National University of Computer and Emerging Sciences



**Laboratory Manuals**

*for*

**Computer Networks**

(CL -307)

Department of Computer Science

FAST-NU, Lahore, Pakistan

**Lab Manual 07**

# Objective:

* TCP Socket Programming using Multithreaded Server to handle multiple clients at the same time

**Lab Statement 1**: Multithreaded ECHO server using TCP (10)

You are required to design a **Multi-Threaded Echo Server and a Simple client**. The server uses a TCP protocol to connect to clients. Server will be listening for clients to connect to it and as soon as a client connects, it assigns a separate thread for further processing. The thread will be responsible to receive the data from the client and *echo* it to the client until the client sends the "DISCONNECT" command. **The server can handle maximum 3 clients at a time.**

Client will be a simple program which after connecting to the server will take the input from the user and send it to the server, then outputs the response on the terminal received from the server. It will do the same until user enters “DISCONNECT”. Upon entering “DISCONNECT” the client shall close the socket and exit.

**Following are the steps which Server should perform:**

1. Receives a connection request from client and pass the socket descriptor returned by the accept() to the thread and goes back to listen for more connections for **clients< 4**. If the fourth client tries to connect then server sends the client message that “**Server Full**”
2. Meanwhile this is what the thread do
   * Receive what the client sends.
   * Echo back what client sends.
   * If client has sent “DISCONNECT” then close the socket and quit.
3. Receive more data from the same client.

You should cater 3multiple clients that will be sending connection requests to server.

**Following are the steps which Clients should perform:**

1. Take input from the user.
2. Send input to the server
3. If input is “DISCONNECT” then close the socket and exit otherwise continue to step 4.
4. Wait for server’s response.
5. Print the server’s repose.
6. Go back to step 1.

**Lab Statement 2**: Online Voting System (10)

**ONLINE VOTING SYSTEM**

You are to devise an online vote casting system using **TCP** multithreaded server-client program. The voters will log on from client systems and cast their votes. Note that

One client can cast a vote ONCE AND ONLY ONCE.

A list of registered voters is available in a text file in the following format:

**Name/CNIC number**

System should be secure and should only allow clients with authentic name and CNIC to cast the vote.

The Server shall

* First of all take the client’s name and CNIC and match it with the content of the text file (Voters\_List.txt).
* If the details match, then it will welcome the voter and **display the names and poll symbols of the candidates on the terminal**.(Candidates\_List.txt)
* The voter will then cast the vote by **MENTIONING THE POLL SYMBOL (not the name) OF THE CANDIDATE. All these details are to be recorded in an output text file.**
* This system should work perfectly for **at least 5 different clients.**

**Note:**

1. You should be familiar with basic file streaming commands in C.
2. The server-client code **should be in C** and only C language. No other language is accepted
3. Student shall be awarded zero in the lab and -5 absolute if plagiarism is found.

**Very important**:

Make sure you add the proper header file **<pthread.h>** to make use of threading in your C language code.

At the compilation time, you must enter **gcc –pthread server.c -o server.out.** Otherwise your code will give errors.

**Pthread function syntax:**

* **int pthread\_create(pthread\_t \*thread, pthread\_attr\_t \*attr, void \*(\*start\_routine)(void \*), void \*arg);**

1. pthread\_t \*thread: the actual thread object that contains pthread id
2. pthread\_attr\_t \*attr: attributes to apply to this thread(use NULL for default attributes)
3. void \*(\*start\_routine)(void \*): the function this thread executes
4. void \*arg: arguments to pass to thread function above

* **voidpthread\_exit(void\*value\_ptr)**

pthread\_exit() terminates the thread and provides the pointer \*value\_ptr available to any pthread\_join() call.

* **intpthread\_join(pthread\_t thread, void\*\*value\_ptr);**

pthread\_join() suspends the calling thread to wait for successful termination of the thread specified as the first argument pthread\_t thread with an optional \*value\_ptr data passed from the terminating thread's call to pthread\_exit().