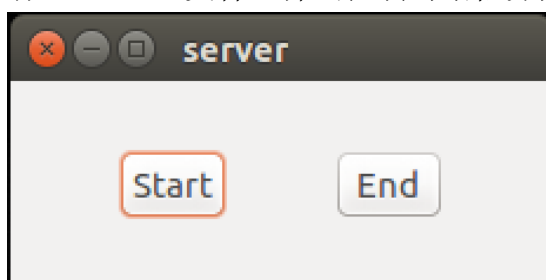
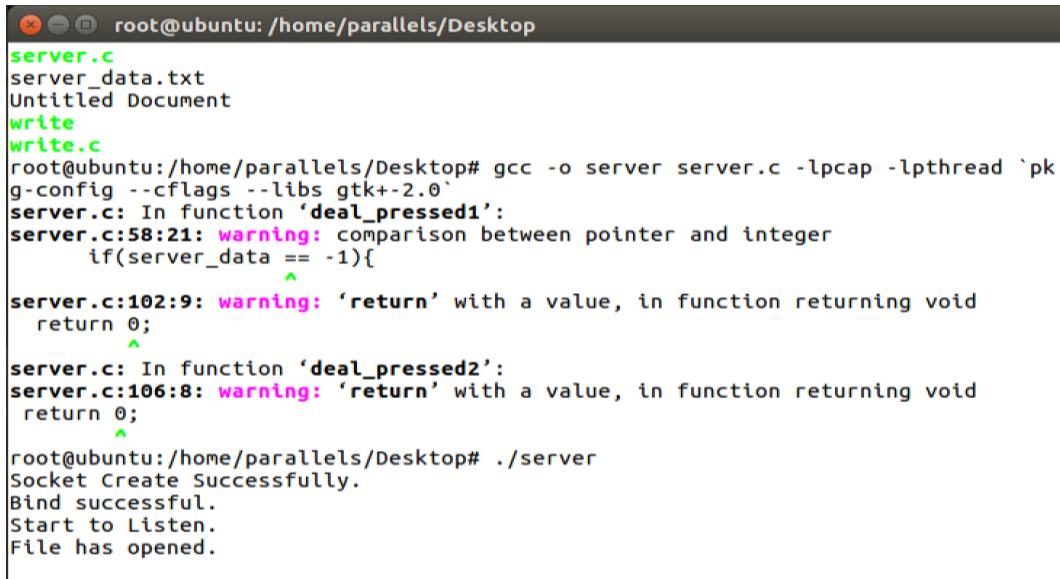


图 12 控制对话框界面

6、命令行显示开始监听，如图 13 所示：



```
root@ubuntu: /home/parallels/Desktop
server.c
server_data.txt
Untitled Document
write
write.c
root@ubuntu:/home/parallels/Desktop# gcc -o server server.c -lpcap -lpthread `pk
g-config --cflags --libs gtk+-2.0`
server.c: In function 'deal_pressed1':
server.c:58:21: warning: comparison between pointer and integer
    if(server_data == -1){
                    ^
server.c:102:9: warning: 'return' with a value, in function returning void
    return 0;
    ^
server.c: In function 'deal_pressed2':
server.c:106:8: warning: 'return' with a value, in function returning void
    return 0;
    ^
root@ubuntu:/home/parallels/Desktop# ./server
Socket Create Successfully.
Bind successful.
Start to Listen.
File has opened.
```

