# **网络编程第三次作业**

**姓名：杨庆 学号：201822090316**

# 1.作业题目

利用多线程技术实现如下并发网络程序，要求对上课时的实现进行完善，利用线程专用数据TSD实现。服务器和客户程序分别实现如下功能：

（1）服务器等待客户连接，连接成功后显示客户地址，接着接收该客户的名字并显示，然后接收来自客户的信息（字符串），将该字符串反转，并将结果送回客户。要求服务器具有同时处理多个客户的能力。当某个客户断开连接时，打印所有该客户输入的数据。

（2）客户端首先与服务器连接，接着接收用户输入客户的名字，将该名字发送给服务器，然后接收用户输入的字符串，并发送给服务器，然后接收服务器返回的经处理后的字符串，并显示之。当用户输入Ctrl+D，终止连接并退出。

# 2.实验步骤及结果

2.1 编译

在程序所在目录下，打开终端程序，输入以下指令

gcc client.c -o client

gcc server.c -o server -lpthread

2.2运行

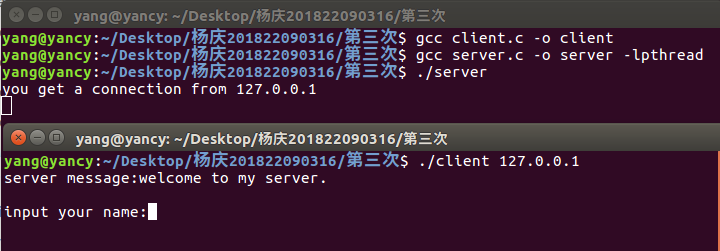
运行服务器程序，在刚才编译的窗口内输入指令：./server。

运行客户端程序。新打开终端，输入以下命令：./client 127.0.0.1

3.3.结果

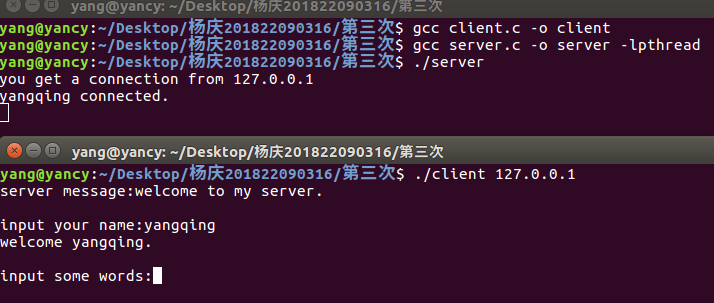
客户端：显示服务器已连接，并提示输入用户名。

服务器：显示所连接的客户端的IP地址，如下图所示。



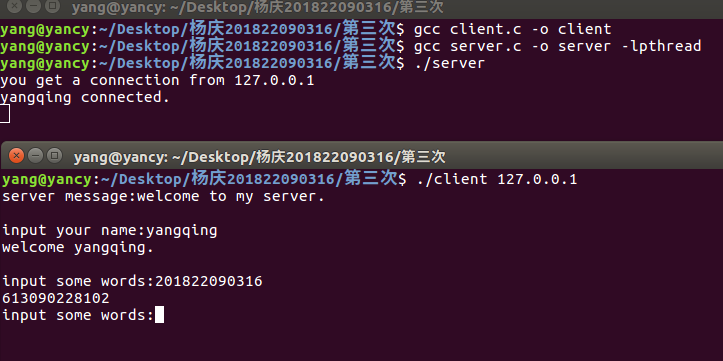
客户端：输入用户名，并发送给服务器，并提示输入字符串。

服务器：显示客户用户名，如下图所示。



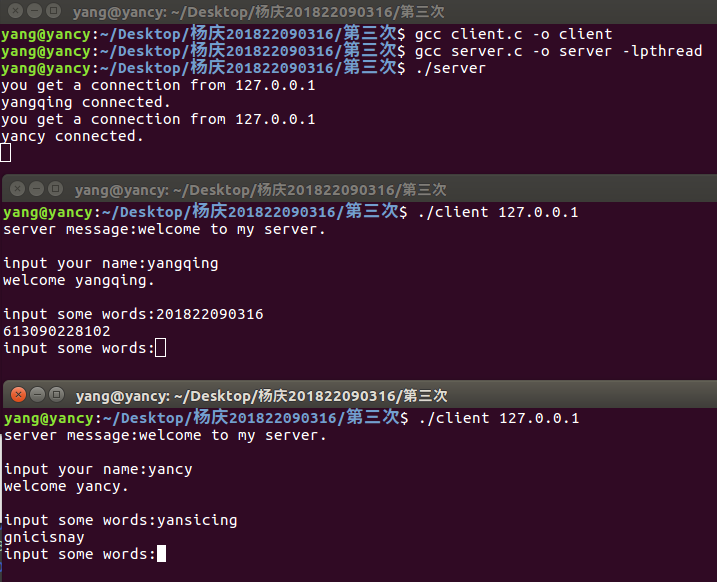
客户端：输入字符串，并发送给服务器，接收服务器返回的字符串并显示输出。

服务器：接收从客户来的字符串，并反转，再将反转结果送回客户端，如下图所示。



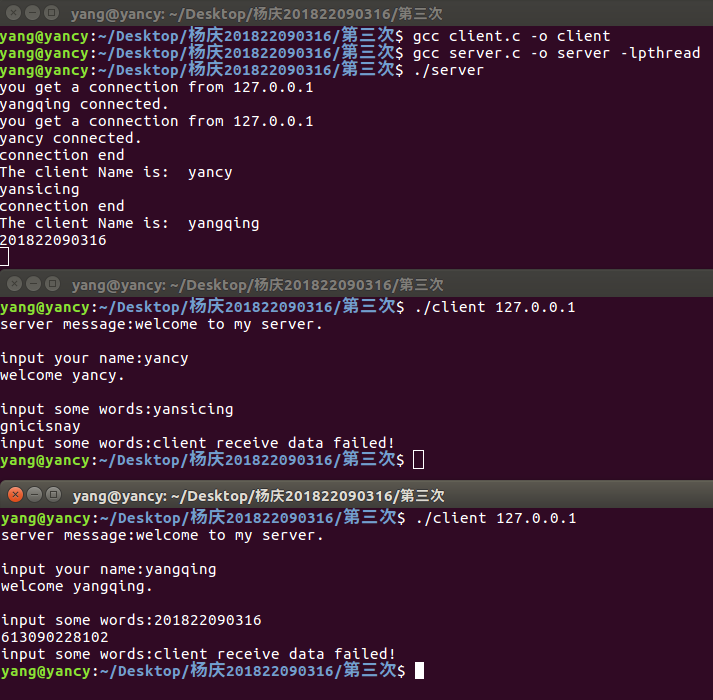
客户端：再次开启一个新客户端，重复上述步骤。

服务器：显示所连接的客户端的IP地址，客户用户名，接收从客户来的字符串，并反转，再将反转结果送回客户端，如下图所示。



客户端：输入Ctrl+D，终止其中一个客户端的连接并退出，然后再断开另一个客户端。

服务器：某个客户断开连接时，打印所有该客户输入的数据，如下图所示。



# 3.客户端源代码

#include <unistd.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <netdb.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define PORT 4321

#define MAXDATASIZE 1000

