<table>
<thead>
<tr>
<th>Experiment No</th>
<th>Aim of Experiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Study of networking devices.</td>
</tr>
<tr>
<td>02</td>
<td>Implementation of client-server using socket programming.</td>
</tr>
<tr>
<td>03</td>
<td>Implementation of echo server.</td>
</tr>
<tr>
<td>04</td>
<td>Implementation of chat server</td>
</tr>
<tr>
<td>05</td>
<td>Implementation of File Transfer Protocol (FTP) server data illustration.</td>
</tr>
<tr>
<td>06</td>
<td>Implementation of Remote Method Invocation (RMI).</td>
</tr>
<tr>
<td>07</td>
<td>Implementation of TCP/IP.</td>
</tr>
<tr>
<td>08</td>
<td>Implementation of UDP.</td>
</tr>
<tr>
<td>09</td>
<td>Study of Windows 2003/2008 system architecture.</td>
</tr>
<tr>
<td>12</td>
<td>Windows 2003/2008-Study &amp; configuration of DHCP service with static as well as dynamic address binding.</td>
</tr>
<tr>
<td>13</td>
<td>Installation of Linux basic configurations and management of users and groups.</td>
</tr>
<tr>
<td>14</td>
<td>Configuration of FTP and Secure Shell in Linux.</td>
</tr>
<tr>
<td>15</td>
<td>Configuration of DNS and DHCP in Linux</td>
</tr>
<tr>
<td>16</td>
<td>Configuration of HTTP server in Linux</td>
</tr>
<tr>
<td>17</td>
<td>Study of Windows 2008 Security system.</td>
</tr>
<tr>
<td>18</td>
<td>Study of Windows 2008 I/O system</td>
</tr>
</tbody>
</table>
EXPERIMENT NO. - 1

TITLE : - Study of networking devices.

AIM : - To study basics of Networking architectures & devices.

OBJECTIVE: - To learn about network-basics, models, architectures and devices.

THEORY:-

Hardware Devices –
1. Switches
2. Hubs
3. Gateways
4. Router

1. Switch

Layer 2 switching is hardware based, which means it uses the media access control address (MAC address) from the host is network interface cards (NICs) to decide where to forward frames. Switches use application –specific integrated circuits (ASICs) to build and maintain filter tables (also known as MAC address tables).One way to think of a layer 2 switch is as a multiport bridge.

Layer 2 switching provides the following.
- Hardware-based bridging(MAC)
- Wire speed
- High speed
- Low latency
- Low cost

Layer 2 switching is highly efficient because there is no modification to the data packet. Only to the frame encapsulation of the packet, and only when the data packet is passing through dissimilar media (such as from Ethernet to FDDI).Layer 2 switching is used for workgroup connectivity and network segmentation (breaking up collision domains).This allows a flatter network design with more network segments than traditional 10 Base T shared network. Layer 2 switching has helped develop new components in the network infrastructure.
- Server frames --- Servers are no longer distributed to physical locations because virtual LANs can be created to broadcast domains in a switched internetwork. This means that all servers can be placed in a central location, yet a certain server can still be part of a workgroup in a remote branch, for example.
- Intranets--- Allows organization –wide client/server communication based on a Web technology.

These new technologies are allowing more data to flow off of local subnets and onto a routed network, where a router is performance can become the bottleneck.

**Limitations**

Layer 2 switches have the same limitations as bridge network. Remember that bridges are good if a network is designed by the 80/20 rule: uses spend 80 percent of their time on their local segments.

Bridges network break up collision domains, but the network one large broadcast domain. Similarly, layer 2 switches (bridges) cannot breakup broadcast domain, which can cause performance issues and limits the size of your network. Broadcast and multicasts, along with the slow convergence of spanning tree, can cause major problems as the network grows. Because of these problems, layer 2 switches cannot completely replace routers in the internetwork.

2. **Hub:**

A network hub or repeater hub is a devices for connecting multiple twisted pair or fiber optic Ethernet devices together, making them act as a single network segment. Hubs work at the physical layer(layer 1) of the OSI model. The devices are thus a form of multiport repeater. Repeater hubs also participate in collision detection, forwarding a jam signal to all ports if it detects a collision. Hubs also often come with a BNC and/or AUI connector to allow connection to legacy 10BASE2 or 10BASE5 network segments. The availability of low-priced network switches has largely rendered hubs obsolete but they are still seen in older installations and more specialized application.
**Technical Information**

A network hub is a fairly unsophisticated broadcast device. Hubs do not manage any of the traffic that comes through them and any packet entering any port is broadcast out on every other port (other than the port of entry). Since every packet is being sent out through every other port, packet collision results—which greatly impedes the smooth flow of traffic.

