# **网络编程第四次作业**

**姓名：杨庆 学号：201822090316**

# 1.作业题目

服务器利用I/O复用技术，实现同时向多个客户提供服务。要求：

（1）服务器：接收客户连接请求，并打印客户IP地址及端口号，然后接收客户发来的字符串，并打印该字符串和其来自与哪个客户。同时向客户返回该字符串。当某一客户断开连接时，要求服务器打印该客户输入的所有字符。

（2）客户端：从命令行接收服务器地址，并向服务器发起连接请求，连接成功后，从标准输入接收字符串并发送给服务器，等待服务器响应并打印接收的信息。

# 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

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。

运行多个客户端程序。新打开终端，输入命令：./ client 127.0.0.1

3.3.结果

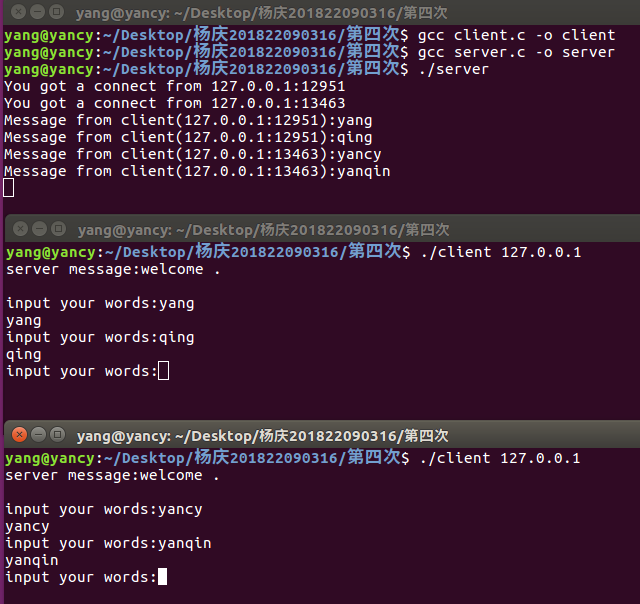
客户端：显示服务器已连接，并提示输入字符串。

服务器端：显示所连接的客户端的IP地址和端口号，如下图所示。



客户端：标准输入接收字符串并发送给服务器，等待服务器响应并打印接收的信息。

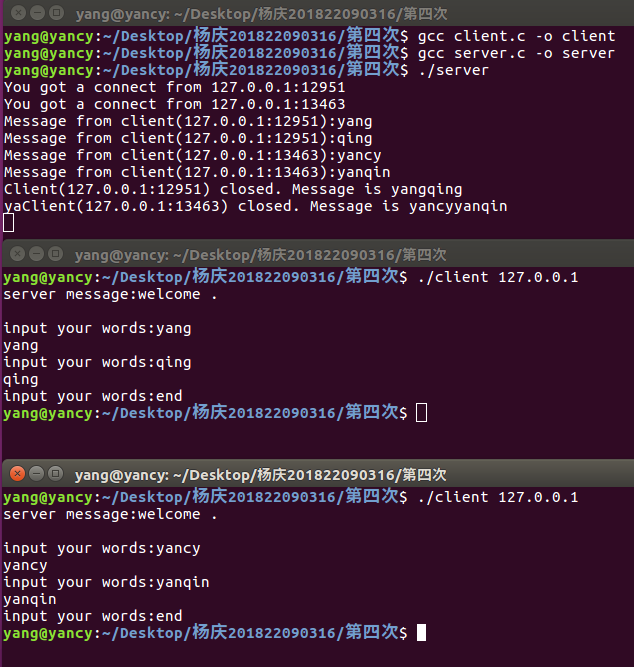
服务器：接收客户发来的字符串，并打印该字符串和其来自哪个客户。同时向客户返回该字符串，如下图所示。



客户端：断开连接。

服务器：当某一客户断开连接时，要求服务器打印该客户输入的所有字符。

断开两个客户端。



# 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 buf[MAXDATASIZE];

struct hostent \*he;

struct sockaddr\_in server;

if(argc!=2){

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

exit(-1);

}

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

perror("gethostbyname error.");

exit(1);

}

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

perror("Create socket failed.");

exit(1);

