# Linux平台的Socket中级应用-TCP文件传输

实验名称：Linux平台的Socket中级应用-TCP文件传输

学时安排：2课时 指导老师：李赞

实验类别：验证型、设计型 实验要求：1人1组

学号： 姓名： 班级：

## 一、实验目的和任务

1. 复习ubuntu的IP配置知识。

2. 本实验要求掌握TCP通讯流程。

3. 了解TCP编程的基本方法，拼包现象及文件传输的实现。

## 二、实验设备介绍

1.软件需求： win10操作系统， ubuntu18。

## 三、注意事项和要求

1.启动虚拟机前应首先使用"虚拟网络编辑器"配置要桥接的网卡。

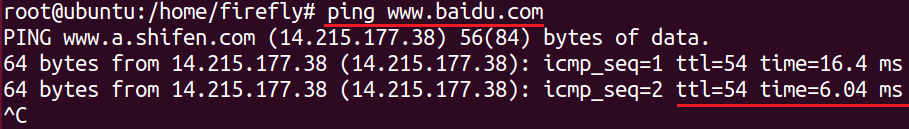
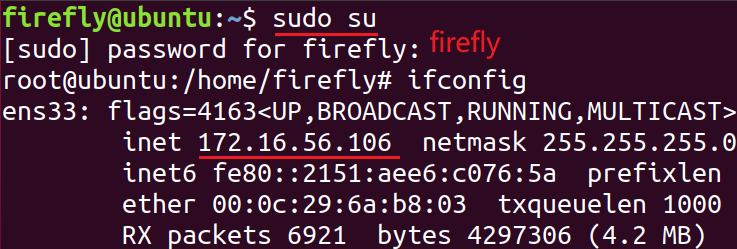
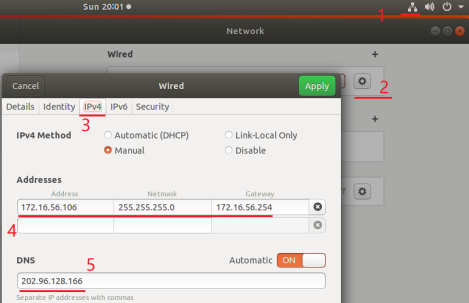
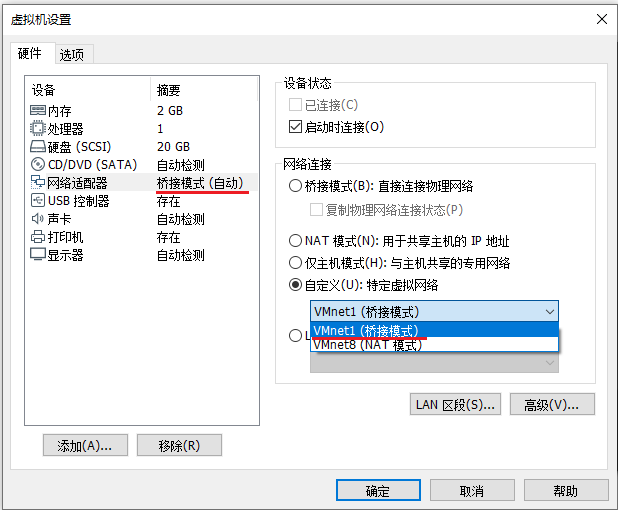
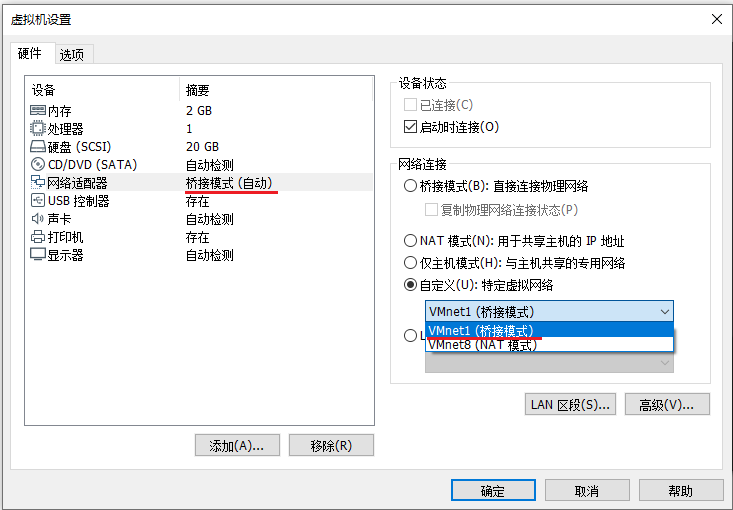
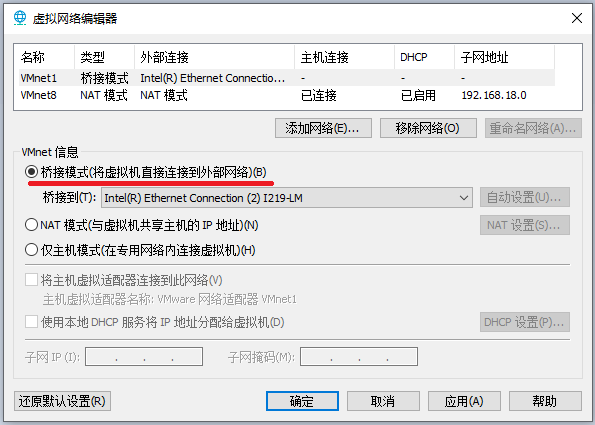
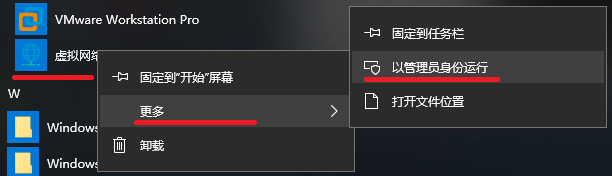
2.TCP通信流程中，send和recv方法数据缓冲区存在不对应现象。