The need for hosts to be able to detect collisions limits the number of hubs and the total size of the network. For 10 M bit/s network, up to 5 segments, the limit is reduced to 3 segments (2 hubs) between any two end stations, and even that is only allowed if the hubs are of the low delay variety. Some hubs have special (and generally manufacturer specific) stack ports allowing them to be combined in a way that allows more hubs than simple chaining through Ethernet cables, but even so a large Fast Ethernet network is likely to require switches to avoid the chaining limits of hubs.

Most hub detect typical problem, such as excessive collisions on individual ports, and partition the port, disconnecting it from the shared medium. Thus hub-based Ethernet is generally more robust than coaxial cable-based Ethernet. Where a misbehaving device can disable the entire segment. Even if not partitioned automatically, a hub makes troubleshooting easier because status lights can indicate the possible problem source.

3. **Gateways**-

There are multiple devices that have been described as "residential gateways" each with a different function. Each type of device allows the connection of a LAN (used in the home) to a WAN. The WAN can often be the Internet or can merely be a larger LAN of which the home is a part (such as a municipal WAN that provides connectivity to the residences within the municipality). WAN connectivity may be provided through DSL, cable modem, a broadband mobile phone network, or other connections.

The term "residential gateway" was originally used to distinguish the inexpensive networking devices designated for use in the home from similar devices used in corporate LAN environments (which generally offered a greater array of capabilities). In recent years, however, the less expensive "residential gateways" have gained many of the capabilities of corporate gateways and the distinctions are fewer.
Many home LANs now are able to provide most of the functions of small corporate LANs.

Therefore the term "residential gateway" was becoming obsolete and merely implies a less expensive, lower capability networking device.

Nowadays, the home gateway tends to have abundant interfaces, powerful functions and a more user-friendly interface. It is a manageable terminal with auto-configuration, and multi-service perceiving and bearing. The home gateway provides Quality of Service to simultaneously support different types of services. As a part of the carrier network, the home gateway shall support remote control, detection and configuration.

4. Router-

A device that forwards data packets along networks. A router is connected to at least two networks, commonly two LANs or WANs or a LAN and its ISP’s network. Routers are located at gateways, the places where two or more networks connect.

. Routers use headers and forwarding tables to determine the best path for forwarding the packets, and they use protocols such as ICMP to communicate with each other and configure the best route between any two hosts. Very little filtering of data is done through routers.

CONCLUSION :- Different networking devices are useful to perform communication effectively.
EXPERIMENT NO. – 2

**TITLE**: - Implementation of client - server using socket programming.

**AIM** : - To implement client – server using socket programming

**OBJECTIVE**: - To understand network programming using socket programming

**THEORY**:

**Class Socket (TCP Client Connections)**

A socket is a Java representation of TCP network connections. In order to communicate with a remote host the java client must first create a Socket, which will establish the TCP connection. In doing so a host name and port number must be specified. There must be a server actively listening on the specified port or the connection will fail with IO Exception.

These constructors allow the Socket connection to be established.

**Socket(String host, int port) throws IO Exception:**

This creates a Socket and connects to the specified host and port. Host can be a host name or IP address and port must be in a range of 1-65535.

**Socket(InetAddress address, int port) throws IO Exception:**

This creates a Socket and connects to the specified port of the host address. The port must be in a range of 1-65535.

These methods allow the remote host address and local or remote port numbers to be identified. These methods also allow for the creation of input and output streams.

**InetAddress getInetAddress():**

This method returns the IP address of a remote host.

**int get Port():**

This method returns the port number of the remote host to which the Socket is connected.
int getLocalPort():

    The local port number is returned by this method, the port number used to create the socket.

InputStream getInputStream() throws IOException:

    This method returns an InputStream that allows the Socket to receive data across the TCP connection. An InputStream can be buffered or standard.

OutputStream getOutputStream() throws IOException:

    This method returns an OutputStream that allows the Socket to send data across the TCP connection. An OutputStream should be buffered to avoid lost bytes, especially when the Socket is closed.

void close() throws IOException:

    This method closes the Socket, releasing any network or system resources being used.

Exception Errors

IOException:

    This is a generic Input Output error that can be thrown by many of the method mentioned here.

DataOutputStream(OutputStream out):

    This constructs a DataOutputStream attached to a specified OutputStream. A data is broken down into byte format and transmitted to the attached stream.

