PB_05_Kushagra Suryawanshi

CN LAB 6

Title: TCP Socket

Aim:

Write a C program for wired network using TCP socket to demonstrate

a. Message transfer from one machine to another machine. (50% students of the batch should implement)

b. File transfer application / Mathematical operations. (Remaining 50% students of the batch should implement)

Objectives:

1.To understand concept of socket programming:

Theory:

PB_05 Kushagra Suryawanshi

CN LAB Assignment 6

TCP ~~script~~ SOCKET

AIM: Write a C program for wired network using TCP socket to demonstrate: message transfer from one machine to another

OBJECTIVES:

1. To understand concept of socket programming.

THEORY:

i) Client / Server communication :-
It involves 2 components namely a client and a server. They are usually multiple clients in communication with single server. The client sends request to server and Server responds to it.

ii) Introduction to TCP :-
It is a standard that defines how to establish and maintain a network conversion through which application program can exchange data.

iii) TCP Segment Header :-

Source Port Address (16 bit Address)	Destination Port Address (16 bits)

Sequence no. (32 bits)

Acknowledgement no (32 bits)

HLEN	Reserved	U R G	A C K	P S H	R S T	Window size (16 bits)
Checksum (16 bits)						Urgent Pointer (16 bits)

Options & Padding (upto 40 bytes)

(iv) TCP connection establishment & release.
To establish a connection, TCP uses a 3 way handshake. Before a client attempts to connect with a ~~server~~ server, the server must ~~have~~ first bind to and listen at a port to open it up for connections; this is called passive open. Once it is established, a client may initiate an active open.

(v) Introduction to sockets:
Socket is described by 5 things:
- Family: IPV_4, IPV_6 or unix domain.
- Type: Stream or datagram or raw.
- Protocol: 0 for TCP/UDP
- local socket address
- Remote socket address

(vi) TCP Socket functions :

socket() : creates a socket
bind() : associate local address with socket.
listen() : establishes a socket to listen for incoming connection.
connect() : establishes a connection to peer.

(vii) TCP socket flow description on server :

1. The socket() f^n returns a socket descriptor representing an end point.
2. bind() : unique name for socket.
3. listen() : allow server to accept incoming client connections.
4. accept() : by server to accept incoming connection request
5. select() : allows process to wait for an event to occur.
6. recv() : receive data from client application.
7. send() : echos data back to client.
8. close() : closes any open socket descriptors.

(viii) TCP socket flow description on client :-

1. socket() function returns a socket desc representing an endpoint.
2. In client example program, if server string was passed into inet_addr(), f^n was not a dotted decimal

IP address, then it is assumed to be hostname of server.
3. The connect() f^n is used to establish a connection to the server.
4. The send() f^n sends bytes of data to the server.
5. The recv() f^n waits for server to echo the bytes of data back.
6. The close() f^n closes any socket descriptor.

FAQs.

Q1. State the IANA range for ports. List at least 5 b.

→

(i) 0 - 1023 ⇒ system ports.
(ii) 1024 - 49151 ⇒ user ports.
(iii) 49151 - 65585 ⇒ dynamic / private ports
(iv) 53 - DNS
(v) 25 - SMTP
(vi) 80 - HTTP.

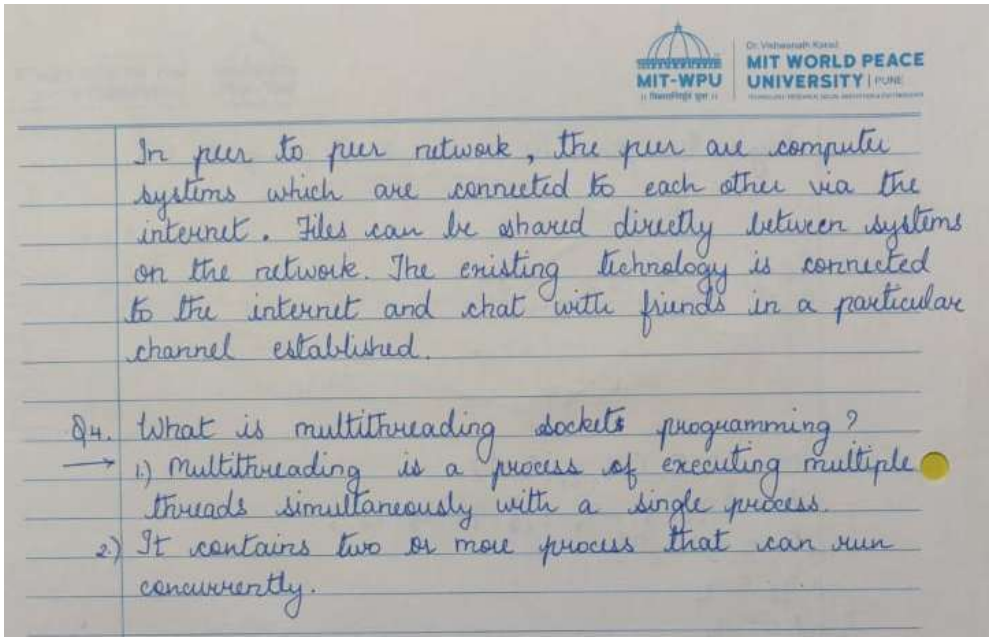
Q2. If bind() fails, what should I do with socket desc?