图 13 运行 server.c 文件

7、新建终端，运行 client.c 客户端文件，生成 client 对话框界面，如图 14 所示：

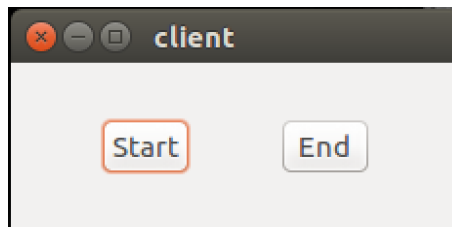
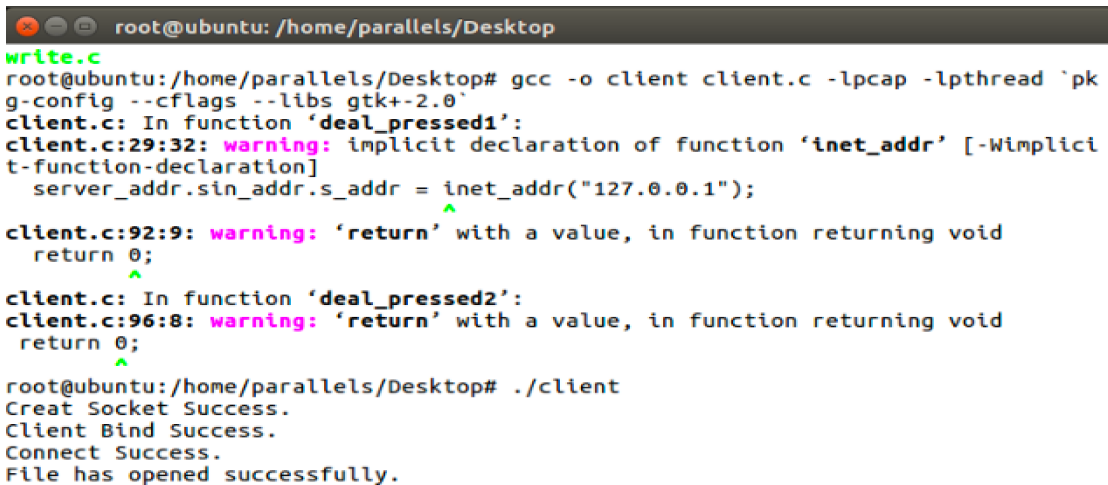


图 14 client 对话框界面

8、客户端发起连接，若连接成功，将“Connect Success”现实在屏幕上，如图 16 所示：



```
root@ubuntu: /home/parallels/Desktop
write.c
root@ubuntu:/home/parallels/Desktop# gcc -o client client.c -lpcap -lpthread `pk
g-config --cflags --libs gtk+-2.0`
client.c: In function 'deal_pressed1':
client.c:29:32: warning: implicit declaration of function 'inet_addr' [-Wimplici
t-function-declaration]
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
                                ^
client.c:92:9: warning: 'return' with a value, in function returning void
    return 0;
    ^
client.c: In function 'deal_pressed2':
client.c:96:8: warning: 'return' with a value, in function returning void
    return 0;
    ^
root@ubuntu:/home/parallels/Desktop# ./client
Creat Socket Success.
Client Bind Success.
Connect Success.
File has opened successfully.
```

