

Ex: 4

Date: 13.09.2020

CHAT USING TCP

Develop a simple chat using TC P socket. To a chat server, multiple stations chat simultaneously.

Sample Input Output

Client

Client :-----

Server:-----

Client :-----

Server:-----

Client :-----

Server:-----

Server

Server: -----

Client1:-----

Client2:-----

Server:-----

Client2:-----

Server:-----

Client 3:-----

Client1:-----

Client 3:-----

Server code:

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

```
#include<unistd.h>
```

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

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

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

```
#include<netinet/in.h>
```

```
#include<string.h>
```

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

```
{
```

```
    int len;
```

```
    int sockfd,newfd,n;
```

```
    struct sockaddr_in servaddr,cliaddr;
```

```
    char buff[1024];
```

```
    char str[1000];
```

```
    sockfd=socket(AF_INET,SOCK_STREAM,0);
```

```
    if(sockfd<0)
```

```
    {
```

```
        perror("cannot create socket");
```

```
        exit(1);
```

```
    }
```

```

bzero(&servaddr,sizeof(servaddr));

servaddr.sin_family=AF_INET;
servaddr.sin_addr.s_addr=INADDR_ANY;
servaddr.sin_port=htons(7228);

if(bind(sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr))<0)
{
    perror("Bind error");
    exit(1);
}
if(listen(sockfd,5)<0)
{
    perror("listen error");
    exit(1);
}
len=sizeof(cliaddr);
while(1)
{
    newfd=accept(sockfd,(struct sockaddr*)&cliaddr,&len);
    pid_t childprocess;
    if((childprocess=fork())==0)
    {
        while((n=read(newfd,buff,sizeof(buff))>0))
        {
            printf("\n\nClient %d : %s",getpid(),buff);
            if(strcmp(buff,"exit")==0)
            {
                printf("\nServer : Connection terminated with Client %d\n",getpid());
            }

            bzero(buff,1024);
            printf("\nServer : ");
            scanf(" %[^\\n]",buff);

            n=write(newfd,buff,sizeof(buff));
            if(strcmp(buff,"exit")==0)
            {
                printf("\nExiting Server\n");
                exit(1);
            }
        }
        close(sockfd);
        close(newfd);
        return 0;
    }
}
}

```

Client code:

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

int main(int argc,char **argv)
{
    int len;
    int sockfd,n;
    struct sockaddr_in servaddr,cliaddr;
    char buff[1024];
    char str[1000];

    sockfd=socket(AF_INET,SOCK_STREAM,0);
    if(sockfd<0)
    {
        perror("cannot create socket");
        exit(1);
    }
    bzero(&servaddr,sizeof(servaddr));