3.程序运行结果截屏后提交到泛雅平台。

## 四、实验内容和步骤

### 4.1虚拟机网络配置

实验室B701机房的机器设置了开机还原功能其中C盘和D盘在每次开机后都会执行还原，E盘数据是不执行还原的，为了减少虚拟机的网络参数重复配置，首先检查E盘是否已经有了ubuntu18的虚拟机文件。如果E盘没有虚拟机文件，则要求同学们将虚拟机文件从D盘拷贝到E盘，即将D:\VMware\_machine\ubuntu18.04 拷贝到E:\VMachine\ubuntu18.04。下面给出了虚拟机网络配置步骤：1.使用虚拟网络编辑器设置要桥接的网卡参数，2.编辑虚拟机的网络编辑器，设置虚拟机所用的桥接网卡，3.配置ubuntu的网络IP4地址和dns，4.检测ubuntu的联网是否成功。下面是参考的操作界面。



虚拟机在配置IP的时候，注意不要与其它主机的IP地址相同。虚拟机联网正常后，即可运行联网程序了。

### 4.2文件发送端程序

发送文件的程序端会执行监听，在运行时要先于接收端程序启动。在ubuntu中使用gcc进行编译。

//《TCP/IP网络编程》page121 fsend.c 030.c 文件传输发送端

//本程序执行socket监听，文件发送操作

//030.c 文件传输发送端

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <string.h>

#include <netdb.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <sys/socket.h>

#include <sys/file.h>

#include <sys/stat.h>

#define SERVPORT 1300

#define MAXDATASIZE 1000

#define FILE\_NAME\_LEN 30

int main(int argc,char \*argv[])

{

int sockfd,sendbytes;

char buf[MAXDATASIZE];

char file\_name[FILE\_NAME\_LEN];

struct hostent \*host;

struct sockaddr\_in serv\_addr;

int fd = 0;

int totalbytes = 0;

int readbytes = 0;

printf("This program is tcp client, prepare to connect and send file.\n");

if(argc < 2){

fprintf(stderr,"Please enter the server's hostname!\n");

exit(1);