}

bzero(&server,sizeof(server));

server.sin\_family=AF\_INET;

server.sin\_port=htons(PORT);

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

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

perror("connect failed.");

exit(1);

}

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

perror("recv error.");

exit(1);

}

buf[numbytes]='\0';

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

while(1){

printf("input your words:");

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

printf("end\n");

break;

}

numbytes=strlen(buf);

send(fd,buf,numbytes,0);

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

perror("recv error.");

exit(1);

}

buf[numbytes]='\0';

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

}

close(fd);

}

# 4.服务器源代码

#include <unistd.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <pthread.h>

#define PORT 4321

#define BACKLOG 5

#define MAXDATASIZE 1000

typedef struct{

int fd;

char \*name;

struct sockaddr\_in addr;

char \*data;

int data\_len;

}CLIENT;

void process\_cli(CLIENT \*client, char \*recvbuf,int len);

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

void inverse\_str(char\* recv\_str,int str\_length){

char temp;

for(int i=0;i<(str\_length/2+str\_length%2);i++){

temp=recv\_str[i];

recv\_str[i]=recv\_str[str\_length-1-i];

recv\_str[str\_length-1-i]=temp;

}

}

int main(void){

int i,maxi,maxfd,sockfd;

int nready;

ssize\_t n;

fd\_set rset,allset;

int listenfd,connectfd;

struct sockaddr\_in server;

CLIENT client[FD\_SETSIZE];

char recvbuf[MAXDATASIZE];

int 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 orror.");

exit(-1);

}

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

maxfd=listenfd;

maxi=-1;

for(i=0;i<FD\_SETSIZE;i++)

client[i].fd=-1;

FD\_ZERO(&allset);

FD\_SET(listenfd,&allset);

while(1){

struct sockaddr\_in addr;

rset=allset;

nready=select(maxfd+1,&rset,NULL,NULL,NULL);

if(FD\_ISSET(listenfd,&rset)){

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

perror("accept error");

continue;

}

for(i=0;i<FD\_SETSIZE;i++){

if(client[i].fd<0){

client[i].fd=connectfd;

client[i].name=(char\*)calloc(20,sizeof(char));

client[i].addr=addr;

client[i].data=(char\*)calloc(MAXDATASIZE,sizeof(char));

client[i].name[0]='\0';

client[i].data[0]='\0';

client[i].data\_len=0;

printf("You got a connect from %s:%d\n",inet\_ntoa(client[i].addr.sin\_addr),client[i].addr.sin\_port);

send(client[i].fd,"welcome .\n",22,0);

break;

}

}

if(i==FD\_SETSIZE) printf("too many clients");

FD\_SET(connectfd,&allset);

if(connectfd>maxfd) maxfd=connectfd;

if(i>maxi) maxi=i;

if(--nready<=0) continue;

}

// printf("3\n");

for(i=0;i<=maxi;i++){

if((sockfd=client[i].fd)<0) continue;

if(FD\_ISSET(sockfd,&rset)){

if((n=recv(sockfd,recvbuf,MAXDATASIZE,0))==0){

close(sockfd);

printf("Client(%s:%d) closed. Message is %s\n",inet\_ntoa(client[i].addr.sin\_addr), client[i].addr.sin\_port,client[i].data);

FD\_CLR(sockfd,&allset);

for(int i=0;i<client[i].data\_len;i++){

printf("%c",client[i].data[i]);

}

client[i].fd=-1;

free(client[i].name);

free(client[i].data);

client[i].data\_len=0;

}else{

process\_cli(&client[i],recvbuf,n);

}

if(--nready<=0) break;

}

}

}

close(listenfd);

}

void process\_cli(CLIENT \*client, char \*recvbuf,int len){

save\_data(client->data,recvbuf,len,&client->data\_len);

send(client->fd,recvbuf,len,0);

printf("Message from client(%s:%d):",inet\_ntoa(client->addr.sin\_addr), client->addr.sin\_port);

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

printf("%c",recvbuf[i]);

}

printf("\n");

}

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

if((\*cli\_len+recv\_len)<5000){

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

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

}

\*cli\_len+=recv\_len;

}else{

for(int i=0;i<5000-\*cli\_len;i++){

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

}

\*cli\_len=5000;

}

}