21BPS1191

SPRIHA ANVI

CN-LAB ASSIGNMENT-2

SOCKET PROGRAMMING

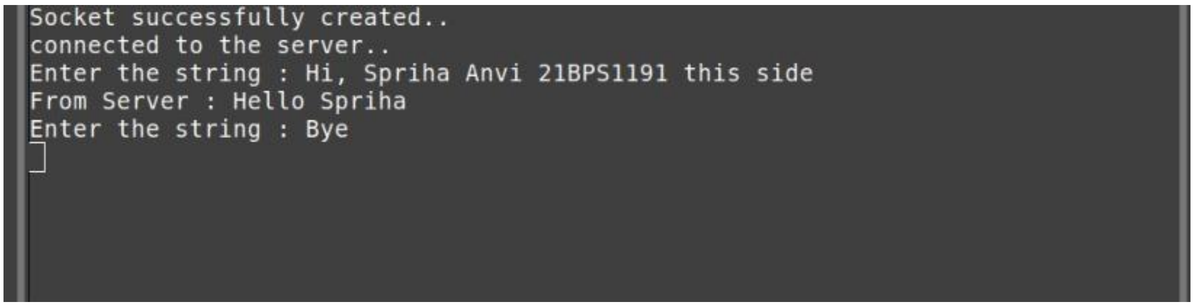
Server side program to demonstrate Socket Programming

```
#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> // read(), write(), close()
#define MAX 80
#define PORT 8080
#define SA struct sockaddr

 Void func(int connfd)
{
        char buff[MAX];
        int n;
        // infinite loop for chat
        for (;;) {
                bzero(buff, MAX);
        read(connfd, buff, sizeof(buff));
                        printf("From client: %s\t To client : ", buff);
                bzero(buff, MAX);
                n = 0;
                // copy server message in the buffer
```

```
                    while ((buff[n++] = getchar()) != '\n')
                              ;

                              write(connfd, buff, sizeof(buff));



                    if (strncmp("exit", buff, 4) == 0) {
          printf("Server Exit...\n");
                              break;
                    }
          }
}

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));

          = AF_INET; servaddr.sin_addr.s_addr
          = htonl(INADDR_ANY);
          servaddr.sin_port = htons(PORT);
```

```
                    if ((bind(sockfd, (SA*)&servaddr, sizeof(servaddr)))
!= 0) {             printf("socket bind failed...\n");
                    exit(0);
          }
          else
                    printf("Socket successfully binded..\n");

if ((listen(sockfd, 5)) != 0)
 {
printf("Listen failed...\n");
                    exit(0);
          }
          else
  printf("Server listening..\n");  len = sizeof(cli);


                    connfd = accept(sockfd, (SA*)&cli, &len);
          if (connfd < 0) {
                    printf("server accept failed...\n");
          exit(0);
          }
          else
                    printf("server accept the client...\n");


          func(connfd);

          valread = read(new_socket, buffer, 1024);

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

             send(new_socket, Hello, strlen(hello), 0);

             printf("Hi, SPRIHA ANVI 21BPS1191 this side
          sent\n");
```

```
        close(new_socket);

        shutdown(server_fd, SHUT_RDWR);

        return 0;
```



```
Socket successfully created..
connected to the server..
Enter the string : Hi, Spriha Anvi 21BPS1191 this side
From Server : Hello Spriha
Enter the string : Bye
```

```
}
```

Client side program to demonstrate Socket programming

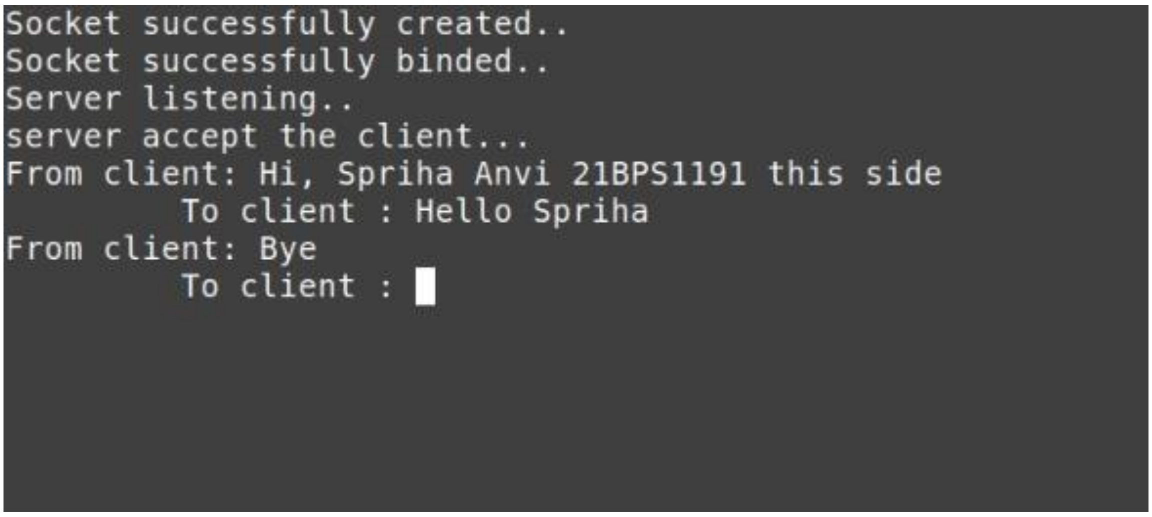
```
#include <arpa/inet.h> // inet_addr()

#include <netdb.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <strings.h> // bzero()

#include <sys/socket.h>

#include <unistd.h> // read(), write(), close()

#define MAX 80

#define PORT 8080 #define

SA struct sockaddr void

func(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", buff);       if
((strncmp(buff, "exit", 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 chat
        func(sockfd);

        // close the socket
        close(sockfd);
}
```