}

if((host=gethostbyname(argv[1]))==NULL){

perror("gethostbyname");

exit(1);

}

if((sockfd=socket(AF\_INET,SOCK\_STREAM,0))==-1){

perror("socket");

exit(1);

}

serv\_addr.sin\_family=AF\_INET;

serv\_addr.sin\_port=htons(SERVPORT);

serv\_addr.sin\_addr=\*((struct in\_addr \*)host->h\_addr);

bzero(&(serv\_addr.sin\_zero),8);

if(connect(sockfd,(struct sockaddr \*)&serv\_addr,\

sizeof(struct sockaddr))==-1){

perror("connect");

exit(1);

}

printf("please input the file you want to open:\n");

scanf("%s",file\_name);

printf("filename is %s\n",file\_name);

fd=open(file\_name, O\_RDONLY, 0666);

if(fd < 0){

perror("open");

exit(1);

}

printf("press s to send file\n");

getchar();

while((readbytes = read(fd, buf,MAXDATASIZE)) )

{

if((sendbytes=send(sockfd, buf, readbytes,0))==-1){

perror("send");

exit(1);

}

totalbytes += sendbytes;

}

//shutdown read and write of socket

shutdown(sockfd,2);

printf("sent %d bytes ..\n", totalbytes);

printf("file sending finished.\n", totalbytes);

close(sockfd);

return 0;

}

使用gcc –o 030 030.c编译程序。

### 4.3文件接收端程序

使用gcc –o 031 031.c编译程序。

//031.c 文件传输接收端

#include <unistd.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <string.h>

#include <unistd.h>

#include <netinet/in.h>

#include <sys/file.h>

#include <sys/types.h>

#include <sys/stat.h>

#define SERVPORT 1300

#define BACKLOG 10

#define MAX\_CONNECTED\_NO 10

#define MAXDATASIZE 50

int main()

{

struct sockaddr\_in server\_sockaddr,client\_sockaddr;

int sin\_size,recvbytes;

int sockfd,client\_fd;

time\_t currentTime;

char timebuffer[MAXDATASIZE+1];

int fd = 0;

int nwrite = 0;

char buf[MAXDATASIZE];

printf("This program is tcp server, prepare to listen and recv file.\n");

if((sockfd = socket(AF\_INET,SOCK\_STREAM,0))==-1){

perror("socket");

exit(1);

}

printf("socket create success!,sockfd=%d\n",sockfd);

server\_sockaddr.sin\_family=AF\_INET;

server\_sockaddr.sin\_port=htons(SERVPORT);

server\_sockaddr.sin\_addr.s\_addr=INADDR\_ANY;

bzero(&(server\_sockaddr.sin\_zero),8);

if(bind(sockfd,(struct sockaddr \*)&server\_sockaddr,sizeof(struct sockaddr))==-1){

perror("bind");

exit(1);

}

printf("bind success!\n");

if(listen(sockfd,BACKLOG)==-1){

perror("listen");

exit(1);

}

printf("listening....\n");

sin\_size = sizeof(struct sockaddr);

if((client\_fd=accept(sockfd,(struct sockaddr \*)&client\_sockaddr,&sin\_size))==-1){

perror("accept");

exit(1);

}

fd=open("recvfile",O\_RDWR | O\_CREAT, 0666);

if(fd < 0){

perror("open");

exit(1);

}

while (1) {

//read data from net

if((recvbytes=recv(client\_fd,buf,MAXDATASIZE,0)) <= 0){

perror("recv");

break;

}

//write bytes into file

if((nwrite=write(fd,buf,recvbytes)) < 0 ){

perror("write file");

}

}

//shutdown read and write from net.

shutdown(client\_fd,2);