DataInputStream(InputStream in):

    This constructs a DataInputStream attached to the specified InputStream. Byte high order format data is read from the InputStream and converted to low order data when passed to the attached DataInputStream.
Class Socket (TCP Client Connections) Sample code:

The following method, connects (int port), Accepts the port number and creates a socket to the host server. Once the Socket is created we must get the raw input and output streams associated with the Socket. These are called rawIn and rawOut. rawOut is then setup as a buffered. In order to understand these streams we must convert them to Data streams. The reason this is required is because the raw I/O streams can only understand byte format data, where as we wish to send strings and integers. Creating a DataInputStream from a raw InputStream allows the conversions to take place automatically.

Implementation of Client-Server Socket

CLIENT PROGRAM:
import java.io.*;
import java.awt.*;
import javax.swing.*;
import java.net.*;
import java.util.*;
import java.awt.event.*;

class chatc
{
    String msg ;
    String msg1;
    chatc()
    {
        try
        {
            Socket s = new Socket("localhost",2222);
BufferedReader br = new BufferedReader(new InputStreamReader(s.getInputStream()));
BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
PrintWriter pw = new PrintWriter(s.getOutputStream(), true);
msg = in.readLine();
while (true)
{
    pw.println(msg);
    if (msg.equals("bye"))
    {
        System.exit(0);
    }
    msg1 = br.readLine();
    System.out.println("Message From Server: " + msg1);
    msg = in.readLine();
}

} catch (Exception e) {
{}
}
class cli
{
    public static void main(String args[])
    {
        chatc ob = new chatc();
    }
}

* OUTPUT *

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>cd\
C:\>cd C:\NE_aparna
C:\NE_aparna>set path=C:\Program Files\Java\jdk1.5.0\bin
C:\NE_aparna>javac cli.java
C:\NE_aparna>java cli
Hi!
Message From Server: Hello!
How are you sister?
Message From Server: I am fine, How are you?
What's your result?
Message From Server: It's 97.64%
Congratulations!
Message From Server: What's your result?
It's 71.50%

Message From Server: Oh! That's good, Congrats to you also!!

See ya!

Message From Server: Ok, See ya!

SERVER PROGRAM:

```java
import java.io.*;
import java.awt.*;
import javax.swing.*;
import java.net.*;
import java.util.*;
import java.awt.event.*;

class chats
{
    String msg ;
    String msg1;
    chats()
    {
        try
        {
            ServerSocket ss = new ServerSocket(2222);
            Socket s = ss.accept();
            BufferedReader br=new BufferedReader(new InputStreamReader(s.getInputStream()));
            BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
            PrintWriter pw=new PrintWriter(s.getOutputStream(),true);
            msg=br.readLine();
            System.out.println(msg);
            pw.println(msg);
        }
    }
}
```

Prepared By: Mr. S.U.Mahamuni.
```java
while(true)
{
    if(msg.equals("bye"))
    {
        System.exit(0);
    }
    System.out.println("Message From Client: " + msg);
    msg1=in.readLine();
    pw.println(msg1);
    msg=br.readLine();
}
}
catch (Exception e)
{
}
}
class ser
{
    public static void main(String args[])
    {
        chats ob = new chats();
    }
}
```
* OUTPUT *

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\Administrator>cd\nC:\>cd C:\NE_aparna
C:\NE_aparna>set path=C:\Program Files\Java\jdk1.5.0\bin
C:\NE_aparna>javac ser.java
C:\NE_aparna>java ser
Message From Client: Hi!
Hello!
Message From Client: How are you sister?
I am fine, How are you?
Message From Client: What's your result?
It's 97.64%
Message From Client: Congratulations!
What's your result?
Message From Client: It's 71.50%
Oh! That's good, Congrats to you also!!
Message From Client: See ya!
Ok, See ya!

CONCLUSION: With the help of socket primitive explained above we have developed the Chat client Server Program.
EXPERIMENT NO. - 3

**TITLE:** - Implementation of Echo server.

**AIM** : - To implement Echo Server

**OBJECTIVE:** - To understand network programming using socket programming

**THEORY:**-

Firstly Echo client connects to the Echo server. Echo server simply receives request from Echo client & echoes it back. Echo client creates a socket by getting connection from Echo server. This client program is straightforward because Echo server implements a simple protocol. Client sends text to Server & server echoes it back.

Procedures of program is-

a) Open a Socket.

b) Open an i/p stream.

c) Read from & write to the stream according to stream server protocol.

d) Close the program.

e) Close the Socket.