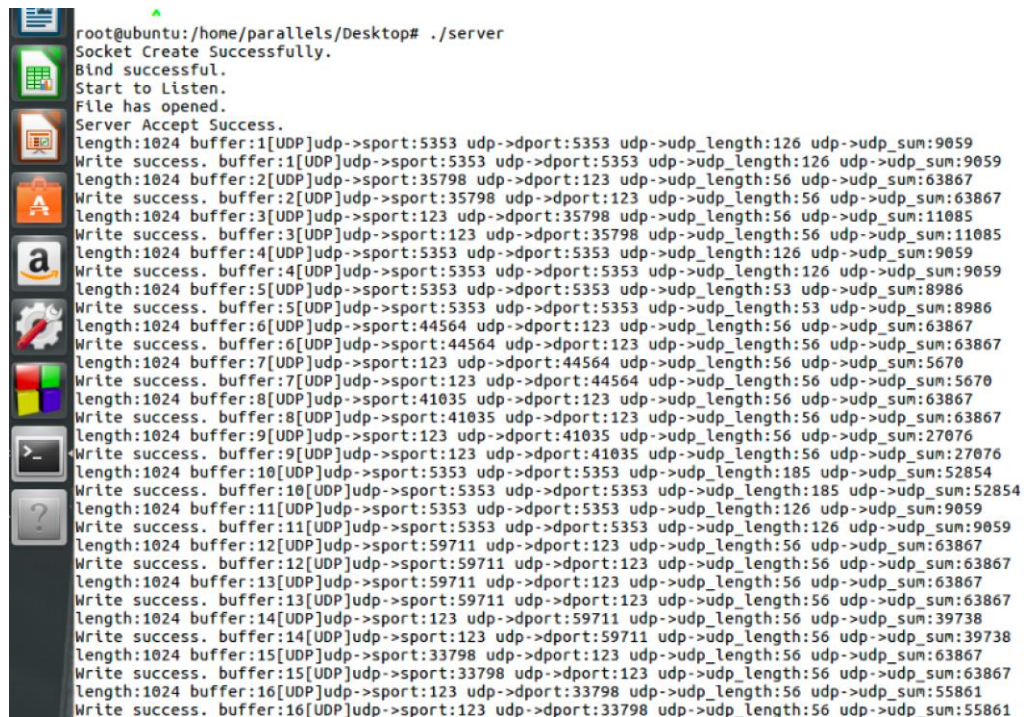
图 15 客户端开始连接

9、用户点击“Start”按钮，开始传输数据，如图 16 所示：

```
root@ubuntu:/home/parallels/Desktop# ./client
Creat Socket Success.
Client Bind Success.
Connect Success.
File has opened successfully.
buffer:1[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:126 udp->udp_sum:9059
length:1024
Send success.buffer:2[UDP]udp->sport:35798 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
length:1024
Send success.buffer:3[UDP]udp->sport:123 udp->dport:35798 udp->udp_length:56 udp->udp_sum:11085
length:1024
Send success.buffer:4[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:126 udp->udp_sum:9059
length:1024
Send success.buffer:5[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:53 udp->udp_sum:8986
length:1024
Send success.buffer:6[UDP]udp->sport:44564 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
length:1024
Send success.buffer:7[UDP]udp->sport:123 udp->dport:44564 udp->udp_length:56 udp->udp_sum:5670
length:1024
Send success.buffer:8[UDP]udp->sport:41035 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
length:1024
```

