CHAUDHARY HAMDAN

1905387

Networks Lab 8

02/08/2021

1. Non blocking communication by IO Multiplexing.

Code (Server):

#include <stdio.h> /\* These are the usual header files \*/

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <string.h>

#include <errno.h>

#include <stdlib.h>

#include <sys/select.h>

#include <sys/time.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/ioctl.h>

#define BACKLOG 2 /\* Number of connections in queue before being rejected\*/

#define MAXDATASIZE 1000

#define STDIN 0

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

{

int listen\_fd, conn\_fd, i;

int client\_fd[FD\_SETSIZE];

int max\_index;

int sin\_size;

struct sockaddr\_in server; /\* server's address information \*/

struct sockaddr\_in client; /\* client's address information \*/

struct sockaddr\_in temp;

if (argc != 2) { /\* this is used because our program will need 1 argument (port) \*/

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

exit(-1);

}

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

printf("socket() error\n");

exit(-1);

}

server.sin\_family = AF\_INET;

server.sin\_port = htons(atoi(argv[1])); /\* Remember htons() from "Conversions" section? =) \*/

server.sin\_addr.s\_addr = INADDR\_ANY;/\* INADDR\_ANY puts your IP address automatically \*/

bzero(&(server.sin\_zero), 8); /\* zero the rest of the structure \*/

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

printf("bind() error\n");

exit(-1);

}

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

printf("listen() error\n");

exit(-1);

}

//------------------------------------------------------------------------------------------------------------------------------

fd\_set read\_set, write\_set, all\_set;

struct timeval timeout;

int ret\_val;

max\_index = 0;

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

{

client\_fd[i] = -1;

}

//highest-numbered file descriptor in any of the three sets.

int max\_fd = listen\_fd;

/\*initialising fd sets that are storing status of various fd\*/

FD\_ZERO(&read\_set);

FD\_ZERO(&write\_set);

FD\_ZERO(&all\_set);

//Make the standard input socket non-blocking

//fcntl(STDIN, F\_SETFL, O\_NONBLOCK);

FD\_SET(listen\_fd, &all\_set);

while (1)

{

read\_set = all\_set;

write\_set = all\_set;

timeout.tv\_sec = 100;

timeout.tv\_usec = 0;

ret\_val = select(max\_fd + 1, &read\_set, NULL, NULL, &timeout);

/\*if(ret\_val == 0){

printf("\nTimeout occurred! No data after 10 seconds.\n");

}

else\*/

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

if (ret\_val == -1) {

perror("select");

exit(-1);

}

else if (FD\_ISSET(listen\_fd, &read\_set)) { //server listen\_fd is set to accept new client connection

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

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

perror("accept");

exit(-1);

}

printf("accepted a new connection\n");

/\*char temp\_buf[INET\_ADDRSTRLEN];

inet\_ntop(AF\_INET,&(client.sin\_addr),temp\_buf,INET\_ADDRSTRLEN);

if(temp\_buf == NULL)

printf("inet\_ntop error\n");

else

printf("accepted connection from client with IP Addr : %s and port Number :%u\n",temp\_buf,ntohs(client.sin\_port));\*/

//store the new connection fd in the array

if (max\_index == FD\_SETSIZE)

printf("too many clients to handle\n");

else {

client\_fd[max\_index] = conn\_fd;

max\_index++;

}

//Set the conection fd to read and write data

FD\_SET(conn\_fd, &all\_set);

if (conn\_fd > max\_fd)

max\_fd = conn\_fd;

} //end of FD\_ISSET

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

else { //Check the client for data

for (i = 0; i < FD\_SETSIZE; i++) //loop checks all possible sockets that are connected

{

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

char temp\_buf[INET\_ADDRSTRLEN];

char buf[1000];

//checks if a particular socket is ready for reading

if (FD\_ISSET(client\_fd[i], &read\_set))

{

int numbytes;

numbytes = recv(client\_fd[i], buf, MAXDATASIZE, 0);

if (numbytes > 0)

{

printf("Message from the client is %s\n", buf);