Only steps 3 differ from client to client depending on the server. Other steps remain same.

**Knock Knock Server**-

Server Program begins by creating a new server socket object to listen on a specific port. Knock Knock server listens on port 4444
Socket client Socket=null;
{
    Try
    {
        client_Socket=serverSocket.accepts;
    }
    Catch(IOException e)
    {
        System.out.println ("Accept failed:4444");
    }
}

Accept method waits until a client starts & requests a connection on host & port of this server. When connection is requested & successfully established, the accept method returns a new socket object which is bound to the same local port & has it's remote address & remote port.

Server communicates with client over that new socket & continues to listen for client connection request on the original server socket.

**Knock Knock Protocol**

Knock Knock protocol class implements the protocol that client & server use to communicate. Knock Knock server is designed to listen & handle a single connection request.
Supporting Multiple Client-

Client connection requests are queued at port, so the server must accept the connection sequentially. Server provides services to Client Simultaneously through the threads.

While(true)
{
    Accept connection;
    Create a thread to deal with the Client;
    End while;
}
Thread reads from & writes to the client connection as necessary.

CLIENT PROGRAM :
import java.io.*;
import java.awt.*;
import javax.swing.*;
import java.net.*;
import java.util.*;
import java.awt.event.*;
import java.awt.datatransfer.*;

class chatc
{
    String msg ;
    String msg1;
    chatc()
    {

```java
// msg = new String[20];
// msg1= new String[20];
try {
    Socket s = new Socket("localhost",2222);
    BufferedReader br=new BufferedReader(new InputStreamReader(s.getInputStream()));
    BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
    PrintWriter pw=new PrintWriter(s.getOutputStream(),true);

    msg=in.readLine();
    while(true)
    {
        pw.println(msg);
        if(msg.equals("bye"))
        {
            System.exit(0);
        }
        msg1=br.readLine();
        System.out.println("Message From Server: " + msg1);
        msg=in.readLine();
    }
} catch (Exception e) {
    }
```

class cli
{
    public static void main(String args[])
    {
        chatc ob = new chatc();
    }
}

* OUTPUT *

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\Administrator>cd\
C:/>cd C:\Program Files\Java\jdk1.5.0\bin
C:\Program Files\Java\jdk1.5.0\bin>javac cli.java
C:\Program Files\Java\jdk1.5.0\bin>java cli
hello!
Message From Server: hello!
How are you?
Message From Server: How are you?
SERVER PROGRAM :
import java.io.*;
import java.awt.*;
import javax.swing.*;
import java.net.*;
import java.util.*;
import java.awt.event.*;
import java.awt.*;
import java.awt.event.*;
class chats
{
    String msg ;
    String msg1;
    chats()
    {
        try
        {
            ServerSocket ss = new ServerSocket(2222);
            Socket s = ss.accept();
            BufferedReader br=new BufferedReader(new InputStreamReader(s.getInputStream()));
            PrintWriter pw=new PrintWriter(s.getOutputStream(),true);
            msg=br.readLine();
            while(true)
            {
                if(msg.equals("bye"))
                {
                    System.exit(0);
                }
                System.out.println("Message From Client: "+msg);
            }
        }
    }
}
pw.println(msg);
msg=br.readLine();
}

try {
    String msg = br.readLine();
    pw.println(msg);
    pw.flush();
} catch (IOException e) {
}
}
}

class ser
{
    public static void main(String args[])
    {
        chats ob = new chats();
    }
}

CONCLUSION:
We implemented Echo server using Socket primitive.
EXPERIMENT NO. - 4

**TITLE**: Implementation of Chat server.

**AIM**: To implement Chat Server using multithreading.

**OBJECTIVE**: To understand network programming using socket programming

**THEORY**:

The benefit of multithreading is that One thread can pause without stopping other parts of your program. For example, the idle time created when a thread reads data from a network or waits for user input can be utilized elsewhere.

**Thread Priorities**:

Priority that determines how that thread should be treated with respect to the others. Thread priorities are integers that specify the relative priority of one thread to another. As an absolute value, a priority is meaningless; a higher-priority thread doesn’t run any faster than a lower-priority thread if it is the only thread running. Instead, a thread’s priority is used to decide when to switch from one running thread to the next. This is called a *context switch*.

**Messaging**:

After you divide your program into separate threads, you need to define how they will communicate with each other. When programming with most other languages, you must depend on the operating system to establish communication between threads. Messaging system allows a thread to enter a synchronized method on an object, and then wait there until some other thread explicitly notifies it to come out. The Thread Class and the Runnable Interface multithreading system is built upon the Thread class, its methods, and its companion interface, Runnable. Thread encapsulates a thread of execution.

The Thread class defines several methods that help manage threads.

- `getName` - Obtain a thread’s name.
- `getPriority` - Obtain a thread’s priority.
- `isAlive` - Determine if a thread is still running.
- `join` - Wait for a thread to terminate.
- `run` - Entry point for the thread.
- `sleep` - Suspend a thread for a period of time.
- `start` - Start a thread by calling its run method.
The sleep() method causes the thread from which it is called to suspend execution for the specified period of milliseconds.

Its general form is shown here:
static void sleep(long milliseconds) throws InterruptedException

The number of milliseconds to suspend is specified in milliseconds. This method may throw an InterruptedException.

Interprocess communication mechanism is done via the wait(), notify(), and notifyAll() methods. These methods are implemented as final methods in Object, so all classes have them. All three methods can be called only from within a synchronized context.

wait() tells the calling thread to give up the monitor and go to sleep until some other thread enters the same monitor and calls notify().

notify() wakes up the first thread that called wait() on the same object.
notifyAll() wakes up all the threads that called wait() on the same object.
The highest priority thread will run first.

These methods are declared within Object, as shown here:
final void wait() throws InterruptedException
- final void notify()
- final void notifyAll()

Additional forms of wait() exist that allow you to specify a period of time to wait.

The Thread class also defines a method called stop() that stops a thread. Its signature is shown here:
final void stop()

Once a thread has been stopped, it cannot be restarted using resume().

While the suspend(), resume(), and stop() methods defined by Thread seem to be a perfectly reasonable and convenient approach to managing the execution of threads, suspend() can sometimes cause serious system failures. Assume that a thread has obtained locks on critical data structures. If that thread is suspended at that point, those locks are not relinquished. Other threads that may be waiting for those resources can be deadlocked.
The resume() method is also deprecated. It does not cause problems, but cannot be used without the suspend() method as its counterpart.
run( ) method periodically checks to determine whether that thread should suspend, resume, or stop its own execution. Typically, this is accomplished by establishing a flag variable that indicates the execution state of the thread. As long as this flag is set to “running,” the run( ) method must continue to let the thread execute. If this variable is set to “suspend,” the thread must pause. If it is set to “stop,” the thread must terminate.

Implementation of Chat Client-Server Socket (chat using threading concept)

SERVER PROGRAM :

```java
import java.net.*;
import java.io.*;
import java.util.*;
public class chatser extends Thread
{
    public static void main(String a[]){
        Socket cli;
        int counter=0;
        ServerSocket serv=new ServerSocket(4546);
        System.out.println("Waiting for new connection...");
        while(true)
        {
            cli=serv.accept();
            System.out.println("New Client No : "+(++counter));
            Thread th=new ThreadServer(cli,counter);
            th.start();
        }
    }
}
```

class ThreadServer extends Thread {
    Socket sock;
    int ID,clictr;

    public ThreadServer(Socket s,int no) {
        sock=s;
        ID=no;
    }

    public void run() {
        try {
            BufferedReader br=new BufferedReader(new InputStreamReader(sock.getInputStream()));
            PrintWriter pr=new PrintWriter(sock.getOutputStream(),true);
            BufferedReader br1=new BufferedReader(new InputStreamReader(System.in));
            PrintWriter pr1=new PrintWriter(System.out,true);
            boolean a=true;
            while(a) {
                String str=br.readLine();
                pr1.println("msg from client="+str);
                pr1.println("msg for client=");
                str=br1.readLine();
                pr.println(str);
            }
        } catch(Exception e) {} 
    }
}

* OUTPUT *

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\Administrator>cd\
C:\>cd C:\NE_aparna
C:\NE_aparna>set path=C:\Program Files\Java\jdk1.5.0\bin
C:\NE_aparna>javac chatser.java
C:\NE_aparna>java chatser
Waiting for new connection...
New Client No : 1
msg from client=Hi! I am client 1...
msg for client=
Good 1
New Client No : 2
msg from client=Hi! I am client 2..
msg for client=
Good 2
msg from client=Hello 2
msg for client=
Hi 2
msg from client=Hello 1
msg for client=
Hi 1

CONCLUSION :- With the help of multithreading we have implemented the Chat client Server Program.
EXPERIMENT NO. - 5


AIM: - To implement File Transfer Protocol (FTP)

OBJECTIVE: - To understand network programming using socket programming

THEORY:-

FTP runs exclusively over TCP. It defaults to listen on port 21 for incoming connections from FTP clients. A connection to this port from the FTP client forms the control stream on which commands are passed to the FTP server from the FTP client & on occasion from the FTP client to the FTP server. FTP uses out-of-band control, which means it uses a separate connection for control & data. Thus, for the actual file transfer to take place, a different connection is required which is called the data stream. Depending on the transfer mode, the process of setting up the data stream is different. Port 21 for control (or program) & port 20 for data.