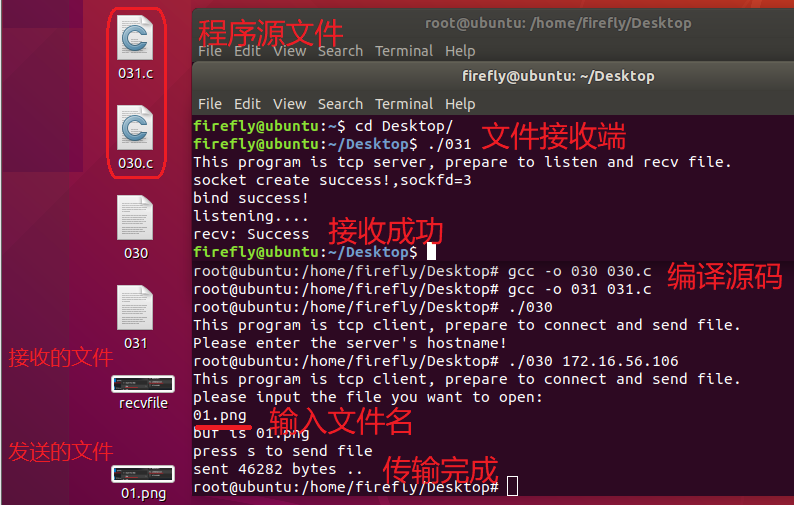
close(sockfd);

close(fd);//close file

return 0;