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

int fd,numbytes,scan\_state;

char buffer[MAXDATASIZE];

struct hostent \*host;

struct sockaddr\_in server;

if(argc!=2){

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

return 1;

}

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

perror("gethostbyname error.");

return 1;

}

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

perror("Create socket failed.");

return 1;

}

bzero(&server,sizeof(server));

server.sin\_family=AF\_INET;

server.sin\_port=htons(PORT);

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

if(connect(fd,(struct sockaddr \*)&server,sizeof(struct sockaddr))==-1){

perror("connect failed.");

return 1;

}

if(((numbytes=recv(fd,buffer,MAXDATASIZE,0))==-1)){

perror("recv error.");

return 1;

}

buffer[numbytes]='\0';

printf("server message:%s\n",buffer);

printf("input your name:");

if((scan\_state=scanf("%s",buffer))==-1){

printf("break\n");

close(fd);

}

numbytes=strlen(buffer);

send(fd,buffer,numbytes,0);

if(((numbytes=recv(fd,buffer,MAXDATASIZE,0))==-1)){

perror("recv error.");

return 1;

}

buffer[numbytes]='\0';

printf("%s\n",buffer);

while(1){

printf("input some words:");

if((scan\_state=scanf("%s",buffer))==-1){

printf("client receive data failed!\n");

break;