Echo service

Server side

```
#include <stdio.h> // perror, printf
#include <stdlib.h> // exit, atoi
#include <unistd.h> // read, write, close
#include <arpa/inet.h> // sockaddr_in, AF_INET, SOCK_STREAM, INADDR_ANY, socket
etc...
#include <string.h> // memset  Int
main(int argc, char const *argv[]) {  int
serverFd, clientFd;  struct sockaddr_in
server, client;  int len; int port = 1234;
char buffer[1024];  if (argc == 2) { port =
atoi(argv[1]);
}
serverFd = socket(AF_INET, SOCK_STREAM, 0);

if (serverFd < 0) { perror("Cannot create socket");

exit(1);

 }
 server.sin_family = AF_INET;  server.sin_addr.s_addr
 = INADDR_ANY;
 server.sin_port = htons(port); len = sizeof(server);
 if (bind(serverFd, (struct sockaddr *)&server, len) <
 0) { perror("Cannot bind
   sokcet");
   exit(2);
 }
 if (listen(serverFd, 10) < 0)
 {
   perror("Listen error");
   exit(3);
 }
 while (1) {
   len = sizeof(client); printf("waiting for clients\n");  if ((clientFd =
   accept(serverFd, (struct sockaddr *)&client, &len)) < 0)
   {
     perror("accept error");
     exit(4);
   }
   char *client_ip = inet_ntoa(client.sin_addr);
   printf("Accepted new connection from a client %s:%d\n", client_ip,
   ntohs(client.sin_port)); memset(buffer, 0, sizeof(buffer)); int size = read(clientFd,
   buffer, sizeof(buffer)); if ( size < 0 ) {
     perror("read error");
     exit(5);
   }
printf("received %s from client\n", buffer); if
   (write(clientFd, buffer, size) < 0) {
     perror("write error");
     exit(6);
   }
   close(clientFd);
 }
 close(serverFd);
 return 0;
}

send(client_fd, hello, strlen(hello), 0);
```

```
  printf("Hello message sent\n");
  valread = read(client_fd, buffer, 1024);
  printf("%s\n", buffer);
```



```
Socket successfully created..
Socket successfully binded..
Server listening..
server accept the client...
From client: Hi, Spriha Anvi 21BPS1191 this side
         To client : Hello Spriha
From client: Bye
         To client : 
```

Client side

```
#include <stdio.h> // perror, printf

#include <stdlib.h> // exit, atoi

#include <unistd.h> // write, read, close

#include <arpa/inet.h> // sockaddr_in, AF_INET, SOCK_STREAM, INADDR_ANY, socket
etc...

#include <string.h> // strlen, memset const char message[] = "Hello sockets world\n";

int main(int argc, char const *argv[]) {

int serverFd; struct sockaddr_in server; int len; int port = 1234; char *server_ip = "127.0.0.1";
char *buffer = "hello server";  if (argc == 3) {

server_ip = argv[1]; port = atoi(argv[2]);

 }
 serverFd=socket(AF_INET, SOCK_STREAM,
 0);  if (serverFd <
 0) {
   perror("Cannot create socket");
   exit(1);
 }
 server.sin_family = AF_INET; server.sin_addr.s_addr =
 inet_addr(server_ip); server.sin_port = htons(port); len =
 sizeof(server);  if (connect(serverFd, (struct sockaddr
 *)&server, len)< 0)
 {
   perror("Cannot connect to
   server");
   exit(2);
 }

 if (write(serverFd, buffer, strlen(buffer)) < 0) {
   perror("Cannot write");
   exit(3);
 }
 char recv[1024]; memset(recv,
 0, sizeof(recv));
 if (read(serverFd,recv, sizeof(recv)) < 0)
 {
   perror("cannot read");
   exit(4);
 }
 printf("Received %s from server\n", recv);
 close(serverFd); return 0;
}
```

