|  |  |  |
| --- | --- | --- |
|  | **Namal Institute Mianwali** |  |
| Computer Networks Laboratory Manual #7 |
|  |
| **Network Programming using UNIX TCP/IP Sockets in C**  **Part-1 : Iterative vs Concurrent Server** | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Course Title | Computer Networks | Course Number | CS – 270 L |
| Instructor | Shahzad Arif  shahzad.arif@namal.edu.pk | Lab Engineer | Asad Majeed  [asad.majeed@namal.edu.pk](mailto:asad.majeed@namal.edu.pk) |

# 1. Introduction:

In the previous Laboratory exercises, we learnt how the network is implemented for a few services. What if we want to write our own application? In this Lab, we shall develop our knowledge of the Application Programmer Interface (API) used to interact with the Operating System, to pass messages over the network using SOCKETS in C language.

Higher level languages such as Python, Java, .NET etc wrap these functions and use them. So an understanding of the functions helps understand the

# 2. Objective:

* To learn basic difference between an iterative and a concurrent server.
* To be able to comprehend underlying implementation issues associated with function use.

# 3. Background:

Figure(1) is an example of simple Server which runs from top to down, and shuts down after the execution of code. In reality, server is supposed to be up and running all the time. A server which could serve more than one client one after the other, is called an “iterative server”. The server does not close its execution, but is looped back to accepting new connections. The iterative server is depicted in Figure (2).

In another way to serve multiple clients, is to use the phenomenon of concurrency. In Unix, forking a process allows to invoke a child process to cater to the needs of client. In this approach, the system calls are not blocked nor the clients have to wait for the first client to complete its transactions.

Figure 1: A simple client-server flow chart.

