

ASSIGNMENT-3



Name: Sayantan Roy

Roll Number: GCECTB-R18-3025

Subject: Computer Network Lab

Dept: CSE

Year:3rd

ASSIGNMENT -3

TCP TIME CLIENT SERVER PROGRAM

IN C

**TCP DATETIME CLIENT SERVER PROGRAM
IN C**

SYSTEM USED :

OPERATING SYSTEM : kali linux 2020.1

KERNEL: 4.19.153

GUI: DEFAULT XSESSION (XFCE4)

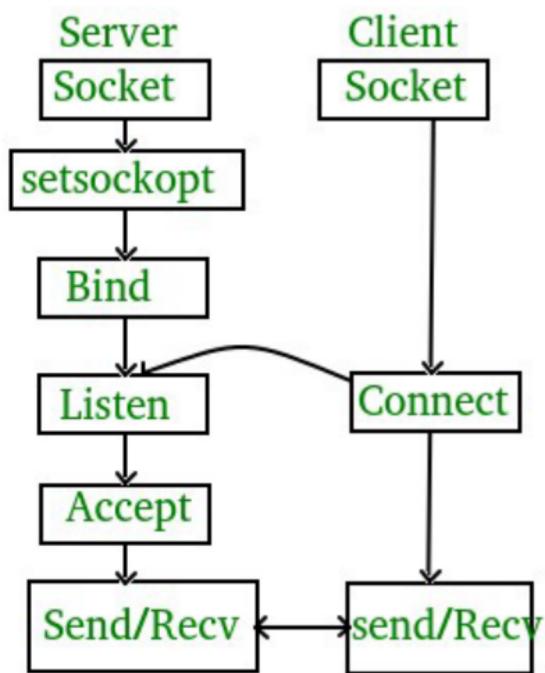
TEXT EDITOR :NANO

system owned and project done by :Sayantan Roy

TCP DATETIME CLIENT SERVER PROGRAM IN C

If we are creating a connection between client and server using TCP then it has few functionality like, TCP is suited for applications that require high reliability, and transmission time is relatively less critical. It is used by other protocols like HTTP, HTTPS, FTP, SMTP, Telnet. TCP rearranges data packets in the order specified. There is absolute guarantee that the data transferred remains intact and arrives in the same order in which it was sent. TCP does Flow Control and requires three packets to set up a socket connection, before any user data can be sent. TCP handles reliability and congestion control. It also does error checking and error recovery. Erroneous packets are retransmitted from the source to the destination.

The entire process can be broken down into following steps:



The entire process can be broken down into following steps:

TCP Server –

- 1.using **create()**, Create TCP socket.
- 2.using **bind()**, Bind the socket to server address.
- 3.using **listen()**, put the server socket in a passive mode, where it waits for the client to approach the server to make a connection
- 4.using **accept()**, At this point, connection is established between client and server, and they are ready to transfer data.
- 5.Go back to Step 3.

TCP Client –

- 1.Create TCP socket.
- 2.connect newly created client socket to server.

Arguments :

domain –Specifies the communication domain (AF_INET for IPv4/ AF_INET6 for IPv6)

type –Type of socket to be created (SOCK_STREAM for TCP / SOCK_DGRAM for UDP)

protocol –Protocol to be used by socket.
0 means use default protocol for the address family.

bind : assigns address to the unbound socket

sockfd –File descriptor of socket.

addr –Structure in which address to be binded to is specified

Functions:

`socket(int domain, int type, int protocol)`

Creates an unbound socket in the specified domain.
Returns socket file descriptor.

SERVERSIDE PROGRAM:

filename :tcpdatetimeserver.c

```
#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);
}
}
```

CLIENTSIDE PROGRAM:

filename:tcpdatetimeclient.c

```
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);
}}
```

output:

run:

```
gcc tcpdatetimeserver.c -o tcpdatetimeserver
gcc tcpdatetimeclient.c -o tcpdatetimeclient
```

```
sayantan@localhost:~$ ./tcpdatetimeclient
```

```
Socket is Opened
connected successfully
Message Read Mon Jun  7 04:13:12 2021
sayantan@localhost:~$
```

```
sayantan@localhost:~$ ./tcpdatetimeserver
Socket openedBinded SuccessfullyAcceptedMon Jun  7 04:13:12 2021
```

The screenshot shows a terminal window titled "ShellNo.1" with the file "tcpdatetimeserver.c" open in the nano editor. The code implements a TCP server that binds to port 5600 and handles incoming connections by sending the current time to the client. The terminal also displays the command line with various tabs like "udp ech...", "TCP DA...", etc.

```
GNU nano 5.3                               tcpdatetimeserver.c
#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,connfd;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);
connfd=accept(sockfd,(struct sockaddr*)&cli,&len);
printf("Accepted");
tick=time(NULL);
snprintf(str,sizeof(str),"%"PRIu64,ctime(&tick));
printf("%s",str);write(connfd,str,100);
}
}

```

File Actions Edit View Help

Read 37 lines]

[Ctrl-A Help Ctrl-W Write Out Ctrl-F Where Is Ctrl-X Cut Ctrl-U Paste Ctrl-Z Execute Ctrl-C Location Ctrl-G Go To Line M-U Undo Ctrl-R Redo M-A Set Mark M-D Copy Ctrl-Q To Bracket Ctrl-Q Where Was M-W Previous Ctrl-Q Next

file:tcpdatetimeserver.c

The screenshot shows a terminal window titled "Shell No.1" with the file "tcpdateclient.c" open in the nano editor. The code is a C program that includes headers for netinet/in.h, sys/socket.h, and stdio.h. It defines a main() function that creates a socket, connects to port 5680, reads data from it, and prints the message received. The terminal window also displays a menu bar with File, Actions, Edit, View, Help, and a toolbar with various icons.