/\*buf[numbytes]='\0';

if(getpeername(client\_fd[i],(struct sockaddr \*)&temp,&sin\_size)==-1)

{

printf("Peername error\n");

exit(-1);

}

inet\_ntop(AF\_INET,&(temp.sin\_addr),temp\_buf,INET\_ADDRSTRLEN);

if(temp\_buf == NULL)

printf("inet\_ntop error\n");

else

printf("Message from client %s on port %u :%s",temp\_buf,ntohs(temp.sin\_port),buf);\*/

}//if(numbytes>0)

printf("Enter your message for the client\n");

fgets(buf, 1000, stdin);

send(client\_fd[i], buf, strlen(buf), 0);

/\*if(getpeername(client\_fd[i],(struct sockaddr \*)&temp,&sin\_size)==-1)

{

printf("Peername error\n");

exit(-1);

}

inet\_ntop(AF\_INET,&(temp.sin\_addr),temp\_buf,INET\_ADDRSTRLEN);

if(temp\_buf == NULL)

printf("inet\_ntop error\n");

else

printf("My message to client %s on port %u :",temp\_buf,ntohs(temp.sin\_port));

fgets(buf,1000,stdin);

send(client\_fd[i],buf,strlen(buf),0);\*/ /\* send message to client \*/

} //if(FD\_ISSET(client\_fd[i],&read\_set))

}// end of for

}//end of else

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}//end of while

}

\_\_\_\_\_\_\_\_

Code (Client):

#include <stdio.h>

#include <sys/types.h>

#include <errno.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <netdb.h> /\* netbd.h is needed for struct hostent =) \*/

#include <fcntl.h>

#define MAXDATASIZE 100 /\* Max number of bytes of data \*/

#define STDIN 0

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

{

int fd, numbytes, nval; /\* files descriptors \*/

char buf[MAXDATASIZE]; /\* buf will store received text \*/

struct hostent \*he; /\* structure that will get information about remote host \*/

struct sockaddr\_in server; /\* server's address information \*/

if (argc !=3) { /\* this is used because our program will need 2 arguments (IP,port) \*/

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

exit(-1);

}

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

printf("socket() error\n");

exit(-1);

}

server.sin\_family = AF\_INET;

server.sin\_port = htons(atoi(argv[2])); /\* htons() is needed again \*/

//server.sin\_addr = \*((struct in\_addr \*)he->h\_addr); /\*he->h\_addr passes "\*he"'s info to "h\_addr" \*/

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

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

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

printf("connect() error\n");

exit(-1);

}

static char buf1[1000];

//defining sets to hold the fd for using them with select function. Master is necessary because these sets will change once they are sent to the select function

fd\_set rset,master,wset;

struct timeval timeout;

/\*initialising fd sets that are storing status of fd\*/

FD\_ZERO(&rset);

FD\_ZERO(&master);

//FD\_ZERO(&wset);

//adding fd to master set

FD\_SET(fd,&master);

FD\_SET(STDIN,&master);

do

{

rset=master;

//wset=master;

//time tells select to monitor for that much amount of time.

timeout.tv\_sec=100;

timeout.tv\_usec=0;

//select system call takes two sets to check if fd is ready for readio or writeio

nval=select(fd+1,&rset,NULL,NULL,&timeout);//using select to verify if it is possible to

//nval=select(fd+1,&rset,&wset,NULL,NULL);//using select to verify if it is possible to

//if some data waiting to be read. then read it

if(FD\_ISSET(fd,&rset))

{

numbytes=recv(fd,buf,MAXDATASIZE,0);

buf[numbytes]='\0';

printf("Server Message: %s\n",buf);

}

//printf("My Message to server:");

//if socket is ready to be written then scan data from user and send it

if(FD\_ISSET(STDIN,&rset))

{

fgets(buf1,1000,stdin);

send(fd,buf1,strlen(buf1),0);

}

}while(strncmp(buf,"exit",4)!=0);

close(fd); /\* close fd =) \*/

}

Output:

