Madhurya Mozumder

RA1911028010036

CSE-CC

I2

Experiment 5 Concurrent TCP/IP Day-Time Server

## Aim:

To implement a TCP/IP day time server (concurrent server) that handles multiple client requests. Once the client establishes connection with the server, the server sends its day-time details to the Client which the client prints in its console.

## Procedure:

Server

* Include the necessary header files.
* Create a socket using the socket function with family AF\_INET, type as SOCK\_STREAM.
* Initialize server address to 0 using the bzero function.
* Assign the sin\_family to AF\_INET, sin\_addr to INADDR\_ANY, sin\_port to statically assigned port number.
* Bind the local host address to the socket using the bind function.
* Within a for loop, accept connection requests from the client using the accept function.
* Use the fork system call to spawn the processes.
* Calculate the current date and time using the ctime() function. Change the format so that it is appropriate for human readable form and send the date and time to the client using the write function.

Client

* Include the necessary header files.
* Create a socket using the socket function with family AF\_INET, type as SOCK\_STREAM.
* Initialize server address to 0 using the bzero function.
* Assign the sin\_family to AF\_INET.
* Get the server IP address from the console.
* Using gethostbyname function assign it to a hostent structure, and assign it to sin\_addr of the server address structure.
* Request a connection from the server using the connect function.
* Within an infinite loop, receive the date and time from the server using the read function and print the date and time on the console.

## Code & Output-

1. Server

#include<time.h> #include<sys/types.h> #include<sys/socket.h> #include<unistd.h> #include<stdio.h> #include<string.h> #include<netinet/in.h> #include<netdb.h>

int main(int argc,char \*argv[])

{

int sd,ad;

char buff[1024];

struct sockaddr\_in servaddr,cliaddr;

//socklen\_t clilen=sizeof(cliaddr); time\_t t1; bzero(&servaddr,sizeof(servaddr));

/\*Socket address structure\*/ servaddr.sin\_family=AF\_INET; servaddr.sin\_addr.s\_addr=htonl(INADDR\_ANY); servaddr.sin\_port=htons(1507);

/\*TCP socket is created, an Internet socket address structure is filled with wildcard address & server’s well known port\*/ sd=socket(AF\_INET,SOCK\_STREAM,0);

/\*Bind function assigns a local protocol address to the socket\*/ bind(sd,(struct sockaddr\*)&servaddr,sizeof(servaddr));

/\*Listen function specifies the maximum number of connections that kernel should queue for this socket\*/

listen(sd,5);

printf("Server is running...\n");

/\*The server to return the next completed connection from the front of the completed connection Queue calls it\*/

ad=accept(sd,(struct sockaddr \*)NULL,NULL); while(1)

{

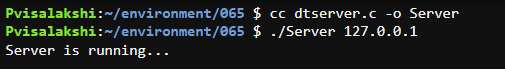
bzero(&buff,sizeof(buff));

/\*Library function time returns the Coordinated Universal Time\*/ t1=time(NULL);

/\*Prints the converted string format\*/ snprintf(buff,sizeof(buff),"%24s\r\n",ctime(&t1)); send(ad,buff,sizeof(buff),0);

}

}



1. Client

#include<stdio.h> #include<sys/types.h> #include<sys/socket.h> #include<netdb.h> #include<netinet/in.h> #include<unistd.h> #include<time.h>

int main(int argc,char \*argv[])

{

int sd,ad;

char buff[1024];

struct sockaddr\_in cliaddr,servaddr; struct hostent \*h; h=gethostbyname(argv[1]); bzero(&servaddr,sizeof(servaddr));

/\*Socket address structure\*/ servaddr.sin\_family=AF\_INET;

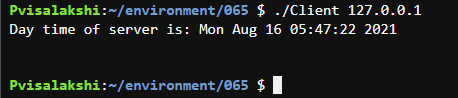
memcpy((char\*)&servaddr.sin\_addr.s\_addr,h->h\_addr\_list[0],h->h\_length); servaddr.sin\_port=htons(1507);

/\*TCP socket is created, an Internet socket address structure is filled with wildcard address & server’s well known port\*/ sd=socket(AF\_INET,SOCK\_STREAM,0);

/\*Connect establishes connection with the server using server IP address\*/ connect(sd,(struct sockaddr\*)&servaddr,sizeof(servaddr)); recv(sd,buff,sizeof(buff),0);

printf("Day time of server is: %s\n",buff);

}



The time shown is of the AWS Cloud 9 console.

## Result:

Thus, the concurrent daytime client- server communication is established by sending the request message from the client to the concurrent server and the server sends its time to all the clients and displays it.