**CONCURRENT MULTIPROCESSING TCP SERVER PROGRAM**

**WITH ONE PROCESS PER REQUEST:**

**OVERVIEW:** In this program we have designed a SERVER part for concurrent-Multi processing tcp connection. First the Server socket is created. Then we call bind function to the associated port number. After which the socket would be placed in the listening mode to accept new incoming connections from the clients. Then the server would create number of process after connection is accepted. In this procedure the cost of creating the process is not paid off before. Whenever the server is ready to accept new connections from the client it creates one process per request sent by the client. Then this process performs the function needed by the client. The function of the client is to fetch the file that is present in the server. First the server would display list of files that are present in the sever. Then client would choose the file that has to be transferred from the server to the client. All the data is put in the send buffer and is sent to the client side. Then in the client side it would store the file in the local machine.

--------------------------------------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------------------------------------

**MAIN:**

The main part receives 3 arguments from the users. The First one would be the port number through which the server is configured to listen for the new connection. And the second and the third argument would take the file name present in the server where the actual data is present. Then creating the new socket by taking its family name, type of the of the data being send like SOCK\_STREAM or SOCK\_DGRAM. Then after that socket is created successfully. Input all the necessary corresponding values to server structure. Use bind function by passing the server structure and the size of the client. After successful binding process, the port number which to which the server has been configured. Then we have used a for loop to keep the sever waiting for a new connection from the port number, previously that had been confirmed earlier. After a waiting for the new connection, then the server would find a new request and call the accept function, to accept a new connection from the client. As this is Multi processing program, after the connection is accept each new request is handled by the process created by the fork(). In the doit() function all the functionalities of the original program had before. The only main difference is the each process would handle all the connection separately as well as Using the write function send all the names of the files that are residing in file server. Then client would enter a valid result or valid file name. Then this file name input from the client is checked by the server to open corresponding file using the fopen() function. Then the contents of the file is transferred to buffer. Using the write() or send() we can send the contents in the buffer to the client side. Then client would save the file in local system.

**Execution in Command line:**

gcc -o tcp\_server tcp\_server.c

./tcp\_server 10000 File.txt Fil2.txt