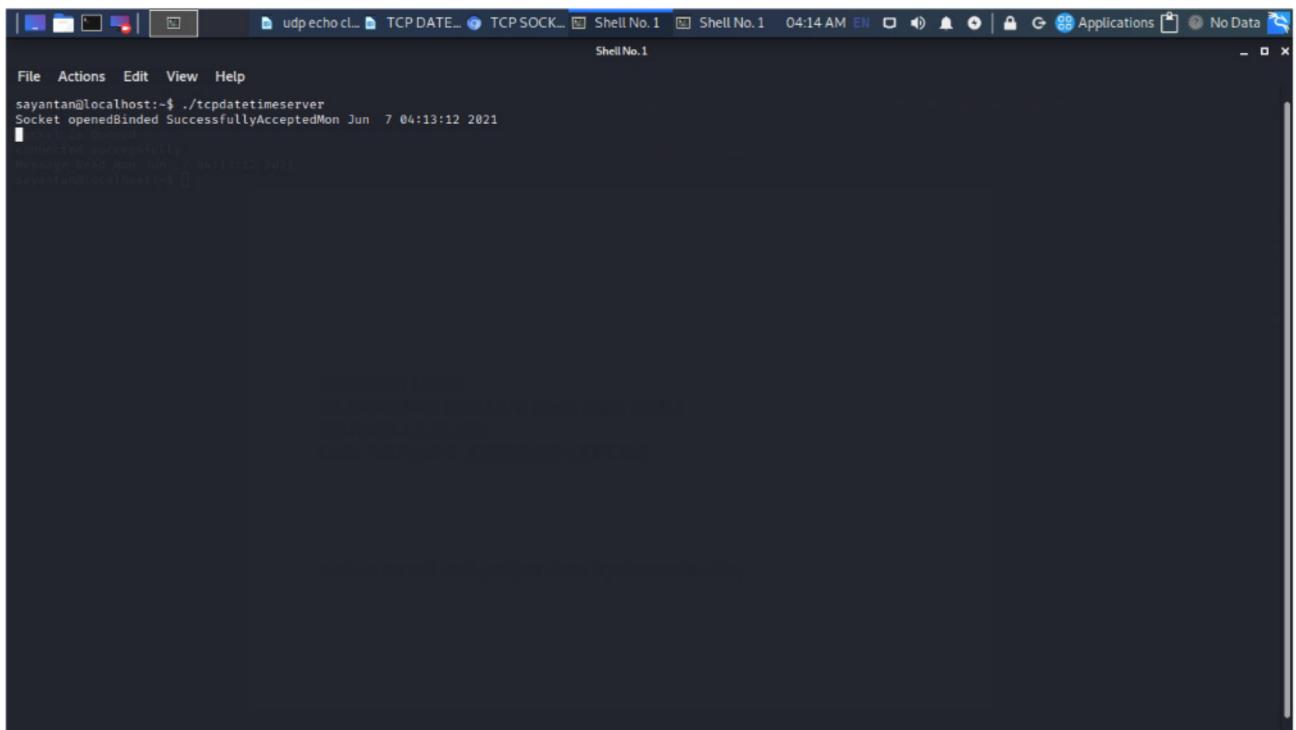
```
GNU nano 5.3                               tcpdateclient.c
#include<netinet/in.h>
#include<sys/socket.h>
#include<stdio.h>
main()
{
struct sockaddr_in sa,cli;
int n,sockfd;
int len;char buff[100];
sockfd=socket(AF_INET,SOCK_STREAM,0);
if(sockfd<0){ printf("\nError in Socket");
exit(0);
}
else printf("\nSocket is Opened");
bzero(&sa,sizeof(sa));
sa.sin_family=AF_INET;
sa.sin_port=htons(5680);
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);
}}
```

file:tcpdateclient.c

A screenshot of a terminal window titled "Shell No.1". The window shows the command `sayantani@localhost:~$./tcpdateclient` being run. The output indicates a successful connection: "Socket is Opened", "connected successfully", and "Message Read Mon Jun 7 04:13:12 2021". The terminal interface includes a menu bar with File, Actions, Edit, View, Help, and a toolbar with various icons.

```
sayantani@localhost:~$ ./tcpdateclient
Socket is Opened
connected successfully
Message Read Mon Jun 7 04:13:12 2021
sayantani@localhost:~$
```

here the client is connected sucessfully and it sends time to the server



A screenshot of a terminal window titled "Shell No.1". The window shows the command `sayantan@localhost:~$./tcpdatetimeserver` being run. The output indicates that the socket was successfully bound to port 8080 on Monday, June 7, at 04:13:12 2021. A client connection was accepted from the IP address 127.0.0.1 at the same timestamp. The message "Message sent" is also visible.

```
sayantan@localhost:~$ ./tcpdatetimeserver
Socket openedBinded SuccessfullyAcceptedMon Jun  7 04:13:12 2021
[ 127.0.0.1] connected successfully
Message sentMon Jun  7 04:13:12 2021
sayantan@localhost:~$
```

here the socket is binded sucessfully and it gets date &time from the client