}

下面是在B701的运行效果截图。

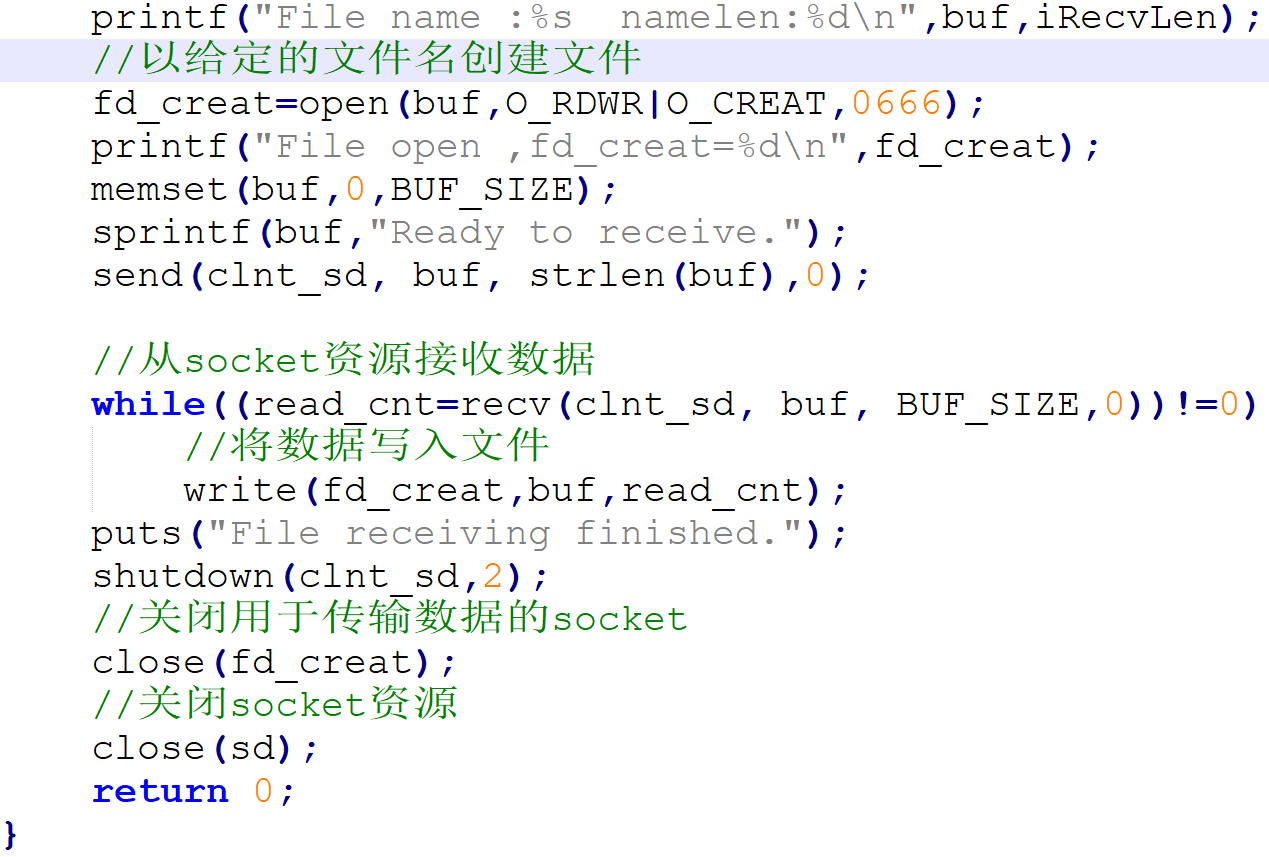


### 4.4 支持文件名的文件传输程序

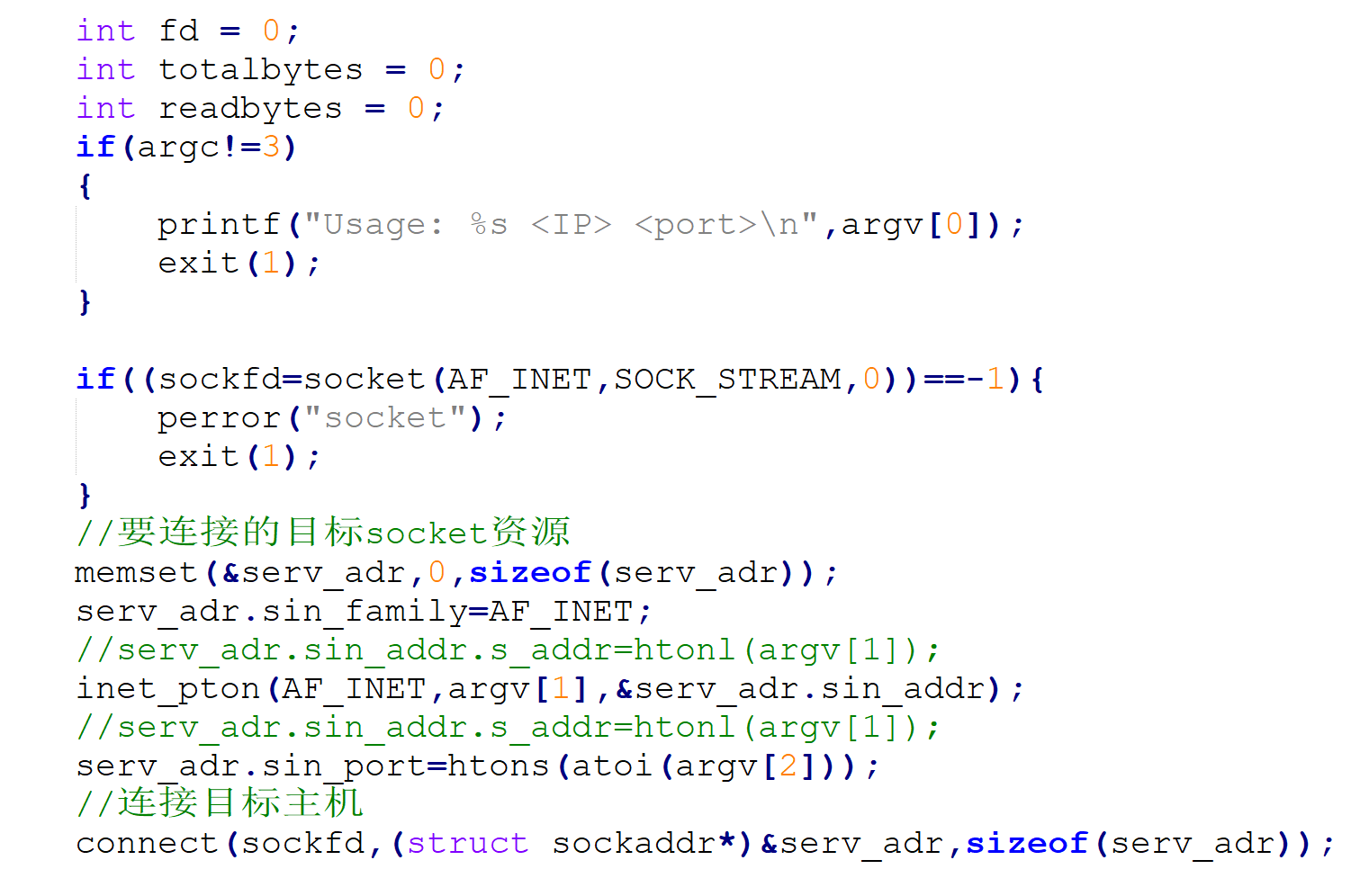