}

numbytes=strlen(buffer);

send(fd,buffer,numbytes,0);

if(((numbytes=recv(fd,buffer,MAXDATASIZE,0))==-1)){

perror("recv error.");

return 1;

}

buffer[numbytes]='\0';

printf("%s\n",buffer);

}

close(fd);

}

# 4.服务器源代码

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <pthread.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#define PORT 4321

#define BACKLOG 2

#define MAXDATASIZE 1000

void process\_client(int connectfd,struct sockaddr\_in client);

void save\_client(char\* cli\_buf,char \* recv\_buf,int recv\_len,int \* cli\_len);

void \*start\_routine(void \*arg);

struct ARG{

int connfd;

struct sockaddr\_in client;

};

void inverse\_str(char\* str,int len)

{

char temp;

char \*strLeft = str;

char \*strRight = str + len - 1;

while(strRight > strLeft)

{

temp = \*strLeft;

\*strLeft = \*strRight;

\*strRight = temp;

++strLeft;

--strRight;

}

}

int main(void)

{

int listenfd,connectfd;

struct sockaddr\_in server,client;

pthread\_t tid;

struct ARG \*arg;

socklen\_t sin\_size;

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

perror("Create socket failed.");

exit(-1);

}

int opt=SO\_REUSEADDR;

setsockopt(listenfd,SOL\_SOCKET,SO\_REUSEADDR,&opt,sizeof(opt));

bzero(&server,sizeof(server));

server.sin\_family=AF\_INET;

server.sin\_port=htons(PORT);

server.sin\_addr.s\_addr=htonl(INADDR\_ANY);

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

perror("Bind error.");

exit(-1);

}

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

perror("listen error.");

exit(-1);

}

sin\_size=sizeof(struct sockaddr\_in);

while(1){

if((connectfd=accept(listenfd,(struct sockaddr \*)&client,&sin\_size))==-1){

perror("accept error");

exit(-1);

}

arg=(struct ARG \*)malloc(sizeof(struct ARG));

arg->connfd=connectfd;

memcpy((void\*)&arg->client,&client,sizeof(client));

if(pthread\_create(&tid,NULL,start\_routine,(void\*)arg)){

perror("thread create failed\n");

exit(1);

}

}

close(listenfd);

}

void \*start\_routine(void \*arg){

struct ARG \*info;

info=(struct ARG \*)arg;

process\_client(info->connfd,info->client);

free(arg);

pthread\_exit(NULL);

}

void process\_client(int connectfd,struct sockaddr\_in client){

int numbytes,cli\_len;

char cli\_data[5000],recv\_buf[MAXDATASIZE],name[20];

printf("you get a connection from %s\n",inet\_ntoa(client.sin\_addr));

send(connectfd,"welcome to my server.\n",22,0);

if(((numbytes=recv(connectfd,recv\_buf,MAXDATASIZE,0))==-1)){

perror("recv error.");

exit(1);

}

if(numbytes>=20){

for(int i=0;i<20;i++){

name[i]=recv\_buf[i];

}

name[19]='\0';

}else{

for(int i=0;i<numbytes;i++){

name[i]=recv\_buf[i];

}

name[numbytes]='\0';

}

printf("%s connected.\n",name);

sprintf(recv\_buf,"welcome %s.\n",name);

send(connectfd,recv\_buf,strlen(recv\_buf),0);

while(1){

if(((numbytes=recv(connectfd,recv\_buf,MAXDATASIZE,0))==-1)){

perror("recv error.");

exit(1);

}else if(numbytes==0){

printf("connection end\n");

break;

}

save\_client(cli\_data,recv\_buf,numbytes,&cli\_len);

send(connectfd,recv\_buf,numbytes,0);

}

printf("The client Name is: %s\n",name);

inverse\_str(cli\_data,cli\_len);

for(int i=0;i<cli\_len;i++){

printf("%c",cli\_data[i]);

}

printf("\n");

close(connectfd);

}

void save\_client(char\* cli\_buf,char \* recv\_buf,int recv\_len,int \* cli\_len){

inverse\_str(recv\_buf,recv\_len);

for(int i=\*cli\_len-1;i>=0; i--) {

cli\_buf[i+recv\_len]=cli\_buf[i];

}

for(int i=0;i<recv\_len;i++){

cli\_buf[i]=recv\_buf[i];

}

\*cli\_len+=recv\_len;

}