**Practical No. 5**

**Aim:** To implement a stateful file server

**Theory:**

In computing, a file server (or fileserver) is a computer attached to a network that provides a location for [shared disk access](https://en.wikipedia.org/wiki/Shared_disk_access), i.e. shared storage of computer files (such as documents, sound files, photographs, movies, images, databases, etc.) that can be accessed by the workstations that are able to reach the computer that shares the access through a computer network. The term server highlights the role of the machine in the [client–server](https://en.wikipedia.org/wiki/Client%E2%80%93server) scheme, where the clients are the workstations using the storage. It is common that a file server does not perform computational tasks, and does not run programs on behalf of its clients. It is designed primarily to enable the storage and retrieval of data while the computation is carried out by the workstations.

File servers are commonly found in schools and offices, where users use a [LAN](https://en.wikipedia.org/wiki/Local_area_network) to connect their client computers.

**Types of File Servers**

A file server may be dedicated or non-dedicated. A dedicated server is designed specifically for use as a file server, with workstations attached for reading and writing files and databases.

File servers may also be categorized by the method of access: Internet file servers are frequently accessed by [File Transfer Protocol](https://en.wikipedia.org/wiki/File_Transfer_Protocol) (FTP) or by HTTP (but are different from [web servers](https://en.wikipedia.org/wiki/Web_server), that often provide dynamic web content in addition to static files). Servers on a LAN are usually accessed by [SMB](https://en.wikipedia.org/wiki/Server_Message_Block)/[CIFS](https://en.wikipedia.org/wiki/CIFS) protocol ([Windows](https://en.wikipedia.org/wiki/Microsoft_Windows) and [Unix-like](https://en.wikipedia.org/wiki/Unix-like)) or [NFS](https://en.wikipedia.org/wiki/Network_File_System_(protocol)" \o "Network File System (protocol))protocol (Unix-like systems). [Database servers](https://en.wikipedia.org/wiki/Database_server), that provide access to a shared database via a database device driver, are not regarded as file servers as they may require [Record locking](https://en.wikipedia.org/wiki/Record_locking).

**Stateful vs StatelessServer**

* A stateful server remembers client data from one request to another
* A stateless server keeps no state information
* Using a stateless file server, the client must specify complete file names in each request specify location for reading or writing, re-authenticate for each request
* Using a stateful server , the client can send less data with each request
* A statefull server is simpler
* On the other hand a stateless server is more robust. Lost connections can’t leave a file in an invalid state. Rebooting the server does not loose state information. Rebooting the client does not confuse a stateless server.

**Design of File Servers**

The design of file servers is complicated by competing demands for storage space, access speed, [recoverability](https://en.wikipedia.org/wiki/Recoverability), ease of administration, [security](https://en.wikipedia.org/wiki/Security), and budget. This is further complicated by a constantly changing environment, where new hardware and technology rapidly obsolesces old equipment, and yet must seamlessly come online in a fashion [compatible](https://en.wikipedia.org/wiki/Computer_compatibility) with the older machinery. To manage [throughput](https://en.wikipedia.org/wiki/Throughput), peak loads, and [response time](https://en.wikipedia.org/wiki/Responsiveness), [vendors](https://en.wikipedia.org/wiki/Vendor) may utilize queuing theory to model how the combination of hardware and software will respond over various levels of demand. Servers may also employ dynamic [load balancing](https://en.wikipedia.org/wiki/Load_balancing_(computing)) scheme to distribute requests across various pieces of hardware.

The primary piece of hardware equipment for servers over the last couple of decades has proven to be the [hard disk drive](https://en.wikipedia.org/wiki/Hard_disk_drive). Although other forms of storage are viable (such as [magnetic tape](https://en.wikipedia.org/wiki/Magnetic_tape) and [solid-state drives](https://en.wikipedia.org/wiki/Solid-state_drives)) disk drives have continued to offer the best fit for cost, performance, and capacity.

**Program:**

1. **dc5s.java**

import java.io.\*;

import java.net.\*;

public class dc5s

{

private static Socket socket;

public static void main(String[] args) throws IOException

{

try

{

int port = 1024;int count=0,a=0;

ServerSocket sSocket = new ServerSocket(port);

System.out.println("Server Started and listening to the port 1024");

while(true)

{

socket = sSocket.accept();

InputStream is = socket.getInputStream();

InputStreamReader isr = new InputStreamReader(is);

BufferedReader br = new BufferedReader(isr);

String file = br.readLine();

String dataprint;

System.out.println("file received from client is "+file);

//File transferFile =new File ("abc.txt");

byte [] bytearray = new byte [1024];

OutputStream os = socket.getOutputStream();

FileInputStream fin =new FileInputStream("F:/code.c");

BufferedInputStream bin = new BufferedInputStream(fin);

do

{ count++;

bin.read(bytearray,0,count\*10);

dataprint = new String(bytearray);