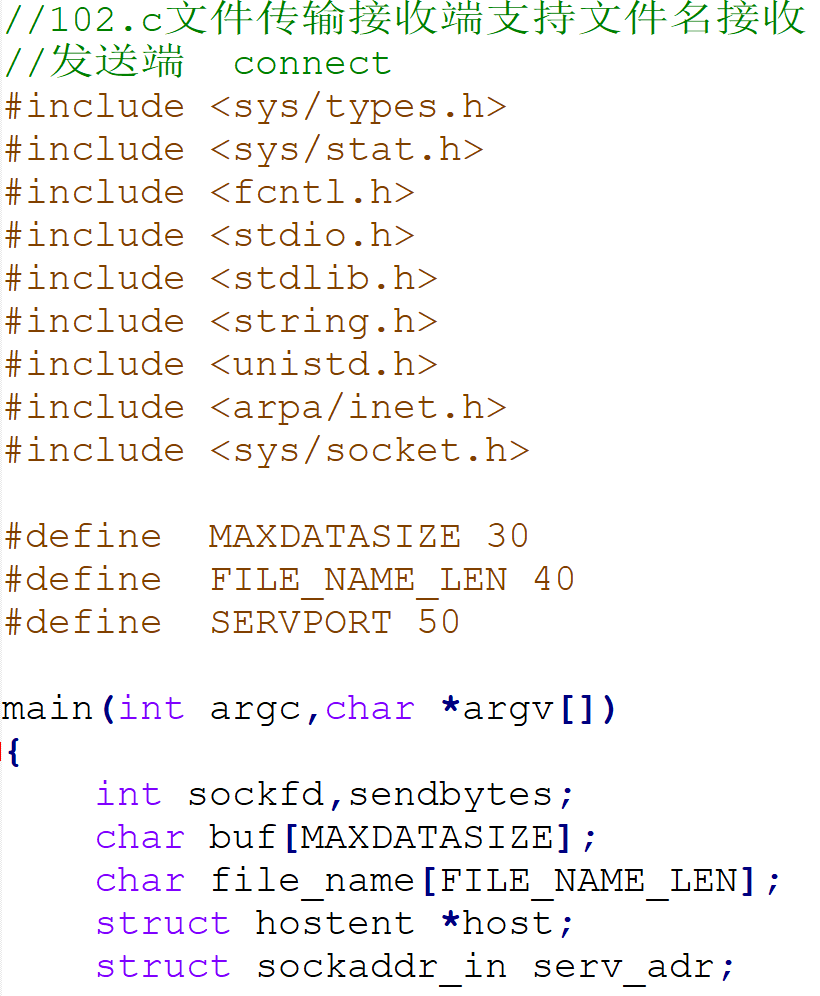
在上述程序中，接收端使用固定的文件名接收文件数据，修改程序[发送端和接收端]使得接收端使用原本的名称保存。发送文件前将文件名字符串发送给接收端，要注意的是发送端程序与接收端程序必须在两个不同的文件目录中运行，避免对同一文件执行又读又写而出现错误。