→

(i) If you are about to exit, uninet will close open file descr on exit.
(ii) If you are not exiting through it can be closed with regular close.

Q3. Explain with example peer to peer chatting and multiuse chatting.

→



Steps Followed:

1. Write c code for client and server on an editor.

2. Compile and create an exe file for client and server.

3. Use '. /Server1 3000' to execute the server script and './Client1.exe 3000' to execute client script (USE TWO SEPARATE TERMINALS).

4. Send messages back and forth between the terminals.

5. 'exit' message will terminate the chat and socket will be closed.

CLIENT.C

```
#include<stdio.h> //printf
#include<string.h> //strlen
#include<sys/socket.h> //socket
#include<arpa/inet.h> //inet_addz

int main(int arge, char *argv[1])
{

   int sock;
   struct sockaddr_in server;
   char message [1000], server_reply[2000];
   char ml[10], m2[10], m3[3];

   //Create socket

   sock =socket(AF_INET, SOCK_STREAM, 0);

   if (sock == -1)
   {
```

```
    printf("Could not create socket");
  }
  puts("Socket created");
  server.sin_addr.s_addr = inet_addr ("127.0.0.1");
  server.sin_family= AF_INET;
  server.sin_port = htons ( 8888 ); //Connect to remote server

  if (connect (sock, (struct sockaddr *)&server, sizeof (server)) < 0)
  {
  perror ("connect failed. Error");
  return 1;
  }
  puts ("Connected\n");
  //keep communicating with server
  while (1)
  {
    bzero (message, 2000);
    printf("Enter message: ");
    scanf ("%s", message);

    //Send some data
    if ( send (sock, message, strlen(message), 0) < 0)
    {
      puts ("Send failed");
      return 1;
    }
    //Receive a reply from the server
    if (recv(sock, server_reply, 2000, 0) < 0)
    {
    puts ("recv failed");
    break;
    }

    puts("Server reply :");
    puts (server_reply);
    strncpy (server_reply, " ", 2000);
  }
  close(sock);
  return 0;
}
```

SERVER.C

```
#include<stdio.h>

#include<string.h>

#include<stdlib.h>

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

```
#include<arpa/inet.h>
#include<unistd.h>
#include<pthread.h>

void *connection_handler(void * socket_desc)
{
    int sock = (int)socket_desc;
    int read_size;
    char *message, client_message[2000];
    //Receive a message from client
        while( (read_size= recv(sock, client_message, 2000, 0)) > 0) //Send the message back to client
        {
            write (sock, client_message, strlen(client_message));
        }
    if (read_size== 0)
    {
      puts ("Client disconnected");
      fflush(stdout);
    }
    else if (read_size == -1)
    {
       perror ("recv failed");
    }

    free (socket_desc);
    return 0;
}

int main(int argc, char *argv[])
{
    int socket_desc, client_sock, c, *new_sock;
    struct sockaddr_in server, client;
```

```
socket_desc = socket(AF_INET, SOCK_STREAM, 0);
if(socket_desc == -1)
{
   printf("Could not create socket");
}
puts("Socket Created");
server.sin_family = AF_INET;
server.sin_addr.s_addr = INADDR_ANY;
server.sin_port = htons(8888);
if(bind(socket_desc,(struct sockaddr *)&server, sizeof(server))<0)
{
   perror("Bind failed. Error!");
   return 1;
}
puts("Bind Done");

listen(socket_desc,3);
puts("Waiting for incoming connections...");
c = sizeof(struct sockaddr_in);

while((client_sock = accept(socket_desc,(struct sockaddr )&client,(socklen_t)&c)))
{
   puts("Connection accepted");

pthread_t sniffer_thread;
new_sock=malloc(1);
*new_sock = client_sock;
if( pthread_create(&sniffer_thread,NULL,connection_handler, (void*) new_sock)<0)
{
   perror("Could not create thread!");
   return 1;
}
```

```
    pthread_join(sniffer_thread,NULL);
    puts("Handler assigned");
    }
    if(client_sock<0)
    {
        perror("Accept failed!");
        return 1;
    }
}
```

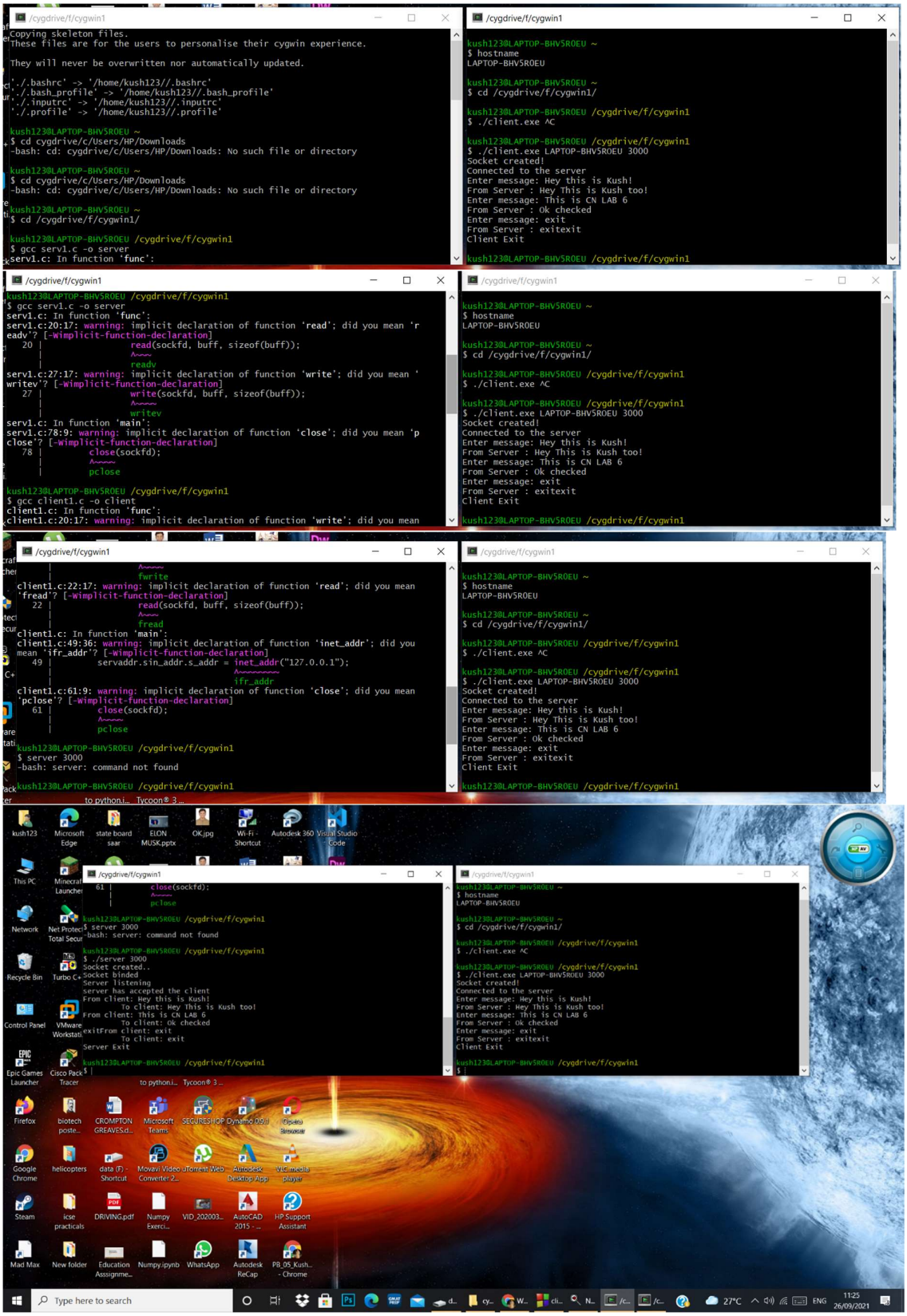
ALGORITHM:

Client:

1. Initialise sock, sock address, message, reply and variables.
2. Create socket and store value in sock variable.
3. If sock already exists or is equal to -1 do not create socket; else create socket.
4. Net address of socket 127.0.0.1; family = AF_INET; port = 8888 (connecting to remote server)
5. Socket values less than 0, show message connection failed else show connected.
6. Enter message and communicate amongst client and server. Show error message if send or receive failed.
7. End

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.


/cygdrive/f/cygwin1
Copying skeleton files.
These files are for the users to personalise their cygwin experience.
They will never be overwritten nor automatically updated.
'./.bashrc' -> '/home/kush123//.bashrc'
'./.bash_profile' -> '/home/kush123//.bash_profile'
'./.inputrc' -> '/home/kush123//.inputrc'
'./.profile' -> '/home/kush123//.profile'
kush123@LAPTOP-BHV5ROEU ~
$ cd cygdrive/c/Users/HP/Downloads
-bash: cd: cygdrive/c/Users/HP/Downloads: No such file or directory
kush123@LAPTOP-BHV5ROEU ~
$ cd cygdrive/c/Users/HP/Downloads
-bash: cd: cygdrive/c/Users/HP/Downloads: No such file or directory
kush123@LAPTOP-BHV5ROEU ~
$ cd /cygdrive/f/cygwin1/
kush123@LAPTOP-BHV5ROEU /cygdrive/f/cygwin1
$ gcc serv1.c -o server
serv1.c: In function 'func':
/cygdrive/f/cygwin1
kush123@LAPTOP-BHV5ROEU ~
$ hostname
LAPTOP-BHV5ROEU
kush123@LAPTOP-BHV5ROEU ~
$ cd /cygdrive/f/cygwin1/
kush123@LAPTOP-BHV5ROEU /cygdrive/f/cygwin1
$ ./client.exe ^C
kush123@LAPTOP-BHV5ROEU /cygdrive/f/cygwin1
$ ./client.exe LAPTOP-BHV5ROEU 3000
Socket created!
Connected to the server
Enter message: Hey this is Kush!
From Server : Hey This is Kush too!
Enter message: This is CN LAB 6
From Server : Ok checked
Enter message: exit
From Server : exitexit
Client Exit
kush123@LAPTOP-BHV5ROEU /cygdrive/f/cygwin1
/cygdrive/f/cygwin1
kush123@LAPTOP-BHV5ROEU /cygdrive/f/cygwin1
$ gcc serv1.c -o server
serv1.c: In function 'func':
serv1.c:20:17: warning: implicit declaration of function 'read'; did you mean 'readv'? [-Wimplicit-function-declaration]
