CN Lab Week 2

Sample:

Server:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <string.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <netdb.h>

#include <arpa/inet.h>

#include <sys/wait.h>

#include <signal.h>

void main()

{

int sd,nd,n,len,reult;

struct sockaddr\_in seradress,cliaddr;

char buf[256];

sd=socket(AF\_INET,SOCK\_STREAM,0);

seradress.sin\_family=AF\_INET;

seradress.sin\_addr.s\_addr=INADDR\_ANY;

seradress.sin\_port=htons(10200);

bind(sd,(struct sockaddr\*)&seradress,sizeof(seradress));

listen(sd,5);

len=sizeof(cliaddr);

while(1){

nd=accept(sd,(struct sockaddr\*)&cliaddr,&len);

if(fork()==0){

close(sd);

n=read(nd,buf,sizeof(buf));

printf("message from client %s\n",buf);

getchar();

}

close(nd);

}

}

Client:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <string.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <netdb.h>

#include <arpa/inet.h>

#include <sys/wait.h>

#include <signal.h>

int main()

{

int sd,nd,n,len,reult,n1;

struct sockaddr\_in seraddress, cliaddr;

char buf[256], buf1[256];

sd=socket(AF\_INET, SOCK\_STREAM,0);

seraddress.sin\_family=AF\_INET;

seraddress.sin\_addr.s\_addr=INADDR\_ANY;

seraddress.sin\_port=htons(10200);

len=sizeof(seraddress);

connect(sd,(struct sockaddr\*)&seraddress,len);

printf("enter the message tosen \n");

gets(buf);

n=write(sd,buf,strlen(buf));

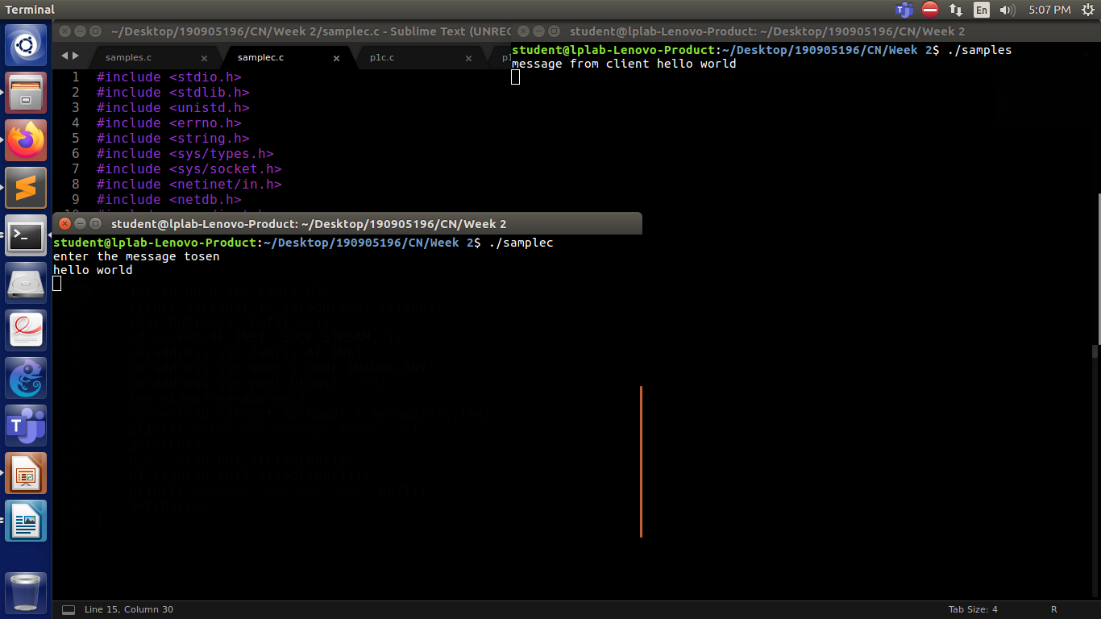
n1=read(sd,buf1,sizeof(buf1));

printf("message from ser %s\n",buf1);

getchar();

}

Output:



A 1)

Server:

Client:

Output:

A 2)

Server:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <ctype.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#define MAXSIZE 150

#define PORT 5000

#define MAXLINE 1000

typedef struct obj

{

double a,b,r;

char op;

char ans[10];

}obj1,\*obj\_ptr;

void main()