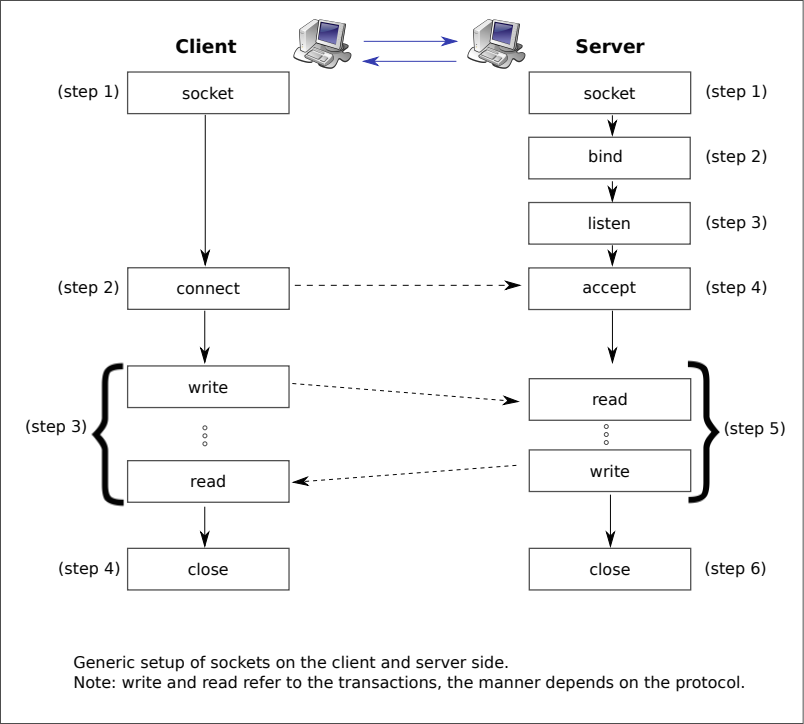


Figure 2: Iterative Server flow chart

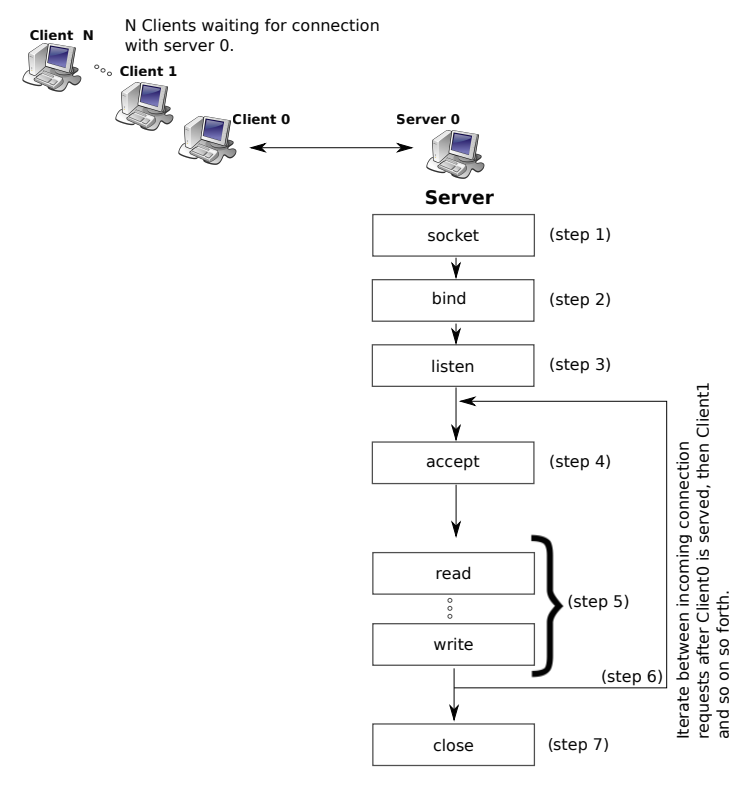
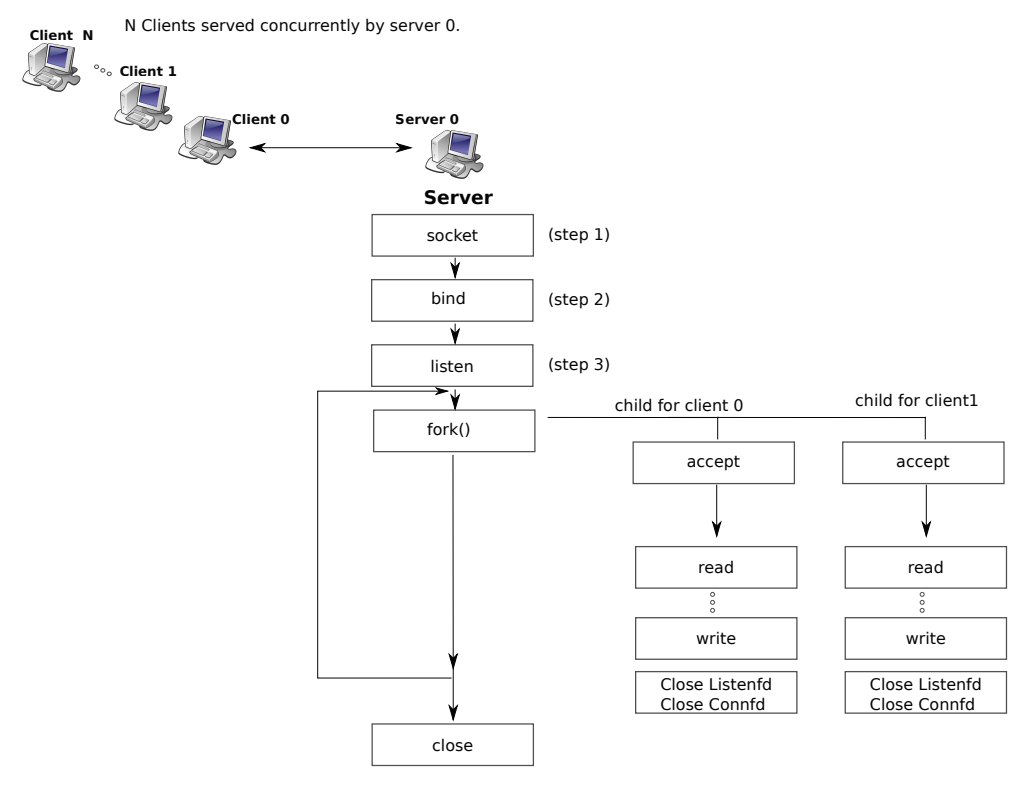


Figure 3: Concurrent processes for serving each client in a child process.



# 4. Activities:

## 4.1 Activity A:

**Observe the “server.c” program file that you have used in Lab 6 and identify whether it is an iterative server or not give your reason.**

**Observation and Reason:**

* In lab 6, server’s.c file has an iterative server code. This server performs/completes the request as they come into Queue. After completing one request come to another request.
* In concurrent processes, each process is given a time in the child process to complete. In a concurrent process, every process gives time to completion, not waiting for process completion to start the next process.
* In parallelism multiple processes run on multiple cores of a processor.

## 4.2 Activity B:

**Implement the concurrent version of the server by modifying the “server.c” program file.**

The answer file attached has a “concurrent version” code present.

Hint: Hint will be given in the class

**End of Activities.**