**ASSIGNMENT NO. 1**

**AIM:** **Design a distributed application which consist of a server and client using threads.**

**OBJECTIVE**:

 Understand the working of sockets and threads.

 Understand how sockets are used to communicate among processes.

 Learn the use of sockets using C under Linux and about Java sockets.

**PROBLEM STATEMENT:** This assignment involves the implementation of the Client and Server communication on single and multiple host using threads and socket programming.

**THEORY: Socket functions:**

Client Side steps:

**1.** socket()

**2.** connect()

**3.** send()

**4.** recv()

**5.** close()

Server Side steps:

**1.** socket()

**2.** bind()

**3.** listen()

**4.** accept()

**5.** recv()

**6.** send()

**7.** close()

The steps involved in establishing a socket on the client side are as follows:

 Create a socket with the socket() system call

 Connect the socket to the address of the server using the connect() system call

 Send and receive data. There are a number of ways to do this, but the simplest is to use the read() and write() system calls.

The steps involved in establishing a socket on the server side are as follows:

 Create a socket with the socket() system call

 Bind the socket to an address using the bind() system call. For a server socket on the Internet, an address consists of a port number on the host machine.

 Listen for connections with the listen() system call

 Accept a connection with the accept() system call. This call typically blocks until a client connects with the server.

 Send and receive data

**Java Socket Programming**

Java Socket programming is used for communication between the applications running on different JRE.

Java Socket programming can be connection-oriented or connection-less.

Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming.

The client in socket programming must know two information:

1. IP Address of Server, and

2. Port number.

**Socket class**

A socket is simply an endpoint for communications between the machines. The Socket class can be used to create a socket.

|  |  |
| --- | --- |
| **Important methods Method** | **Description** |
| 1) public InputStream getInputStream() | Returns the InputStream attached with this socket. |
| 2) public OutputStream getOutputStream() | Returns the OutputStream attached with this socket. |
| 3) public synchronized void close() | Closes this socket |

**ServerSocket class**

The ServerSocket class can be used to create a server socket. This object is used to establish communication with the clients

|  |  |
| --- | --- |
| **Important methods Method** | **Description** |
| 1) public Socket accept() | returns the socket and establish a connection between server and client. |
| 2) public synchronized void close() | closes the server socket. |

**C Socket Programming**

At the beginning, a socket function needs to be declared to get the socket descriptor.

int socket(int domain, int type, int protocol)



Next, decide which struct needs to be used based on what domain is used above.





On client side, serv\_addr.sin\_port = htons(127.0.0.1) is declared in order to listen internal network.

The flow chart below shows the interaction between client and server. The flow chart might looks complicated but make sure you don’t lost your patient due to the following flow chart. Because every process on the flow chart is needed and it acts a very important roles on network connection.



After all setup on struct sockaddr\_in is done, declare bind function. As flow chart, bind function must be declared on both server and client.



Server and client will start interact with each other after the bind function and it is the most important session. From what flow chart shows, **listen**, **accept**, **connect**, three function play a very important roles.

Imagine that server looks like an ATM, and only one person can be used the ATM. So, what happen if there is 2 or more people come at one time? The answer is simple, lining up and wait the front people finished using with ATM. It is exactly same as what happening in server.

Listen function acts as waiting room, asking the traffic wait on the waiting room. Accept function acts as person who asking the traffic waiting inside the waiting room to be ready for the meeting between server. Last, connect function acts as the person who want to carry out some work with server.

**listen function**







Finally, after the request accepted. What should server and client do is send and read data. It is the simplest part in whole of this article. **read** function used to read the buffer data and **write** function used to send the data. That’s all.

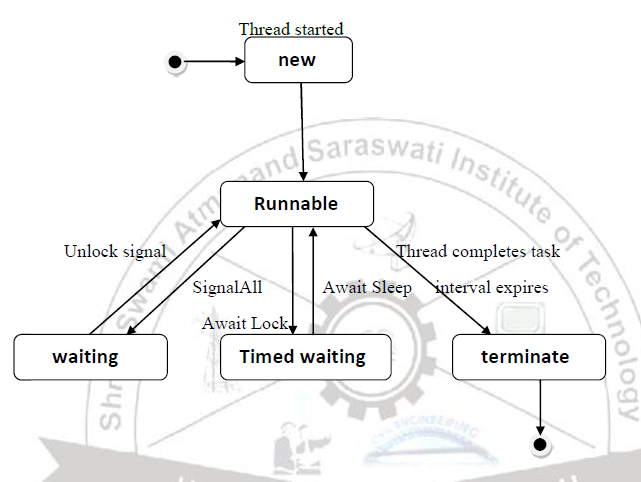




**Difference between a Thread and Process**

|  |  |
| --- | --- |
| **Thread** | **Process** |
| Threads (Light weight Processes) share the address space of the process that created it | processes have their own address. |
| Threads have direct access to the data segment of its process | processes have their own copy of the data segment of the parent process |
| Threads can directly communicate with other threads of its process | processes must use inter process communication to communicate with sibling processes |
| Threads have almost no overhead | processes have considerable overhead. |
| New threads are easily created | new processes require duplication of the parent process. |
| Threads can exercise control over threads of the same process | processes can only exercise control over child processes |
| Changes to the main thread (cancellation, priority change, etc.) may affect the behavior of the other threads of the process | changes to the parent process do not affect child processes |

**Thread LifeCycle:**

****

**Thread has Five different states:**

 **New:** A new thread begins its life cycle in the new state. It remains in this state until the program starts the thread. It is also referred to as a born thread.

 **Runnable:** After a newly born thread is started, the thread becomes runnable. A thread in this state is considered to be executing its task.

 **Waiting:** Sometimes a thread transitions to the waiting state while the thread waits for another thread to perform a task.A thread transitions back to the runnable state only when another thread signals the waiting thread to continue executing

 **Timed waiting:** A runnable thread can enter the timed waiting state for a specified interval of time. A thread in this state transitions back to the runnable state when that time interval expires or when the event it is waiting for occurs.

 **Terminated:** A runnable thread enters the terminated state when it completes its task or otherwise terminates

Write the different way to create thread using java.

By Implementing Runnable interface

To create thread using Runnable interface, a class first need to instantiate an object of type Thread from within that class. Thread defines several constructors. The one that we will use is shown here:

**Thread(Runnable threadOb, String threadName);**

Here threadOb is an instance of a class that implements the Runnable interface and the name of the new thread is specified by threadName.

After the new thread is created, we need to start the execution of thread.It is done using its **start( )** method, which is declared within Thread.

**void start( );**

we can define the code that constitutes the new thread inside run() method. run() method can call other methods, use other classes, and declare variables, just like the main thread can.

**public void run( )**

By extending the Thread class

To create a new thread first need to extend **Thread** Super class and create an instance of that class. The newly created(extended) class must override the **run()** method, which is the entry point for the new thread. It must also call **start( )** to begin execution of the new thread. Same as Runnable Interface

**CONCLUSION:** Thus, we have implemented given problems using Client and Server communication on single and multiple host using threads and socket programming.