//OutputStream os = socket.getOutputStream();

System.out.println("Sending Files...");

System.out.println(dataprint);

os.write(bytearray,0,count\*10);

os.flush();// os.close();

System.out.println("Want to send more?(0-yess||1-no)");

BufferedReader brd=new BufferedReader(new InputStreamReader(System.in));

a=Integer.parseInt(brd.readLine());

}while (a==0);

if(a==1) break;

}

//socket.close();

System.out.println("File transfer complete");

}

finally

{

{

socket.close();

}

//catch(Exception e){}

}

}

}

1. **dc5c.java**

import java.io.\*;

import java.net.\*;

public class dc5c

{

private static Socket socket;

public static void main(String args[]) throws IOException

{

try

{

int filesize=1022386;

byte [] bytearray = new byte [filesize];

int bytesRead;

int current = 0,count=1,a=0,cc=0;

String host ="10.21.21.32";

String datawrite;

BufferedWriter bww = null;

FileWriter fw = null;

int port = 1024;

InetAddress address = InetAddress.getByName("localhost");

socket = new Socket(address, port);

//Send the message to the server

OutputStream os = socket.getOutputStream();

OutputStreamWriter osw = new OutputStreamWriter(os);

BufferedWriter bw = new BufferedWriter(osw);

String file = "abc.txt";

String sendMessage = file + "\n";

bw.write(sendMessage);

bw.flush();

System.out.println("Message sent to the server : "+sendMessage);

while(a==0)

{

//Get the return message from the server

InputStream is = socket.getInputStream();

FileOutputStream fos = new FileOutputStream("E:/abcr.txt");

BufferedOutputStream bos = new BufferedOutputStream(fos);

fw = new FileWriter("E:/abcr.txt",true);

bww = new BufferedWriter(fw);

bytesRead = is.read(bytearray,0,bytearray.length);

current = bytesRead;

//bos.write(bytearray, cc , 10);

datawrite = new String(bytearray);

bww.write(datawrite);

cc=cc+10;

count++;

bos.flush();

bos.close();

System.out.println("Want to receive more?(0-yess||1-no)");

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

a=Integer.parseInt(br.readLine());

}

}

//catch (Exception e)

//{

// e.printStackTrace();

//}

finally

{

//Closing the socket

//try

{

socket.close();

}

// catch(Exception e)

{

// e.printStackTrace();

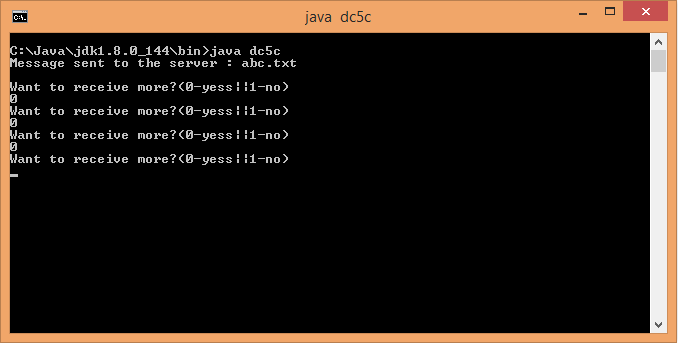
}

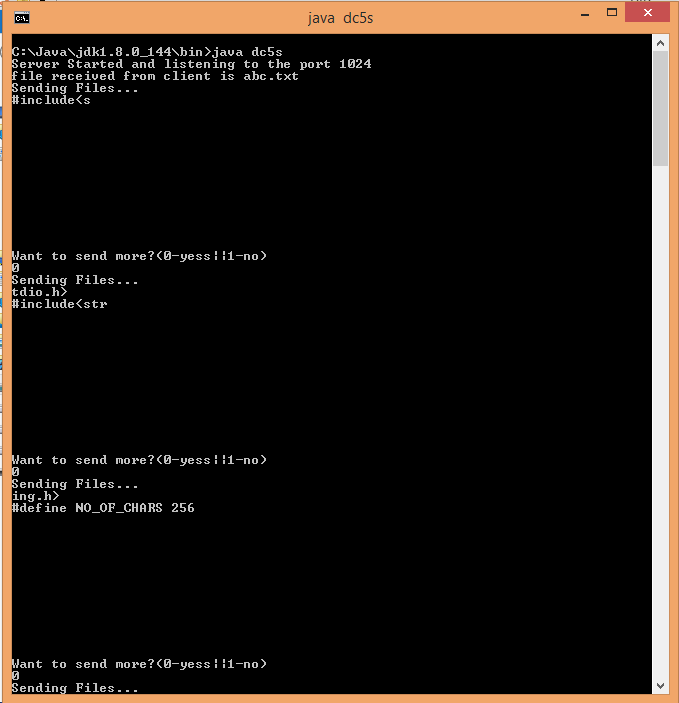
}

}

}

**Output:**

****

****

**Concllusion:** A stateful file server has been implemented.