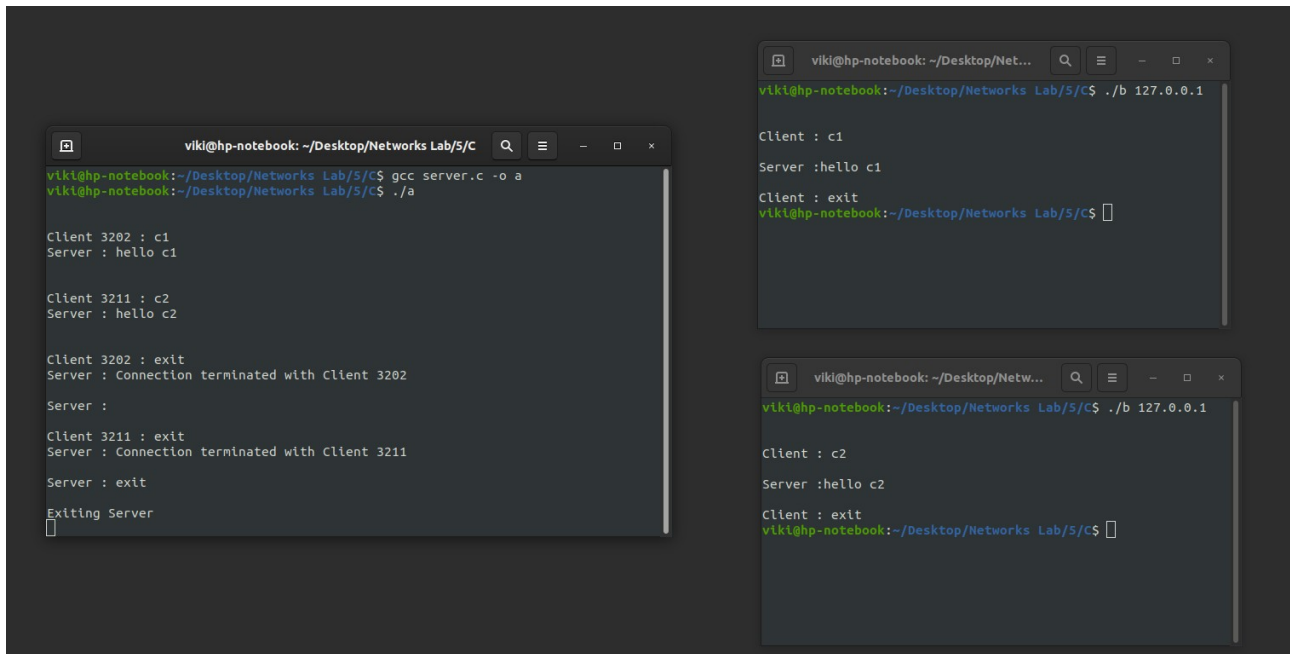
    servaddr.sin_family=AF_INET;
    servaddr.sin_addr.s_addr=inet_addr(argv[1]);
    servaddr.sin_port=htons(7228);

    if(connect(sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr))<0)
    {
        perror("Connection error");
        exit(1);
    }

    while(1)
    {
        printf("\n\nClient : ");
        scanf(" %c[^\n]",buff);
        n=write(sockfd,buff,sizeof(buff));
        if(strcmp(buff,"exit")==0)
            break;
        bzero(buff,1024);
        n=read(sockfd,buff,sizeof(buff));
        if(strcmp(buff,"exit")==0)
            break;
        printf("\nServer :%s",buff);
    }

    close(sockfd);
    return 0;
}
```

Sample I/O:



The image displays three terminal windows from a user named 'viki' on a system named 'hp-notebook'. The windows show the process of compiling and running a C program that acts as a server.

Terminal 1 (Left): Shows the compilation of 'server.c' into 'a' and its execution. The server listens on port 127.0.0.1 and successfully handles two clients, 'c1' and 'c2', before exiting.

```
viki@hp-notebook: ~/Desktop/Networks Lab/5/C
viki@hp-notebook:~/Desktop/Networks Lab/5/C$ gcc server.c -o a
viki@hp-notebook:~/Desktop/Networks Lab/5/C$ ./a

Client 3202 : c1
Server : hello c1

Client 3211 : c2
Server : hello c2

Client 3202 : exit
Server : Connection terminated with Client 3202

Server :

Client 3211 : exit
Server : Connection terminated with Client 3211

Server : exit

Exiting Server
█
```

Terminal 2 (Top Right): Shows the server being run with the command './b 127.0.0.1'. It receives a connection from 'c1' and responds with 'hello c1'.

```
viki@hp-notebook: ~/Desktop/Net...
viki@hp-notebook:~/Desktop/Networks Lab/5/C$ ./b 127.0.0.1

Client : c1
Server :hello c1

Client : exit
viki@hp-notebook:~/Desktop/Networks Lab/5/C$ █
```

Terminal 3 (Bottom Right): Shows the server being run with the command './b 127.0.0.1'. It receives a connection from 'c2' and responds with 'hello c2'.

```
viki@hp-notebook: ~/Desktop/Netw...
viki@hp-notebook:~/Desktop/Networks Lab/5/C$ ./b 127.0.0.1

Client : c2
Server :hello c2

Client : exit
viki@hp-notebook:~/Desktop/Networks Lab/5/C$ █
```