Multithreading Client Server Communication

LOGESHWARAN T	COE18B033
PAVENDHAN N	COE18B041
SHARAN SK	CED18I049
VS PRAVEEN	CED18I054
YOGA SRI VARSHAN	CED18I058
JOHN ZAKKAM	CED18I059

1: Develop client and server program, where the server generates a separate thread for each incoming client request. Server shows the client accepted port number and ip address.

----------CLIENT-CODE----------

```
#include <stdio.h>
#include <stdlib.h>

#include <string.h>
#include <unistd.h>
#include <fcntl.h>

#include <sys/time.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/uio.h>
#include <sys/stat.h>

#define SIZE 512
char buf[SIZE];

int main(){
  char *ip = "127.0.0.1"; //localhost
  int port = 8082;
  int e;
  int l = sizeof(struct sockaddr_in);
  int sockfd;
  struct sockaddr_in server_addr;

  sockfd = socket(AF_INET, SOCK_STREAM, 0);
  if(sockfd < 0) {
    perror("[-]Error in socket");
    exit(1);
  }
  printf("[+]Server socket created successfully.\n");

  server_addr.sin_family = AF_INET;
  server_addr.sin_port = port;
  server_addr.sin_addr.s_addr = inet_addr(ip);

  e = connect(sockfd, (struct sockaddr*)&server_addr, sizeof(server_addr));
  if(e == -1) {
    perror("[-]Error in socket");
    exit(1);
```

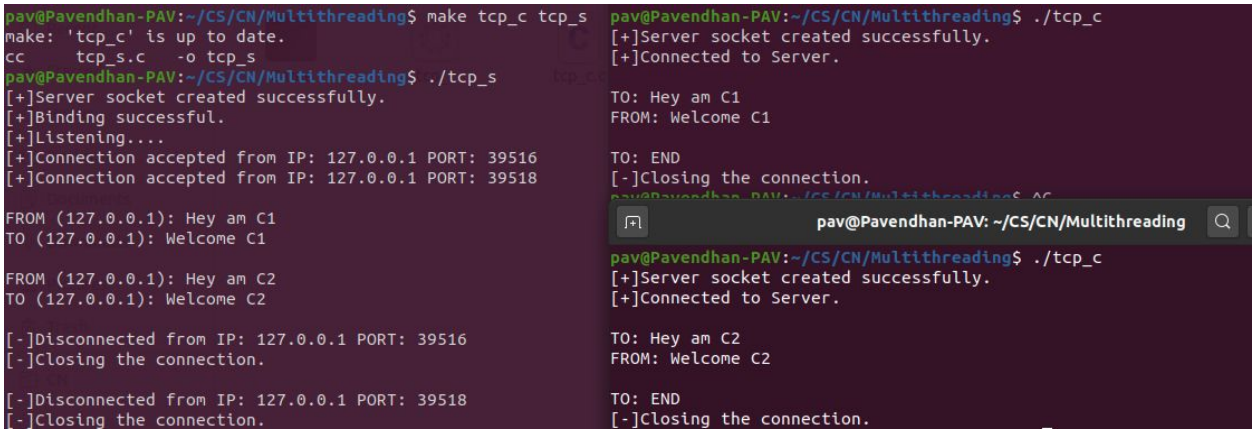
```
 }
        printf("[+]Connected to Server.\n");

//------------------------SEND-----------------------------------------------

        while(1)
        {
                printf("\nTO: ");
                fgets(buf,SIZE,stdin);
                send(sockfd, buf, SIZE, 0);

                if(strncmp(buf,"END",3) == 0)
                        break;

                recv(sockfd, buf, SIZE, 0);
                printf("FROM: %s",buf);
        }

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

        printf("[-]Closing the connection.\n");
        close(sockfd);

        return 0;
}

-------------------------------------------------------SERVER-CODE-----------------------------------------------------
#include <stdio.h>
#include <stdlib.h>

#include <string.h>
#include <unistd.h>
#include <fcntl.h>

#include<sys/wait.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/uio.h>
#include <sys/stat.h>

#define SIZE 512
char buf[SIZE];
int flag=0;
```

```
int main(){

  int port = 8082;
  int e,n,m;
  struct timeval start, stop, delta;

  int sockfd, new_sock,fd;
  struct sockaddr_in server_addr, new_addr;
  socklen_t addr_size;


  sockfd = socket(AF_INET, SOCK_STREAM, 0);
  if(sockfd < 0) {
    perror("[-]Error in socket");
    exit(1);
  }
  printf("[+]Server socket created successfully.\n");

  server_addr.sin_family = AF_INET;
  server_addr.sin_port = port;
  server_addr.sin_addr.s_addr = INADDR_ANY;

  e = bind(sockfd, (struct sockaddr*)&server_addr, sizeof(server_addr));
  if(e < 0) {
    perror("[-]Error in bind");
    exit(1);
  }
  printf("[+]Binding successful.\n");

  if(listen(sockfd, 10) == 0){
                printf("[+]Listening....\n");
        }else{
                perror("[-]Error in listening\n");
    exit(1);
        }

  addr_size = sizeof(struct sockaddr_in);


//------------------------------RECV-----------------------------------------------

        while(1)
```

```
	{
		new_sock = accept(sockfd, (struct sockaddr*)&new_addr, &addr_size);
		if(new_sock < 0)
		{
			printf("[-]Error in accepting %s.",inet_ntoa(new_addr.sin_addr));
			exit(1);
		}
		else
			printf("[+]Connection accepted from IP: %s PORT:
%d\n",inet_ntoa(new_addr.sin_addr),ntohs(new_addr.sin_port));

		pid_t pid;
		if((pid=fork()) == 0)
		{
			close(sockfd);
			while(1)
			{
				recv(new_sock, buf, SIZE, 0);

				if(strncmp(buf,"END",3) == 0)
				{
					printf("\n[-]Disconnected from IP: %s PORT:
%d\n",inet_ntoa(new_addr.sin_addr),ntohs(new_addr.sin_port));
					flag=1;
					break;
				}
				else
				{
					printf("\nFROM (%s):
%s",inet_ntoa(new_addr.sin_addr),buf);

					printf("TO (%s): ",inet_ntoa(new_addr.sin_addr));
					fgets(buf,SIZE,stdin);
					send(new_sock, buf, SIZE, 0);
					bzero(buf, SIZE);
				}
			}
		}
		if(pid==0)
			break;
	}

//-------------------------------------------------------------------------------
```

```
  printf("[-]Closing the connection.\n");
  close(sockfd);
  close(new_sock);
  return 0;
}
```

----------OUTPUT----------



2: Develop a game , where multiuser participate in the game. Server has 5 questions with four options. if users are connected to the server, the server starts sending the question one by one with a timestamp reply of 1 minute. If multiple users are playing the game, then the winner is decided by the server based on the average minimum time taken for a client to reply to all answers. If a client gives a wrong reply to anyone of the answers, then the server sends a message to client "better luck next time" and terminates the connection of the client.

----------CLIENT-CODE----------

```
#include<stdio.h>
#include<string.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<stdlib.h>
#include<unistd.h>
#include <fcntl.h>
#include<poll.h>
#include<sys/wait.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/uio.h>
```

```
#include <sys/stat.h>

void interact(int c_sock,socklen_t add)
{
char msg[250];
struct pollfd mypoll={STDIN_FILENO,POLLIN|POLLPRI};
//type yes
recv(c_sock,msg,sizeof(msg),0);
printf("Server: %s\n\n",msg);


scanf("%s",msg);
send(c_sock,msg,sizeof(msg),0);
printf("\n");
bzero(msg,250);


recv(c_sock,msg,sizeof(msg),0);

if(strcmp(msg,"ack")==0)
{
while(1)
{

//Question
recv(c_sock,msg,sizeof(msg),0);
printf("Server: %s\n\n",msg);

if(strcmp(msg,"You win")==0 || strcmp(msg,"Better luck next time")==0 )
break;
//printf("Client: ");
//Answer send

if( poll (&mypoll,1,60000))
{scanf("%s",msg);}
else
{strcpy(msg,"T");
printf("Time up!!!\n");}
send(c_sock,msg,sizeof(msg),0);
printf("\n");

bzero(msg,250);
recv(c_sock,msg,sizeof(msg),0);
```

```
if(strcmp(msg,"Better luck next time")==0 )
{
printf("Server: %s\n\n",msg);
break;

}


}

}

else
{
printf("Server: %s\n",msg);
exit(0);

}
}

int main()
{

int c_sock;


//Socket Descriptor
c_sock=socket(AF_INET,SOCK_STREAM,0);
if(c_sock>0)
{

struct sockaddr_in server_addr;

//Server info
server_addr.sin_family=AF_INET;
server_addr.sin_port=htons(9024);
server_addr.sin_addr.s_addr=INADDR_ANY;

int ch=1;
socklen_t add;
add=sizeof(server_addr);
//Message Loop
if(connect(c_sock,(struct sockaddr*)&server_addr,sizeof(server_addr))>=0)
```

```
{
interact(c_sock,add);
}
else
{

printf("[-]ERROR FOUND AT CONNECTION:\n");
}

}
else
{

printf("[-]ERROR FOUND AT SOCKET:\n");
}


close(c_sock);
return 0;



}
```

----------SERVER-CODE----------

```
#include<stdio.h>
#include<string.h>
#include<sys/types.h>
#include<sys/socket.h>
#include <sys/time.h>
#include<netinet/in.h>
#include<stdlib.h>
#include<unistd.h>
#include<time.h>
#include <fcntl.h>

#include<sys/wait.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/uio.h>
#include <sys/stat.h>

int duration(struct timeval *start, struct timeval *stop, struct timeval *delta)
```

```
{
        suseconds_t microstart, microstop, microdelta;

        microstart = (suseconds_t)(100000 * (start->tv_sec)) + start->tv_usec;
        microstop = (suseconds_t)(100000 * (stop->tv_sec)) + stop->tv_usec;
        microdelta = microstop - microstart;

        delta->tv_usec = microdelta % 100000;
        delta->tv_sec = (time_t)(microdelta / 100000);

        if ((*delta).tv_sec < 0 || (*delta).tv_usec < 0)
                return -1;
        else
                return 0;
}

int interact(int c_sock,socklen_t add,struct sockaddr_in client_addr)
{
char msg[250];


                        struct timeval tv;
                         tv.tv_sec = 5;

int count,timestamp;
float sum=0;//for choice ,to display result
//Command Display
struct timeval start,tss,tse, stop, delta,delta1;
char s[5][100];
strcpy(s[4],"what is 1+2?\nChoices:\n  a:3\n  b:2\n  c:3\n  d:4 \nYour Choice");
strcpy(s[3],"what is 3+2?\nChoices:\n  a:1\n  b:5\n  c:3\n  d:2 \nYour Choice");
strcpy(s[2],"what is 5+2?\nChoices:\n  a:1\n  b:2\n  c:7\n  d:4 \nYour Choice");
strcpy(s[1],"what is 7+2?\nChoices:\n  a:1\n  b:10\n  c:3\n  d:9 \nYour Choice");
strcpy(s[0],"what is 9+2?\nChoices:\n  a:1\n  b:2\n  c:11\n  d:4 \nYour Choice");

char ans[5];
ans[4]='a';
ans[3]='b';
ans[2]='c';
ans[1]='d';
ans[0]='c';

strcpy(msg,"Type \"yes\" to start the game");
```

```
send(c_sock,msg,sizeof(msg),0);
bzero(msg,250);

//recv yes
recv(c_sock,msg,sizeof(msg),0);

if(strcmp(msg,"yes")==0)
{

strcpy(msg,"ack");
send(c_sock,msg,sizeof(msg),0);
bzero(msg,250);

count=5;
sum=0;
while(count--)
{
strcpy(msg,s[count]);

send(c_sock,msg,sizeof(msg),0);
gettimeofday(&start, NULL);

recv(c_sock,msg,sizeof(msg),0);

gettimeofday(&stop, NULL);


                duration(&start, &stop, &delta);
                long int sec= delta.tv_sec;
                if(sec<0) sec*=-1;
                //printf("\n Time taken to receive: %ld \n", sec);
                sum+=sec;




if(msg[0]!=ans[count])
{
bzero(msg,250);
strcpy(msg,"Better luck next time");
send(c_sock,msg,sizeof(msg),0);
break;
}
```

```
else
{
bzero(msg,250);
strcpy(msg,"Continue");
send(c_sock,msg,sizeof(msg),0);
}
}
bzero(msg,250);
float avg_time = sum/5.0;

FILE *fp = fopen("Avgtime.txt","a");
fprintf(fp,"%f",avg_time);
fprintf(fp,"%s","\n");
fclose(fp);
fp = fopen("Port.txt","a");
fprintf(fp,"%d",ntohs(client_addr.sin_port));
fprintf(fp,"%s","\n");
fclose(fp);

printf("\nTotal Time Average: %f \n",sum/5);
/*strcpy(msg,"Win");
send(c_sock,msg,sizeof(msg),0);
*/


}
else
{
strcpy(msg,"Invalid Choice");
send(c_sock,msg,sizeof(msg),0);
exit(0);
}
}


int main()
{

int s_sock,c_sock,len;

FILE *fp = fopen("Avgtime.txt","w");
fclose(fp);
fp = fopen("Port.txt","w");
```

```
fclose(fp);
//socket file descriptor
s_sock=socket(AF_INET,SOCK_STREAM,0);

if(s_sock>0)
{

struct sockaddr_in server_addr,client_addr;

memset(&server_addr,0,sizeof(server_addr));
memset(&client_addr,0,sizeof(client_addr));

//server information
server_addr.sin_family=AF_INET;
server_addr.sin_port=htons(9024);
server_addr.sin_addr.s_addr=INADDR_ANY;

socklen_t add;
add=sizeof(client_addr);

if(bind(s_sock,(struct sockaddr*)&server_addr,sizeof(server_addr))>=0)
{printf("[+]Server Binded\n");
if((listen(s_sock,10))!=-1)
{
printf("[+]Server Listening for Connection\n");
len=sizeof(client_addr);
while(1)
{
c_sock=accept(s_sock,(struct sockaddr*)&client_addr,&len);

if(c_sock!=-1)
{

printf("[+]Connection accepted from IP: %s PORT:
%d\n",inet_ntoa(client_addr.sin_addr),ntohs(client_addr.sin_port));
        pid_t pid;
                if((pid=fork()) == 0)
                {
//close(s_sock);
interact(c_sock,add,client_addr);
// to compare time between multiple users
FILE *fp = fopen("Avgtime.txt","r");
char num[20],i=0;
```

```
float avg_time[5];
 while(fscanf(fp, "%s", num)!=EOF)
 {
            avg_time[i] = atof(num);
            i++;
 }
   fclose(fp);

  int j,min=0;
  for(j=0;j<i;j++)
{
          if(avg_time[j] < avg_time[min])
                    min = j;
}
fp = fopen("Port.txt","r");
int k=0,port;
while(k<=min)
{
          fscanf(fp,"%s",num);
          k++;
          port = atoi(num);
}
printf("Average time of winner in port %s : %f\n",num,avg_time[min]);
if(port == ntohs(client_addr.sin_port))
{
          char msg[250];
          strcpy(msg,"You win");
          send(c_sock,msg,sizeof(msg),0);
}
else
{
          char msg[250];
          strcpy(msg,"Better luck next time");
          send(c_sock,msg,sizeof(msg),0);
}

//kill(pid,SIGKILL);
}
if(pid==0)
break;

}//accept
else
```

```
{
printf("[-]ERROR FOUND AT ACCEPT:\n");
}
}
}//listen
else
{
printf("[-]ERROR FOUND AT LISTEN:\n");
}
}//bind
else
{
printf("[-]ERROR FOUND AT BIND:\n");
}
}
else
{

printf("[-]ERROR FOUND AT SOCKET:\n");
}

close(c_sock);
close(s_sock);
return 0;
}
```

----------OUTPUT----------

```
(base) sharan@OMEN:~/Downloads$ ./tcp_c
Server: Type "yes" to start the game

Client: yes

Server: what is 1+2?
Choices:
  a:3
  b:2
  c:3
  d:4

a
Client:
Server: what is 3+2?
Choices:
  a:1
  b:5
  c:3
  d:2

b
Client:
Server: what is 5+2?
Choices:
  a:1
  b:2
  c:7
  d:4

c
Client:
Server: what is 7+2?
Choices:
  a:1
  b:10
  c:3
  d:9

d
Client:
Server: what is 9+2?
Choices:
  a:1
  b:2
  c:11
  d:4

c
Client:
Server: You win
```

In Case of Client takes more than 60 seconds to answer:

```
(base) sharan@OMEN:~/Downloads$ ./tcp_c
Server: Type "yes" to start the game

yes

Server: what is 1+2?
Choices:
  a:3
  b:2
  c:3
  d:4
Your Choice

Time up!!!

Server: Better luck next time
```

Multiple clients accessing the game:

```
sharan@OMEN: ~/Downloads
Choices:
  a:1
  b:10
  c:3
  d:9
Your Choice

d

Server: what is 9+2?
Choices:
  a:1
  b:2
  c:11
  d:4
Your Choice

c

Server: You win

(base) sharan@OMEN:~/Downloads$
```

```
sharan@OMEN: ~/Downloads
Server: what is 7+2?
Choices:
  a:1
  b:10
  c:3
  d:9
Your Choice

d

Server: what is 9+2?
Choices:
  a:1
  b:2
  c:11
  d:4
Your Choice

c

Server: Better luck next time

(base) sharan@OMEN:~/Downloads$
```

```
sharan@OMEN: ~/Downloads
Total Time Average: 0.400000
Average time of winner in port 52004 : 0.400000
^C
(base) sharan@OMEN:~/Downloads$ ./tcp_s
[+]Server Binded
[+]Server Listening for Connection
[+]Connection accepted from IP: 127.0.0.1 PORT: 52006

Total Time Average: 6.400000
Average time of winner in port 52006 : 6.400000
[+]Connection accepted from IP: 127.0.0.1 PORT: 52024

Total Time Average: 12.000000
Average time of winner in port 52006 : 6.400000
[+]Connection accepted from IP: 127.0.0.1 PORT: 52062
[+]Connection accepted from IP: 127.0.0.1 PORT: 52064

Total Time Average: 2.400000
Average time of winner in port 52062 : 2.400000

Total Time Average: 8.200000
Average time of winner in port 52062 : 2.400000
```