In active mode, the FTP client opens the dynamic port. Since the FTP server the dynamic port number on which it is listening over the control stream and waits for a connection from the FTP server, when the FTP server initiates the data connection to the FTP client it binds the source port to port 20 on the FTP Server.

In order to use active mode, the client sends a PORT command, which the IP and port as argument. The format for the IP and port is “h1, h2, h3, h4, p1, p2”. Each field is a decimal representation of 8 bits of the host IP, followed by the chosen data port. For example, A client with an IP of 192.168.0.1, listening on port 49154 for the data connection will send the command “PORT 192,168,0,1,192,2”. The port fields should be interpreted as p1*256+p2= port, or, in this example, 192*256+2=49154.

In passive mode, the FTP server opens a dynamic port, sends the FTP client the servers IP address to connect to and the port on which it is listening (a 16 bit value broken into a high and low byte, as explained above) over the control stream and waits for a connection from the FTP client. In this case, the FTP client binds the source port of the connection to a dynamic port.
To use passive mode the client sends the PASV command to which the server could reply with something similar to “227 Entering Passive Mode (127, 0, 0, 1, 192, and 52)”. The syntax of the IP address and port are the same as for the argument to the port command.

In extended passive mode, the FTP server operates exactly the same as passive mode, however it only transmits the port number (not broken into high and low bytes) and the client is to assume that its connects to the same IP address that was originally connected to. Extended passive mode was added by RFC 2428 in September 1998.

While data is being transferred via the data stream the control stream sits idle. This can cause problems with large data transfers through firewalls which time out sessions after lengthy periods of idleness. While the file may well be successfully transferred, the control session can be disconnected by the firewall, causing an error to be generated.

The FTP protocol supports resuming of interrupted downloads using the REST commands. The client passes the number of bytes it has already received as arguments to the REST command and restarts the transfer. In some command line clients for example there is an often – ignored but valuable commands. “Reset”(meaning “get aging”) that will cause an interrupted “get ” command . hopefully to completion. After communication interruption.

Resuming uploads in not as easy. Although the FTP protocol supports the APPE COMMAND to append data to file on server. The client dose not known the exact position at which transfer got interrupted. It has to obtain the size of the file some other way, for example over a directory listing or using the SIZE command.

In ASCII mode (see below), resuming transfers can be troublesome if client and server use different end of line characters.

The objectives of FTP, as outline by its RFC are:

1. To promote sharing of files (computer programming and/or data).
2. To encourage indirect or implicit use old remote computer.
3. To shield user from variation in file storage systems among different host.
4. to transfer data reliably, and efficiently.

**FTP return codes**

FTP server return codes indicate their status by the digits within them. A brief explanation of various digits meaning are given below:

- **Ixx**: Positive Preliminary reply. The action request id being initiated but there will be another reply before it begins.

- **2xx**: positive Completion reply. The action requested has been completed. The client may now issue anew command.

- **3xx**: Positive Intermediate reply. The command was successful but further command id requested before the sever can act upon the request.

- **4xx**: Trashiest Negative completion reply. The command was not successful, but the client is free to try the command again as the failure is only temporary.

- **5xx**: Permanent Negative completion reply. The command was not successful and the client should not was due to a syntax not to attempt to repeat to it again.

- **X0x**: The failure was due to a syntax error.

- **X1x**: this response is a reply to request for information.

- **x2x**: this response is reply to connection information.

- **X3x**: this response is reply relating to accounting and authorization.

- **x4x**: Unspecified as yet.