{

int sockfd,newsockfd,retval;

socklen\_t actuallen;

int recedbytes,sentbytes, sentans;

struct sockaddr\_in serveraddr,clientaddr;

obj\_ptr buffer = (obj\_ptr)malloc(sizeof(obj1));

sockfd=socket(AF\_INET,SOCK\_STREAM,0);

if(sockfd==-1)

printf("\nSocket creation error");

serveraddr.sin\_family=AF\_INET;

serveraddr.sin\_port=htons(PORT);

serveraddr.sin\_addr.s\_addr=htons(INADDR\_ANY);

bind(sockfd,(struct sockaddr\*)&serveraddr,sizeof(serveraddr));

puts("Server Running");

listen(sockfd,1);

actuallen=sizeof(clientaddr);

newsockfd=accept(sockfd,(struct sockaddr\*)&clientaddr,&actuallen);

do

{

recv(newsockfd,buffer,sizeof(obj1),0);

if(strcmp(buffer->ans, "stop") == 0)

{

puts("Stopping");

close(sockfd);

close(newsockfd);

}

else

{

printf("Client [%s:%d] requested: %.2lf %c %.2lf\n", inet\_ntoa(clientaddr.sin\_addr), ntohs(clientaddr.sin\_port), buffer->a, buffer->op, buffer->b);

switch (buffer->op)

{

case '+': buffer->r = buffer->a + buffer->b;

break;

case '-': buffer->r = buffer->a - buffer->b;

break;

case '\*': buffer->r = buffer->a \* buffer->b;

break;

case '/': buffer->r = buffer->a / buffer->b;

break;

case '%': buffer->r = buffer->a / buffer->b;

break;

default:

break;

}

sentbytes = send(newsockfd,buffer,sizeof(obj1),0);

}

}while(strcmp(buffer->ans, "stop") != 0);

}

Client:

#include <stdio.h>

#include <unistd.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <arpa/inet.h>

#include <string.h>

#include <stdlib.h>

#define MAXSIZE 150

#define PORT 5000

#define MAXLINE 1000

typedef struct obj

{

double a,b,r;

char op;

char ans[10];

}obj1,\*obj\_ptr;

void main()

{

int sockfd,retval;char ch;

int recedbytes,sentbytes, recans;

struct sockaddr\_in serveraddr;

obj\_ptr buffer = (obj\_ptr)malloc(sizeof(obj1));

obj\_ptr buffer1 = (obj\_ptr)malloc(sizeof(obj1));

sockfd=socket(AF\_INET,SOCK\_STREAM,0);

if(sockfd==-1)

printf("\nSocket Creation Error");

printf("\nSocket ID : %d\n",sockfd);

serveraddr.sin\_family=AF\_INET;

serveraddr.sin\_port=htons(PORT);

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

retval=connect(sockfd,(struct sockaddr\*)&serveraddr,sizeof(serveraddr));

if(retval==-1)

printf("Connection error");

do

{

printf("Do you want to request? Yes/Stop\n");

scanf("%c",&ch);

scanf("%[^\n]%\*c",(buffer->ans));

if(strcmp(buffer->ans,"stop")==0)

{

puts("Stopping");

sentbytes=send(sockfd,buffer,sizeof(buffer),0);

close(sockfd);

}

else

{

printf("Enter in form a op b : ");

scanf("%lf %c %lf",&buffer->a, &buffer->op, &buffer->b);

sentbytes=send(sockfd,buffer,sizeof(obj1),0);

recedbytes=recv(sockfd,buffer1,sizeof(obj1),0);

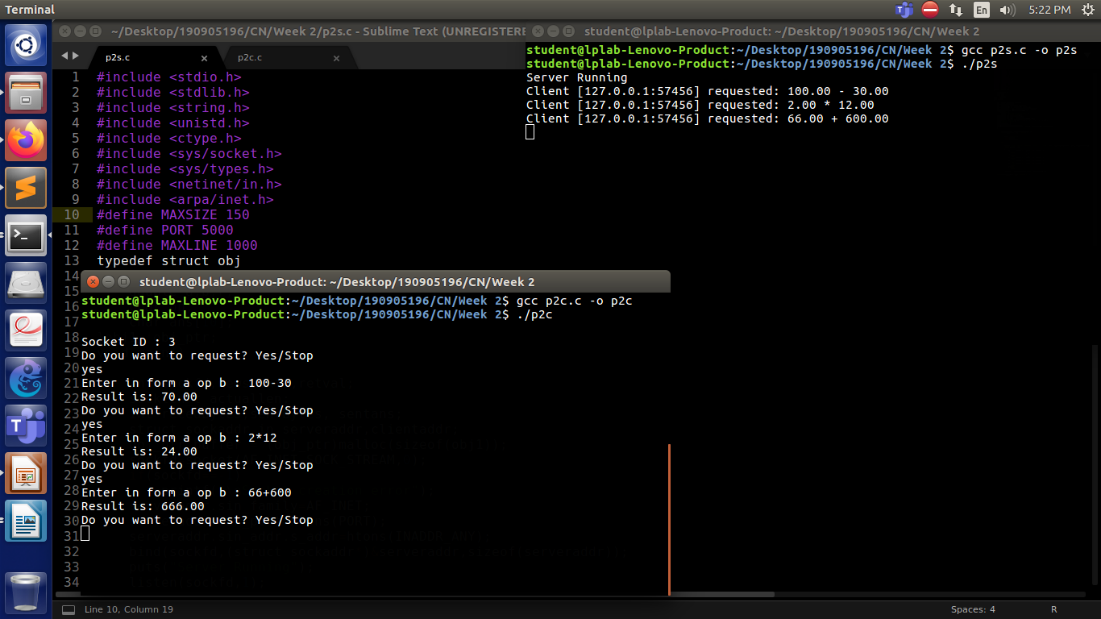
printf("Result is: %.2lf \n",buffer1->r);

}

}while(strcmp(buffer->ans, "stop") != 0);

}

Output:



A 3)

Server:

#include <sys/types.h>

#include <sys/socket.h>

#include <stdio.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <stdlib.h>

#include <time.h>

void main()

{

time\_t rawtime;

struct tm \* timeinfo;

char \*reply;

int server\_sockfd, client\_sockfd;

int server\_len, client\_len;

struct sockaddr\_in server\_address;

struct sockaddr\_in client\_address;

int hour,mins,sec,pid;

/\* Create an unnamed socket for the server. \*/

server\_sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

/\* Name the socket. \*/

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

server\_address.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

server\_address.sin\_port = 9734;

server\_len = sizeof(server\_address);

bind(server\_sockfd, (struct sockaddr \*)&server\_address, server\_len);

/\* Create a connection queue and wait for clients. \*/

listen(server\_sockfd, 5);

while(1)

{

char ch;

printf("server waiting\n");

/\* Accept a connection. \*/

client\_len = sizeof(client\_address);

client\_sockfd = accept(server\_sockfd, (struct sockaddr \*)&client\_address, &client\_len);

/\* We can now read/write to client on client\_sockfd. \*/

//char \*inet\_ntoa(client\_addr.sin\_addr);

char \* ip\_add =inet\_ntoa(client\_address.sin\_addr);

int port=client\_address.sin\_port;

printf("IP:%s PORT:%d\n", ip\_add,port);

//get the time

time ( &rawtime );

timeinfo = localtime ( &rawtime );

reply = asctime(timeinfo);

printf ( "The current date/time is: %s", reply );

hour = timeinfo->tm\_hour;

mins = timeinfo->tm\_min;

sec = timeinfo->tm\_sec;

pid = getpid();

write(client\_sockfd, &hour, 1);

write(client\_sockfd, &mins, 1);

write(client\_sockfd, &sec, 1);

write(client\_sockfd, &pid, 1);

//close(client\_sockfd);

}}

Client:

#include <sys/types.h>

#include <sys/socket.h>

#include <stdio.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <stdlib.h>

#include <time.h>

void main()

{

int sockfd;

int len;

struct sockaddr\_in address;

struct tm \* timeinfo;

int result;

char \*reply;

int hour,mins,sec,pid;

/\* Create a socket for the client. \*/

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

/\* Name the socket, as agreed with the server. \*/

address.sin\_family = AF\_INET;

address.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

address.sin\_port = 9734;

len = sizeof(address);

/\* Now connect our socket to the server socket. \*/

result = connect(sockfd, (struct sockaddr \*)&address, len);

if(result == -1)

{

perror("oops: client2");

exit(1);

}

/\* We can now read/write via sockfd. \*/

printf(" Sending request to get the time\n");

read(sockfd, &hour , 1);

read(sockfd, &mins , 1);

read(sockfd, &sec , 1);

read(sockfd, &pid , 1);

printf("%d:%d:%d", hour, mins, sec);

printf(" The process id is: %d",pid);

close(sockfd);

exit(0);

}

Output:

