Name : Rajvardhan Reddy
Reg No : 180905093
Sec : B
Roll No : 19

CN LAB – SESSION 2 : Socket programming

P1) Solved Problem : Concurrent TCP server and client

Program :
Server side :

```
#include<unistd.h>
#include<stdlib.h>
#include<stdio.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<strings.h> //for bzero
#include<arpa/inet.h>
#include<string.h>
#define PORT 5050
#define ERROR -1
#define MAX 1024
int main(int argc, char const *argv[])
{
int sockfd, clientfd, data_len;
struct sockaddr_in server, client;
char buffer[MAX];
pid_t pid;
//Step 1: Create a socket
if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == ERROR){
perror("server socket: ");
exit(-1);
}
//Step 2: Assign the socket ip and port
server.sin_family = AF_INET;
server.sin_port = htons(PORT);
server.sin_addr.s_addr = INADDR_ANY;
//Clear the structurebzero(&server.sin_zero, 8);
//Step 3: bind the socket to the ip address
if((bind(sockfd, (struct sockaddr*) &server, sizeof(struct sockaddr_in))) == ERROR)
{
perror("sbind: ");
exit(-1);
```

```
}
//Step 4: The listen() function converts an unconnected socket into a passive socket
if((listen(sockfd, 5)) == ERROR){
perror("listen: ");
exit(-1);
}
int len = sizeof(struct sockaddr_in);
//Infinite loop to keep the server from closing
while(1){
//Step 5: accept request
//The accept() is used to retrieve a connect request and convert that into a
request.
clientfd = accept(sockfd, (struct sockaddr*) &client, &len);
pid = fork();
if(pid == 0){
//Child process begins here
//child closes listening socket
close(sockfd);
while(1)
{
//Step 6: process the request here
bzero(buffer, MAX);
data_len = read(clientfd, buffer, sizeof(buffer));
if(strncmp("exit", buffer, 4) == 0){
printf("[-]Disconnected from %s : %d\n\n",
inet_ntoa(client.sin_addr), ntohs(client.sin_port));
break;
}
printf("Message from client: %s\n", buffer);
}
close(clientfd);
exit(0); //Terminate child process
}
}return 0;
}
```

Client side :

```
#include<unistd.h>
#include<stdlib.h>
#include<stdio.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<strings.h> //for bzero
#include<arpa/inet.h>
#include<sys/types.h>
```

```
#include<string.h>
#include<errno.h>
#define PORT 5050
#define ERROR -1
#define MAX 1024
int main(int argc, char const *argv[])
{
int sockfd, len;
struct sockaddr_in remoteServer;
char buffer[MAX];
//Step 1: create a socket
if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == ERROR){
perror("server socket: ");
exit(-1);
}
//Step 2: Assign the socket ip and port
remoteServer.sin_family = AF_INET;
remoteServer.sin_port = htons(PORT);
remoteServer.sin_addr.s_addr = INADDR_ANY;
bzero(&remoteServer.sin_zero, 8);
//Step 3: connect to the remoteserver
if((connect(sockfd, (struct sockaddr*) &remoteServer, sizeof(struct sockaddr_in)))
==
ERROR){
perror("server socket: ");
exit(-1);
}
while(1)
{
printf("\nClient: ");fgets(buffer, MAX, stdin);
len = write(sockfd, buffer, strlen(buffer));
if(strncmp("exit", buffer, 4) == 0){
close(sockfd);
printf("[-]Disconnected from server.\n");
exit(1);
}
}
return 0;
}
```

Output :

server side :

```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB/
lab2$ cc lab2_p1_server.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB/
lab2$ ./a.out
Message from client: hello

Message from client: this is client here

Message from client: bye
```

client side :

```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB/
lab2$ cc lab2_p1_server.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB/
lab2$ ./a.out
Message from client: hello

Message from client: this is client here

Message from client: bye
```

P2) Implement concurrent Remote Math Server To perform arithmetic operations in the server and display the result at the client. The client accepts two integers and an operator from the user and sends it to the server. The server will performs the operation on integers and sends result back to the client which is displayed in the client.

Program :
server code :

```
#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;
struct sockaddr_in seraddress, cliaddr;
sd=socket(AF_INET, SOCK_STREAM,0);
```

```
seraddress.sin_family=AF_INET;
seraddress.sin_addr.s_addr=INADDR_ANY;
seraddress.sin_port=htons(10200);
bind(sd,(struct sockaddr*)&seraddress,sizeof(seraddress));
listen(sd,2);
len = sizeof(cliaddr);while(1)
{
char buff[100];
nd = accept(sd,(struct sockaddr*)&cliaddr,&len);
if (fork()==0)
{
close(sd);
// printf("Forked\n");
read(nd,buff,100);
printf("%d %d are the two received integers with operation
to be performed : %c ",buff[0],buff[1],buff[2]);
char result[100];
// printf("%c\n",buff[2]);
switch(buff[2])
{
case '+': result[0] = (buff[0])+(buff[1]);
break;
case '-': result[0] = (buff[0])-(buff[1]);
break;
case '*': result[0] = (buff[0])*(buff[1]);
break;
case '/': result[0] = (buff[0])/(buff[1]);
break;
}
printf("\n the result is %d ", result[0]);
write(nd, result, 100);
close(nd);
exit(0);
}
}
}
```

client code :

```
#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;
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);
char buff[100];
int a , b;
char op;
printf("Enter two integers to perform the operation \n");
scanf(" %d %d",&a,&b);
printf("Enter the operator:");
scanf(" %c",&op);buff[0]=a;
buff[1]=b;
buff[2]=op;
buff[3]='\0';
write(sd,&buff,100);
char buf1[100];
n1=read(sd,buf1,100);
printf("%d %c %d : %d\n",a,op,b, *buf1);
}
```

Output :
server side :

```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB/
lab2$ cc lab2_p2_server.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB/
lab2$ ./a.out
6 9 are the two received integers with operation to be performed : *
 the result is 54
```

client side :

```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB/
lab2$ cc lab2_p2_client.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB/
lab2$ ./a.out
Enter two integers to perform the operation
6 9
Enter the operator: *
6 * 9 : 54
```

P3) DayTime Server: Where client sends request to time server to send current time. Server responds by sending the current time . [Hint: read man pages of asctime() and localtime()] . Display server process id at client side along with time.

Program :
server code :

```
#include <stdio.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include<unistd.h>
#include<time.h>
#define MAX 80
#define PORT 8088
#define SA struct sockaddr
// Function designed for chat between client and server.
void servfunc(int sockfd)
{
char buff[MAX];
int n;
// infinite loop for chat
for (;;) {
bzero(buff, MAX);
// read the message from client and copy it in buffer
read(sockfd, buff, sizeof(buff));
// print buffer which contains the client contents
printf("From client: %s\t To client : ", buff);
// Read server message from keyboard in the buffer
n=0;
/* Define temporary variables */
time_t current_time;struct tm *local_time;
/* Retrieve the current time */
```

```
current_time = time(NULL);
/* Get the local time using the current time */
local_time = localtime(&current_time);
/* Display the local time */
char* bufft=asctime(local_time);
pid_t pid = getpid();
// printf("pid: %lun", pid);
// and send that buffer to client
write(sockfd, bufft, sizeof(buff));
// if msg contains "Exit" then server exit and session ended.
if (strncmp("quit", buff, 4) == 0) {
printf("Server Exit...\n");
break;
}
}
}
// Driver function
int main()
{
int sockfd, connfd, len;
struct sockaddr_in servaddr, cli;
// socket create and verification
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd == -1) {
printf("socket creation failed...\n");
exit(0);
}
else
printf("Socket successfully created..\n");
bzero(&servaddr, sizeof(servaddr));// assign IP, PORT
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(PORT);
// Binding newly created socket to given IP and verification
if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr))) != 0) {
printf("socket bind failed...\n");
exit(0);
}
else
printf("Socket successfully binded..\n");
// Now server is ready to listen and verification
if ((listen(sockfd, 5)) != 0) {
printf("Listen failed...\n");
exit(0);
}
```

```
else
printf("Server listening..\n");
len = sizeof(cli);
// Accept the data packet from client and verification
connfd = accept(sockfd, (SA*)&cli, &len);
if (connfd < 0) {
printf("server acccept failed...\n");
exit(0);
}
else
printf("server acccept the client...\n");
int prt=ntohs (cli.sin_port);
printf("%d\n",prt);
// Function for chatting between client and server
servfunc(connfd);
// After sending exit message close the socket
close(sockfd);
}
```

client code :

```
#include<unistd.h>
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include<arpa/inet.h>
#include <string.h>
#include <sys/socket.h>
#define MAX 80
#define PORT 8088
#define SA struct sockaddr
void clifunc(int sockfd)
{
char buff[MAX];
int n;
for (;;) {
bzero(buff, sizeof(buff));
printf("Enter the string : ");
n = 0;
while ((buff[n++] = getchar()) != '\n')
;
write(sockfd, buff, sizeof(buff));
bzero(buff, sizeof(buff));
read(sockfd, buff, sizeof(buff));
printf("From Server : %s\n", buff);
pid_t pid = getpid();
```

```
printf("pid: %lu \n",pid);
if ((strncmp(buff, "quit", 4)) == 0) {
printf("Client Exit...\n");
break;
}
}
}int main()
{
int sockfd, connfd;
struct sockaddr_in servaddr, cli;
// socket create and verification
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd == -1) {
printf("socket creation failed...\n");
exit(0);
}
else
printf("Socket successfully created..\n");
bzero(&servaddr, sizeof(servaddr));
// assign IP, PORT
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
servaddr.sin_port = htons(PORT);
// connect the client socket to server socket
if (connect(sockfd, (SA*)&servaddr, sizeof(servaddr)) != 0) {
printf("connection with the server failed...\n");
exit(0);
}
else
printf("connected to the server..\n");
// function for client
clifunc(sockfd);
// close the socket
close(sockfd);
}
```

Output :

server side :

```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB/
lab2$ cc lab2_p3_server.c
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB/
lab2$ ./a.out
Socket successfully created..
Socket successfully binded..
Server listening..
server acccept the client...
33164
From client: hello
        To client : From client: this is client here
        To client : From client: testing out a new feature
        To client : From client: thanks for your time!
```

client side :

```
rajvardhan@rajvardhan-HP-Pavilion-Laptop-15-cc1xx:~/Desktop/5th_sem_LABS/CN_LAB/
lab2$ ./a.out
Socket successfully created..
connected to the server..
Enter the string : hello
From Server : Wed Jun 23 19:31:00 2021

pid: 258277
Enter the string : this is client here
From Server : Wed Jun 23 19:31:12 2021

pid: 258277
Enter the string : testing out a new feature
From Server : Wed Jun 23 19:31:21 2021

pid: 258277
Enter the string : thanks for your time!
From Server : Wed Jun 23 19:31:40 2021

pid: 258277
Enter the string :
```