- **X5x**: These responses indicates the status of the Server file system vis-à-vis the requested transfer or other file system action.
Implementation of FTP Client-Server Socket

CLIENT PROGRAM:

import java.net.*;
import java.io.*;

class FtpClient
{
    public static void main(String ap[])throws IOException
    {
        try
        {
            Socket s1=new Socket("localhost",21);
            BufferedReader br=new BufferedReader(new InputStreamReader(s1.getInputStream()));
            BufferedReader br1=new BufferedReader(new InputStreamReader(System.in));
            PrintWriter p=new PrintWriter(s1.getOutputStream(),true);
            System.out.println("Enter the file name to transfer:-");
            String fname=br1.readLine();
            p.println(fname);
            System.out.println("Enter the destination file name :-");
            String fname1=br1.readLine();
            p.println(fname1);
            s1.close();
        }
    }
}
catch(Exception e)
{
    System.out.println(e);
}

* OUTPUT *

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Program Files\Java\jdk1.5.0\bin>java FtpClient
Enter the file name to transfer:-
C:\Program Files\Java\jdk1.5.0\bin\aa.txt
Enter the destination file name :-
D:\b.txt

C:\Program Files\Java\jdk1.5.0\bin>java FtpClient
Enter the file name to transfer:-
C:\ne\serve.java
Enter the destination file name :-
D:\DB\serve.txt
C:\Program Files\Java\jdk1.5.0\bin>
SERVER PROGRAM:

```java
import java.net.*;
import java.io.*;

class FtpServer
{
    public static void main(String ap[])
    {
        try{
            ServerSocket s=new ServerSocket(21);
            while(true)
            {
                Socket s1=s.accept();
                FTPServerThread t=new FTPServerThread(s1);
                t.start();
            }
        }
        catch(Exception e){  }
    }
}

class FTPServerThread extends Thread
{
    Socket s;
    static String fname=new String();
```
static String fname1=new String();

FTPServerThread(Socket s)
{
    this.s=s;
    System.out.println("Socket is Created");
}

public void run()
{
    try{
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        PrintWriter pr=new PrintWriter(s.getOutputStream(),true);
        BufferedReader br1=new BufferedReader(new InputStreamReader(s.getInputStream()));
        String fname=br1.readLine();
        System.out.println("Socket is Created1");
        String fname1=br1.readLine();
        System.out.println(fname1);
        FileInputStream fin=new FileInputStream(fname);
        FileOutputStream fout=new FileOutputStream(fname1);
        System.out.println(fname);
        while(true)
        {
            int data=fin.read();
            if(data==-1)
break;
    fout.write((char)data);
    System.out.print((char)data);
} }catch(Exception e){
}

*  OUTPUT  *
C:\Program Files\Java\jdk1.5.0\bin>javac FtpServer.java
C:\Program Files\Java\jdk1.5.0\bin>javac Ftpclient.java
C:\Program Files\Java\jdk1.5.0\bin>start
C:\Program Files\Java\jdk1.5.0\bin>java FtpServer
Socket is Created
Socket is Created1
D:\b.txt
Socket is Created
Socket is Created1
D:\DB\serve.txt
C:\ne\serve.java
import java.io.*;
import java.awt.*;
import java.swing.*;
import java.net.*;
import java.util.*;
import java.awt.event.*;

class chats
{
    string msg;
    string msg1;
    chats()
    {
        try
        {
            ServerSocket ss=new ServerSocket(2222);
            Socket s=ss.accept();
            BufferedReader br=new BufferedReader(new InputStreamReader(s.getInputStream()));
            BufferedReader in=new BufferedReader(new InputStreamReader(system.in));
            PrintWriter pw=new PrintWriter(s.getOutputStream(),true);
            msg=br.readline();
            while(true)
            {
                if(msg.equals("bye"))
                {
                    System.exit(0);
                }
                system.out.println("message from client:"+msg);
CONCLUSION: Thus We Studied FTP server data capture illustration.
EXPERIMENT NO. - 6

**TITLE** :- Implementation of Remote Method Invocation (RMI).

**AIM** :- To implement Remote Method Invocation (RMI) using java.

**OBJECTIVE** :- To understand network programming using socket programming

**THEORY** :-

The Java Remote Method Invocation (RMI) application program interface enables client & server communications over the net. Typically, client programs send requests to a server program & the server programs respond to those requests.

A common eg. is sharing a word processing program over a network. The word processor is installed on a server & anyone who wants to use it starts it from his or her machine by double clicking an icon on the desktop or trying at the command line.

The invocation sends a request to a server program for access to the software & the server program responds by making the software available to the requestor.

![Diagram of client-server interaction](image)

The RMI API lets you create a publicly accessible remote server object that enables client & server communications through simple method calls on the server objects. Clients can easily communicate directly with the server object & indirectly with each other through the server object using Uniform Resource Locator (URLs) & HTTP.