frecv文件夹中的源程序。

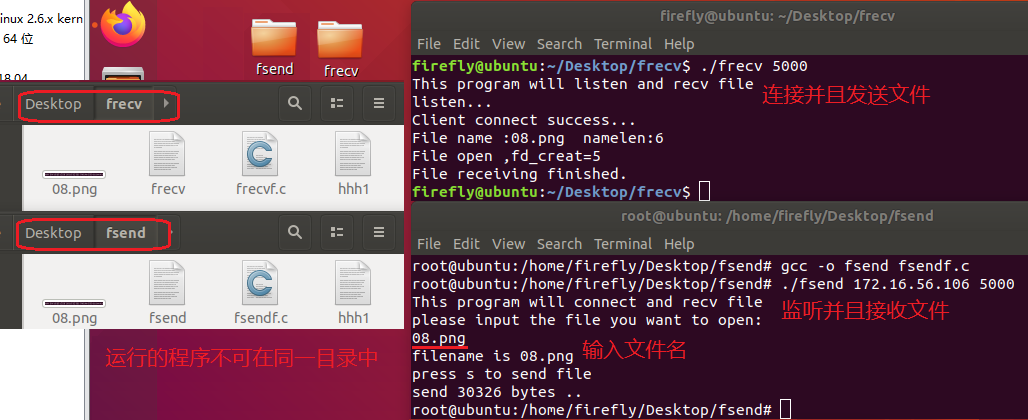




fsend文件夹中的源程序。



下面演示了文件发送前输入文件名和接收的运行效果。



### 4.5 整数的传输程序

网络提供字节的传输服务，不同的数据类型需要转换成字节再进行传输，这可通过union实现转换，这也可称为数据的序列化与反序列化。下面的程序演示了传送整数的过程。

//103.c 网络传输中整数的序列化

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <string.h>

#include <netdb.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <sys/file.h>

#include <sys/stat.h>

#define BUFSIZE 100

#define FILE\_NAME\_LEN 100

void IntToByte(int iData,char \* cData)

{

union IntChar

{

int a; //4 bytes

char b[4]; //4 byte

} InCh;

InCh.a=iData;

cData[0]=InCh.b[0];

cData[1]=InCh.b[1];

cData[2]=InCh.b[2];

cData[3]=InCh.b[3];

}

int ByteToInt(char \* cData)

{

union IntChar

{

int a; //4 bytes

char b[4]; //4 byte

} InCh;

InCh.b[0]=cData[0];

InCh.b[1]=cData[1];

InCh.b[2]=cData[2];

InCh.b[3]=cData[3];

return InCh.a;

}

int main(int argc,char \*argv[])

{

int sockfd,sendbytes;

char buf[BUFSIZE];

struct hostent \*host;

struct sockaddr\_in serv\_addr;

int fd = 0;

int iInput = 0;

printf("This program will connect and send an integer.\n");

//命令 IP PORT

if(argc!=3)

{

printf("Usage :%s <IP> <PORT>\n",argv[0]);

exit(1);

}

if((sockfd=socket(AF\_INET,SOCK\_STREAM,0))==-1){

perror("socket");

exit(1);

}

memset(&serv\_addr,0,sizeof(serv\_addr));

serv\_addr.sin\_family=AF\_INET;

//IP地址,较老的方法

//serv\_addr.sin\_addr.s\_addr=inet\_addr(argv[1]);

//新的地址转换方法

inet\_pton(AF\_INET,argv[1],&serv\_addr.sin\_addr);

//PORT

serv\_addr.sin\_port=htons(atoi(argv[2]));

if(connect(sockfd,(struct sockaddr \*)&serv\_addr,\

sizeof(struct sockaddr))==-1){

perror("connect");

exit(1);

}

printf("Server connected.\n");

printf("Input integer:");

scanf("%d",&iInput);

memset(buf,0,BUFSIZE);

IntToByte(iInput,buf);

send(sockfd,buf,4,0);

printf("Integer value of decimal %d Hex value :%0X\n", iInput,iInput);

close(sockfd);

return 0;

}

//104.c 网络传输中整数的反序列化

#include <sys/types.h>

#include <sys/socket.h>

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <string.h>

#include <unistd.h>

#include <netinet/in.h>

#include <sys/file.h>

#include <sys/types.h>

#include <sys/stat.h>

#define BACKLOG 10

#define MAX\_CONNECTED\_NO 10

#define BUFSIZE 100

void IntToByte(int iData,char \* cData)

{

union IntChar

{

int a; //4 bytes

char b[4]; //4 byte

} InCh;

InCh.a=iData;

cData[0]=InCh.b[0];

cData[1]=InCh.b[1];

cData[2]=InCh.b[2];

cData[3]=InCh.b[3];

}

int ByteToInt(char \* cData)

{

union IntChar

{

int a; //4 bytes

char b[4]; //4 byte

} InCh;

InCh.b[0]=cData[0];

InCh.b[1]=cData[1];

InCh.b[2]=cData[2];

InCh.b[3]=cData[3];

return InCh.a;

}

int main(int argc,char \*argv[])

{

struct sockaddr\_in server\_sockaddr,client\_sockaddr;

int sin\_size,recvbytes;

int listen\_sockfd,client\_fd;

time\_t currentTime;

int nwrite = 0;

char buf[BUFSIZE];

printf("This program will listen and recv an integer.\n");

if(argc!=2)

{

printf("Usage:%s <port>\n",argv[0]);

exit(1);

}

if((listen\_sockfd = socket(AF\_INET,SOCK\_STREAM,0))==-1){

perror("socket");

exit(1);

}

printf("socket success!,listen\_sockfd=%d\n",listen\_sockfd);

server\_sockaddr.sin\_family=AF\_INET;

server\_sockaddr.sin\_port=htons(atoi(argv[1]));

server\_sockaddr.sin\_addr.s\_addr=INADDR\_ANY;

bzero(&(server\_sockaddr.sin\_zero),8);

if(bind(listen\_sockfd,(struct sockaddr \*)&server\_sockaddr,sizeof(struct sockaddr))==-1){

perror("bind");

exit(1);

}

printf("bind success!\n");

if(listen(listen\_sockfd,BACKLOG)==-1){

perror("listen");

exit(1);

}

printf("listening....\n");

sin\_size = sizeof(struct sockaddr);

if((client\_fd=accept(listen\_sockfd,(struct sockaddr \*)&client\_sockaddr,&sin\_size))==-1){

perror("accept");

exit(1);

}

recvbytes=recv(client\_fd,buf,BUFSIZE,0);

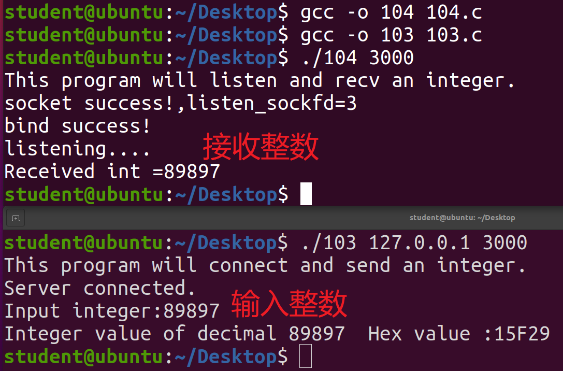
printf("Received int =%d\n",ByteToInt(buf));

shutdown(client\_fd,SHUT\_RDWR);

close(listen\_sockfd);

}

下图是程序运行效果图。



## 五、程序调试中遇到的问题和解决过程及运行结果

1.将上述每个程序先使用gcc 编译并运行，请将程序结果进行截图。

2.分析过程中遇到问题，排查程序出错的原因，并将每个程序结果截图保存，排查程序出错原因，描述解决过程。

4.调试传输文件过程文件名保持，将文件传输的源码和运行结果截图后上传到泛雅平台。