图 16 客户端传输数据

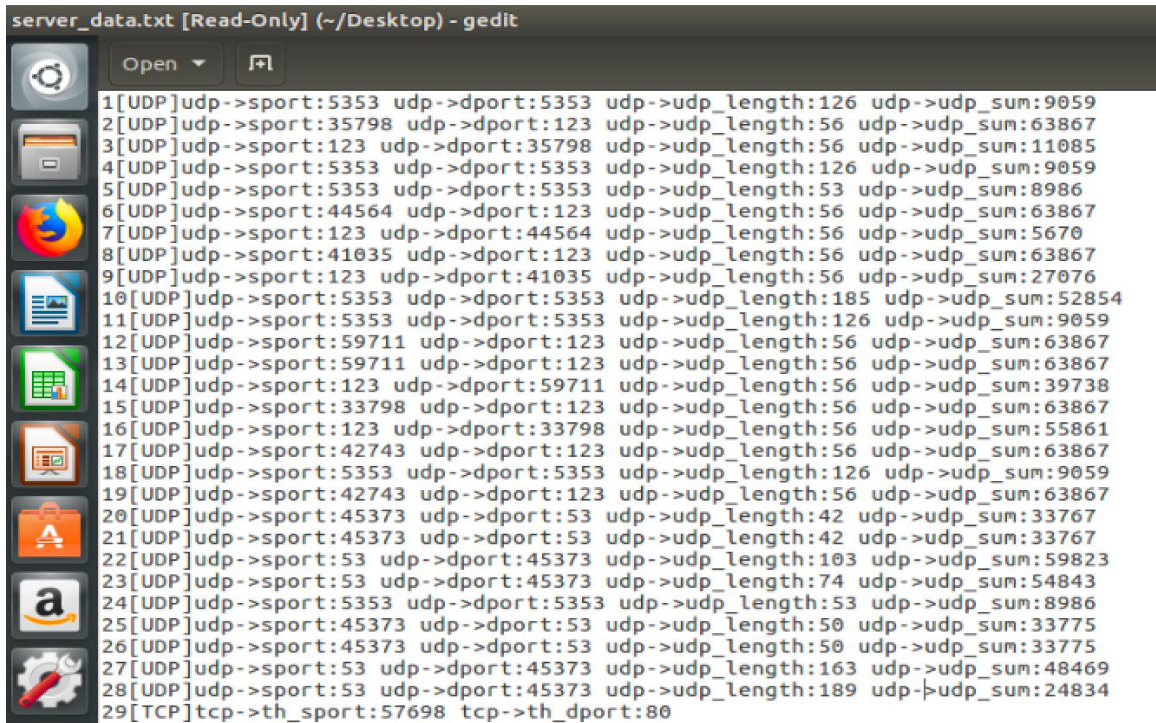
10、同时，服务器将接收到的数据在屏幕上显示，



```
root@ubuntu:/home/parallels/Desktop# ./server
Socket Create Successfully.
Bind successful.
Start to Listen.
File has opened.
Server Accept Success.
length:1024 buffer:1[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:126 udp->udp_sum:9059
Write success. buffer:1[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:126 udp->udp_sum:9059
length:1024 buffer:2[UDP]udp->sport:35798 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
Write success. buffer:2[UDP]udp->sport:35798 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
length:1024 buffer:3[UDP]udp->sport:123 udp->dport:35798 udp->udp_length:56 udp->udp_sum:11085
Write success. buffer:3[UDP]udp->sport:123 udp->dport:35798 udp->udp_length:56 udp->udp_sum:11085
length:1024 buffer:4[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:126 udp->udp_sum:9059
Write success. buffer:4[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:126 udp->udp_sum:9059
length:1024 buffer:5[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:53 udp->udp_sum:8986
Write success. buffer:5[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:53 udp->udp_sum:8986
length:1024 buffer:6[UDP]udp->sport:44564 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
Write success. buffer:6[UDP]udp->sport:44564 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
length:1024 buffer:7[UDP]udp->sport:123 udp->dport:44564 udp->udp_length:56 udp->udp_sum:5670
Write success. buffer:7[UDP]udp->sport:123 udp->dport:44564 udp->udp_length:56 udp->udp_sum:5670
length:1024 buffer:8[UDP]udp->sport:41035 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
Write success. buffer:8[UDP]udp->sport:41035 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
length:1024 buffer:9[UDP]udp->sport:123 udp->dport:41035 udp->udp_length:56 udp->udp_sum:27076
Write success. buffer:9[UDP]udp->sport:123 udp->dport:41035 udp->udp_length:56 udp->udp_sum:27076
length:1024 buffer:10[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:185 udp->udp_sum:52854
Write success. buffer:10[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:185 udp->udp_sum:52854
length:1024 buffer:11[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:126 udp->udp_sum:9059
Write success. buffer:11[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:126 udp->udp_sum:9059
length:1024 buffer:12[UDP]udp->sport:59711 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
Write success. buffer:12[UDP]udp->sport:59711 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
length:1024 buffer:13[UDP]udp->sport:59711 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
Write success. buffer:13[UDP]udp->sport:59711 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
length:1024 buffer:14[UDP]udp->sport:123 udp->dport:59711 udp->udp_length:56 udp->udp_sum:39738
Write success. buffer:14[UDP]udp->sport:123 udp->dport:59711 udp->udp_length:56 udp->udp_sum:39738
length:1024 buffer:15[UDP]udp->sport:33798 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
Write success. buffer:15[UDP]udp->sport:33798 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
length:1024 buffer:16[UDP]udp->sport:123 udp->dport:33798 udp->udp_length:56 udp->udp_sum:55861
Write success. buffer:16[UDP]udp->sport:123 udp->dport:33798 udp->udp_length:56 udp->udp_sum:55861
```