Date and time server:

Server Code:

```
#include"netinet/in.h"

#include"sys/socket.h"

#include"stdio.h"

#include"string.h"

#include"time.h"

main( ) {  struct

sockaddr_in sa;

struct sockaddr_in cli;

int sockfd,conntfd; int

len,ch; char str[100];

time_t tick;

sockfd=socket(AF_INET,SOCK_STREAM,0);
if(sockfd<0) {  printf("error
in socket\n"); exit(0);
}
else
printf("Socket opened");  bzero(&sa,sizeof(sa));
sa.sin_port=htons(5600);
sa.sin_addr.s_addr=htonl(0);
if(bind(sockfd,(struct sockaddr*)&sa,sizeof(sa))<0)
{
printf("Error in binding\n");
}
else
printf("Binded Successfully");
listen(sockfd,50);
for(;;) { len=sizeof(ch); conntfd=accept(sockfd,(struct sockaddr*)&cli,&len);
printf("Accepted");
tick=time(NULL); snprintf(str,sizeof(str),"%s",ctime(&tick));
printf("%s",str);

write(conntfd,str,100);
}
}
```

Client:

```
#include"netinet/in.h"
#include"sys/socket.h"
#include"stdio.h"  main()
{  struct sockaddr_in
sa,cli;  int n,sockfd;  int
len;char buff[100];
sockfd=socket(AF_INET,SOCK_STREAM,0);
if(sockfd<0) {  printf("\nError in
Socket"); exit(0);
}
else
printf("\nSocket is Opened");
bzero(&sa,sizeof(sa));  sa.sin_family=AF_INET;
sa.sin_port=htons(5600);
if(connect(sockfd,(struct sockaddr*)&sa,sizeof(sa))<0)
{
printf("\nError in connection failed");  exit(0);
}
Else
 printf("\nconnected successfully");
if(n=read(sockfd,buff,sizeof(buff))<0)
{
printf("\nError in Reading");  exit(0);
 }
else
{
printf("\nMessage Read %s",buff);
}
}
```

Time of the Day:

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


        BACKLOG 10
```

```
int main(int argc, char **argv){  if(argc

!= 2){

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

 }


 int port = atoi(argv[1]);
 printf("Port: %d\n", port);


 int n_client = 0;


 int sockfd=socket(AF_INET,SOCK_STREAM,

 0);


 struct sockaddr_in serverAddress;

 serverAddress.sin_family = AF_INET;

 serverAddress.sin_addr.s_addr =

 INADDR_ANY;


 serverAddress.sin_port = htons(port);


 bind(sockfd, (struct sockaddr*)&serverAddress,
sizeof(serverAddress)); printf("[+]Bind\n");


 listen(sockfd, BACKLOG);


 printf("[+]Listening for the client\n");
int i = 1;  while(i){ int client_socket =

accept(sockfd, NULL, NULL); n_client++; time_t

currentTime; time(&currentTime);

    printf("Client %d requested for time at %s", n_client,
ctime(&currentTime)); send(client_socket,
ctime(&currentTime), 30, 0);

}


return 0;




}
```

Character Generation

```
#include<stdio.h>
#include<conio.h> #include<graphics.h>
void main()
{
int gd=DETECT,gm,i,j; int
a[20][20]=
{{0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,
1,1,1,1,0,0},
{0,0,1,0,0,0,1,0,0,0,0,0,0,1,0
,0,0,0,1,0},
{0,1,0,0,0,0,0,1,0,0,0,0,1,0,0
,0,0,0,0,1},
{1,0,0,0,0,0,0,0,0,0,0,1,0,0,0
,0,0,0,0,0},
{1,0,0,0,0,0,0,0,0,0,0,1,0,0,0
,0,0,0,0,0},
{1,0,0,0,0,0,0,0,0,0,0,1,0,0,0
,1,1,1,1,0},
{1,0,0,0,0,0,0,0,0,0,0,1,0,0,0
,0,0,0,1,0},
{0,1,0,0,0,0,0,1,0,0,0,1,0,0,0
,0,0,0,1,0},
{0,0,1,0,0,0,1,0,0,0,0,0,1,0,0
,0,0,1,0,0},
{0,0,0,1,1,1,0,0,0,0,0,0,0,1,1
,1,1,0,0,0}};

 initgraph(&gd,&gm,"c:\\tc\\bg
i");
for(i=0;i<19;i++)
{
for(j=0;j<19;j++)
{
if(a[i][j]==1)
putpixel(100+j,200+i,WHITE
);
}
}
getch(); }
```