20 | read(sockfd, buff, sizeof(buff));
| ^~~~
| readv
serv1.c:27:17: warning: implicit declaration of function 'write'; did you mean 'writev'? [-Wimplicit-function-declaration]
27 | write(sockfd, buff, sizeof(buff));
| ^~~~~
| writev
serv1.c: In function 'main':
serv1.c:78:9: warning: implicit declaration of function 'close'; did you mean 'pclose'? [-Wimplicit-function-declaration]
78 | close(sockfd);
| ^~~~~
| pclose
kush123@LAPTOP-BHV5ROEU /cygdrive/f/cygwin1
$ gcc client1.c -o client
client1.c: In function 'func':
client1.c:20:17: warning: implicit declaration of function 'write'; did you mean
/cygdrive/f/cygwin1
kush123@LAPTOP-BHV5ROEU ~
$ hostname
LAPTOP-BHV5ROEU
kush123@LAPTOP-BHV5ROEU ~
$ cd /cygdrive/f/cygwin1/
kush123@LAPTOP-BHV5ROEU /cygdrive/f/cygwin1
$ ./client.exe ^C
kush123@LAPTOP-BHV5ROEU /cygdrive/f/cygwin1
$ ./client.exe LAPTOP-BHV5ROEU 3000
Socket created!
Connected to the server
Enter message: Hey this is Kush!
From Server : Hey This is Kush too!
Enter message: This is CN LAB 6
From Server : Ok checked
Enter message: exit
From Server : exitexit
Client Exit
kush123@LAPTOP-BHV5ROEU /cygdrive/f/cygwin1
/cygdrive/f/cygwin1
fwrite
client1.c:22:17: warning: implicit declaration of function 'read'; did you mean 'fread'? [-Wimplicit-function-declaration]
22 | read(sockfd, buff, sizeof(buff));
| ^~~~
| fread
client1.c: In function 'main':
client1.c:49:36: warning: implicit declaration of function 'inet_addr'; did you mean 'ifr_addr'? [-Wimplicit-function-declaration]
49 | servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
| ^~~~~~~~~
| ifr_addr
client1.c:61:9: warning: implicit declaration of function 'close'; did you mean 'pclose'? [-Wimplicit-function-declaration]
61 | close(sockfd);
| ^~~~~
| pclose
kush123@LAPTOP-BHV5ROEU /cygdrive/f/cygwin1
$ server 3000
-bash: server: command not found
kush123@LAPTOP-BHV5ROEU /cygdrive/f/cygwin1
/cygdrive/f/cygwin1
kush123@LAPTOP-BHV5ROEU ~
$ hostname
LAPTOP-BHV5ROEU
kush123@LAPTOP-BHV5ROEU ~
$ cd /cygdrive/f/cygwin1/
kush123@LAPTOP-BHV5ROEU /cygdrive/f/cygwin1
$ ./client.exe ^C
kush123@LAPTOP-BHV5ROEU /cygdrive/f/cygwin1
$ ./client.exe LAPTOP-BHV5ROEU 3000
Socket created!
Connected to the server
Enter message: Hey this is Kush!
From Server : Hey This is Kush too!
Enter message: This is CN LAB 6