图 17 服务器接收数据

11、服务器将接收到的数据写入 server_data.txt 文件，如图 18 所示：



```
server_data.txt [Read-Only] (~/Desktop) - gedit
1[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:126 udp->udp_sum:9059
2[UDP]udp->sport:35798 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
3[UDP]udp->sport:123 udp->dport:35798 udp->udp_length:56 udp->udp_sum:11085
4[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:126 udp->udp_sum:9059
5[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:53 udp->udp_sum:8986
6[UDP]udp->sport:44564 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
7[UDP]udp->sport:123 udp->dport:44564 udp->udp_length:56 udp->udp_sum:5670
8[UDP]udp->sport:41035 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
9[UDP]udp->sport:123 udp->dport:41035 udp->udp_length:56 udp->udp_sum:27076
10[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:185 udp->udp_sum:52854
11[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:126 udp->udp_sum:9059
12[UDP]udp->sport:59711 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
13[UDP]udp->sport:59711 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
14[UDP]udp->sport:123 udp->dport:59711 udp->udp_length:56 udp->udp_sum:39738
15[UDP]udp->sport:33798 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
16[UDP]udp->sport:123 udp->dport:33798 udp->udp_length:56 udp->udp_sum:55861
17[UDP]udp->sport:42743 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
18[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:126 udp->udp_sum:9059
19[UDP]udp->sport:42743 udp->dport:123 udp->udp_length:56 udp->udp_sum:63867
20[UDP]udp->sport:45373 udp->dport:53 udp->udp_length:42 udp->udp_sum:33767
21[UDP]udp->sport:45373 udp->dport:53 udp->udp_length:42 udp->udp_sum:33767
22[UDP]udp->sport:53 udp->dport:45373 udp->udp_length:103 udp->udp_sum:59823
23[UDP]udp->sport:53 udp->dport:45373 udp->udp_length:74 udp->udp_sum:54843
24[UDP]udp->sport:5353 udp->dport:5353 udp->udp_length:53 udp->udp_sum:8986
25[UDP]udp->sport:45373 udp->dport:53 udp->udp_length:50 udp->udp_sum:33775
26[UDP]udp->sport:45373 udp->dport:53 udp->udp_length:50 udp->udp_sum:33775
27[UDP]udp->sport:53 udp->dport:45373 udp->udp_length:163 udp->udp_sum:48469
28[UDP]udp->sport:53 udp->dport:45373 udp->udp_length:189 udp->udp_sum:24834
29[TCP]tcp->th_sport:57698 tcp->th_dport:80
```

图 18 server_data.txt 文件内容

五、总结

本实验实现了 Linux 系统下 C 语言的抓包系统，本部分从“实验遇到的问题及解决方法”和“讨论优缺点以及改进方法”两个方面进行总结。

1、问题及解决方法

本次作业，从系统安装配置到 Windows 和 Linux 系统之间的文件传输等等过程都遇到了一些问题，花费了一部分时间，但是也积累了很多宝贵的经验。程序设计也遇到了诸多问题，这些问题不仅仅训练我调试错误的能力，更多反映出来编程时的逻辑思路或者细节欠缺的地方，只有意识到了这一点，才能在今后避免同类问题，做一个“合格”的程序员。遇到的主要问题如下：

(1) 缺少互斥量问题。一开始设计了两个线程工作，但是发现每次运行程序都是只有线程一在运行，线程二没有机会运行，抓包就结束了。因此，通过查阅资料，我添加了线程互斥量解决了此问题。

(2) “段错误”问题。花费时间最长的问题就是著名的“段错误”，借此机会系统地学习了一下，并对 Linux 环境下的“段错误”做个总结。“段错误”是指访问的内存超出了系统给这个程序所设定的内存空间，例如访问了不存在的内存地址、访问了系统保护的内存地址、访问了只读的内存地址等等。“段错误”产生的原因主要有以下几点：访问不存在的内存地址；访问系统保护的内存地址；访问只读的内存地址、栈溢出等。知道大概有这几种错误之后，我用 dmesg 命令查看发生“段错误”的程序名称、引起“段错误”发生的内存地址、指令指针地址、堆栈指针地址、

错误代码、错误原因等。但是显示出来的信息过于繁杂，虽然找到了错位的位置，以及错误原因，但是并不知道错误的具体原因，仍然无从下手；最后通过设置程序断点以及结合经验发现，错误产生的原因是由于声明两个不同类型的变量时出现了重复，最后解决了这个问题。

(3) 端口占用问题。每次程序运行结束之后，若想短时间内再次运行程序，都会出现“Address already in use”的错误信息，对测试接下来进行了带来麻烦，我怀疑是申请的 socket 资源没有完全释放所导致的。因此通过查阅资料，我发现 `setsockopt(serverSock, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt));` 参数 `SO_REUSEADDR` 可以让资源立即释放，而不需要等待两分钟。

(4) 抓包为 IPX 包问题。设置网卡为混杂模式后，建立 IP 连接，但是收到的包无法解析，经过调试后发现，收到的是 IPX 包并不是 IP 包，因此为了更方便地接收 IP 包对 TCP 和 UDP 加以区分，我将 ip 的数据结构全部替换成 `sniff_ip`，同理将 `tcp` 和 `udp` 分别替换成 `sniff_tcp` 和 `sniff_udp`，然后再判断 `ip_p` 的参数是 `IPPROTO_TCP`、`IPPROTO_UDP` 或者其他，则可以对其进行简单处理。

(5) 多个进程连接服务器问题。一开始的思路是在两个进程完成抓包后就立即传输到服务器，但是出现了一个现象，即当一个进程传输结束后，会与服务器断开连接；若想再次传输，就要重新建立套接字。这样会消耗更多的资源和时间，尤其是当两个进程切换频繁的时候。因此我改变了思路，选择较为简便的线程，并且当停止抓包后便将文件内容读出，一次性传送给服务器。

2、讨论

(1) 界面设计：本程序的界面设计相对简单。程序编写的时候出于方便，所有可能出错的地方均用 `perror` 显示错误并退出程序，但退出程序这种方法在用户使用的时候体验是不佳的。但由于时间的原因没有进一步完善这个问题，设想的情况应该是：假如出现错误，首先在图形界面上给用户展示错误原因，其次竭力解决。例如：当客户端连接服务器时，首先提示“连接中...”；若连接失败，则继续尝试连接并计时，提示“连接失败，重新连接中...”；3 分钟后，若仍没成功，则放弃连接，提示“连接失败”。

除此之外，我认为客户端可以再添加“开始抓包”按钮和“暂停抓包”按钮，并设计线程的暂停和继续函数，可以更加方便客户使用。

(2) 功能设计：添加多个客户端连接同一服务器的功能；这时候服务器端应该使用多线程，每连接上一个客户端就给该客户端开启一个线程。监听端口的时候也要单独开一个线程，否则会阻塞主线程。但这样做会存在一个明显的缺点，就是当有 N 个客户端请求连接时，就会有 N 个线程，这样对程序的性能和计算机的性能会造成很大的影响，可以使用线程池来进行管理。使用线程池的好处，可以减少因频繁创建和销毁线程带来的开销。因此那些频繁使用且执行时间短的线程需要采用线程池来加以管理。

附录 1：参考网页

1. http://blog.csdn.net/L_yangliu/article/details/11553713
2. <http://bbs.csdn.net/topics/390411637?page=1>
3. <http://blog.csdn.net/ljianhui/article/details/10875883>
4. <https://www.cnblogs.com/railgunman/archive/2010/11/06/1870867.html>
5. <https://www.cnblogs.com/likeyiyyp/p/3670213.html>
6. <https://www.cnblogs.com/panfeng412/archive/2011/11/06/2237857.html>
7. <https://www.cnblogs.com/rosesmall/archive/2012/04/13/2445527.html>
8. <http://blog.csdn.net/u011068702/article/details/53931648>
9. <https://www.cnblogs.com/TianFang/archive/2013/01/20/2868889.html>
10. http://blog.csdn.net/piaojun_pj/article/details/6098438
11. <http://bbs.csdn.net/topics/280060121/>
12. <http://blog.csdn.net/u011068616/article/details/41819787>
13. <http://blog.csdn.net/bytxl/article/details/28230477>
14. <http://bbs.csdn.net/topics/391998869>

附录 2：使用说明



```
使用说明
=====
本程序共有3个文件组成：
1、抓包：write.c
2、服务器：server.c
3、客户端client.c
=====
运行write.c：
=====
sudo su
输入密码：*****
cd (文件位置)
gcc -o write233 write233.c -lpcap -lpthread
./write233

按Ctrl+C结束程序
=====
运行server.c：
=====
运行server.c
gcc -o server server.c
./server
输出：server_data.txt
=====
运行client.c：
=====
在重新开一个终端（右击 -> 新终端）
运行client 方法同上
输出文件Packet.txt和Report.txt
=====
其他参考命令：
chmod +x ./
chmod +777 ./
=====
```