From Server : Ok checked
Enter message: exit
From Server : exitexit
Client Exit
kush123@LAPTOP-BHV5ROEU /cygdrive/f/cygwin1
kush123
Microsoft Edge
state board saar
ELON MUSK.pptx
OK.jpg
Wi-Fi - Shortcut
Autodesk 360
Visual Studio Code
This PC
Minecraft Launcher
Network
Net Protect Total Secu
Recycle Bin
Turbo C++
Control Panel
VMware Workstati
Epic Games Launcher
Cisco Pack Tracer
/cygdrive/f/cygwin1
61 | close(sockfd);
| ^~~~~
| pclose
kush123@LAPTOP-BHV5ROEU /cygdrive/f/cygwin1
$ server 3000
-bash: server: command not found
kush123@LAPTOP-BHV5ROEU /cygdrive/f/cygwin1
$ ./server 3000
Socket created..
Socket binded
Server listening
server has accepted the client
From client: Hey this is Kush!
To client: Hey This is Kush too!
From client: This is CN LAB 6
To client: Ok checked
exitFrom client: exit
To client: exit
Server Exit
kush123@LAPTOP-BHV5ROEU /cygdrive/f/cygwin1
/cygdrive/f/cygwin1
kush123@LAPTOP-BHV5ROEU ~
$ hostname
LAPTOP-BHV5ROEU
kush123@LAPTOP-BHV5ROEU ~
$ cd /cygdrive/f/cygwin1/
kush123@LAPTOP-BHV5ROEU /cygdrive/f/cygwin1
$ ./client.exe ^C
kush123@LAPTOP-BHV5ROEU /cygdrive/f/cygwin1
$ ./client.exe LAPTOP-BHV5ROEU 3000
Socket created!
Connected to the server
Enter message: Hey this is Kush!
From Server : Hey This is Kush too!
Enter message: This is CN LAB 6
From Server : Ok checked
Enter message: exit
From Server : exitexit
Client Exit
kush123@LAPTOP-BHV5ROEU /cygdrive/f/cygwin1
to python.i... Tycoon® 3 ...
Firefox
biotech poste...
CROMPTON GREAVES.d...
Microsoft Teams
SECURES OP Dynamo 0.0.1
Opera Browser
Google Chrome
helicopters
data (F) - Shortcut
Movavi Video Converter 2...
uTorrent Web
Autodesk Desktop App
VLC media player
Steam
icse practicals
DRIVING.pdf
Numpy Exerci...
VID_202003...
AutoCAD 2015 - ...
HP Support Assistant
Mad Max
New folder
Education Assignme...
Numpy.ipynb
WhatsApp
Autodesk ReCap
P8_OS_Kush... - Chrome
Type here to search
27°C
ENG
11:25
26/09/2021

Student Observation:

• TCP ensures reliable transmission

• Exchanges data between applications as a stream of bytes

• To use Linux libraries in Windows OS, Cygwin terminal should be installed;

• Client and server scripts should be executed in different terminals.

Conclusion:

Thus, we studied and implemented a chatting application using socket programming.