Serialization is also needed to implement Remote Method Invocation (RMI). RMI allows a Java object on one machine to invoke a method of a Java object on a different machine. An object may be supplied as an argument to that remote method.
RMI allows a java object that executes on one machine to invoke a method of a Java object that executes on another machine. This is an important feature, because it allows you to build distributed applications. While a complete discussion of RMI is outside the scope of this book, the following example describes the basic principles involved.

A simple Client/Server Application Using RMI :-

This section provides step-by-step directions for building a simple client/server application by using RMI. The server receives the request from the client, processes it, & returns a result. In this eg the request specifies two numbers. The server adds these together & returns the sum.

1. Enter & Compile the source code :-

This application uses four source files. The first file, AddServerIntf.java defines the remote interface that is provided by the server. It contains one method that accepts two double arguments & returns their sum. All remote interfaces must extend the Remote interface, which is part of java.rmi. Remote defines no members. Its purpose is simply to indicate that an interface uses remote methods. All remote methods can throw a RemoteException.

The second source file, AddServerImpl.java, implements the remote interface. The implementation of the add() method is straightforward. All remote objects must extend UnicastRemoteObject, which provides functionality that is needed to make objects available from remote machines.

The third source file, AddServer.java, contains the main program for the server machine. Its primary function is to update the RMI registry on that machine. This is done by using the rebind() method of the Naming class. That method associates a name with an object reference. The first argument to the rebind() method is a string that names the server as "Addserver". Its second argument is a reference to an instance of AddServerImpl.

The fourth source file, AddClient.java, implements the clients side of this distributed application. AddClient.java requires three command line arguments. The first is the IP address or name of the server machine. The second & third arguments are the two numbers that are to be summed.
The application begins by forming a string that follows the URL syntax. This URL uses the rmi protocol. The string includes the IP address or name of the server and the string “AddServer”. The program then invokes the lookup() method of the Naming class. This method accepts one argument, the rmi URL, & returns a reference to an object of type AdServerIntf. All remote method invocations can then be directed to this object.

The program continues by displaying its argument & then invokes the remote add() method. The sum is returned from this method & is then printed.

Generate Stubs & Skeletons :-

To generate stubs & skeletons, you use a tool called the RMI compiler, which is invoked from the command line, as shown here:

```
RmicAddServerImpl
```

This command generates two new files: AddServerImpl_Skel.class (skeleton) & AddServerImpl_Stub.class (stub). When using rmic, be sure that CLASSPATH is set to include the current directory. As you can see, by default, rmic generates both a stub & a skeleton file. If you do not need the skeleton, You have the option to suppress it.

1.Install Files on the Client & Server Machines :-

Copy AddClient.class, AddServerImpl_Stub.class & AddServerIntf.class to a directory on the client machine. Copy AddServerIntf.class, AddServerImpl.class, AddServerImpl_Skel.class, AddServer.class to a directory on the server machine.

2.Start the RMI registry on the Server Machine :-

The Java 2 SDK provides a program called rmiregistry, which executes on the server machine. It maps names to object references. First, check that the CLASSPATH environment variable includes the directory in which your files are located. Then, start the RMI Registry from the command line, as shown here:

```
Start rmiregistry
```

When this command returns, you should see that a new window has been created. You need to leave this window open until you are done experimenting with the RMI eg.
3. **Start the Server :-**
   The Server code is started from the command line as shown here:
   
   ```java
   Java AddServer
   ```
   
   Recall that the AddServer code instantiates AddServerImpl & registers that object with the name “Add Server”.

4. **Start the Client :-**
   The AddClient software requires three arguments: the name or IP address of the server machine & the two numbers that are to be summed together. You may invoke it from the command line by using one of the two formats shown here:
   
   ```java
   Java AddClient server1 8 9
   Java AddClient 11.12.13.14 8 9
   ```

**Implementation of RMI**

**RMIServer Program :**

```java
import java.rmi.*;
import java.rmi.server.*;
import java.io.*;
interface RMIMethods extends Remote
{
    double circleArea(double r) throws RemoteException;
    double rectangleArea(double a,double b) throws RemoteException;
    double cubeArea(double s) throws RemoteException;
    double ellipseArea(double a,double b) throws RemoteException;
}
```
public class RMIServer extends UnicastRemoteObject implements RMIMethods {

    public RMIServer() throws RemoteException {
    }

    public double circleArea(double rad) throws RemoteException {
        return 3.14*rad*rad;
    }

    public double rectangleArea(double side1, double side2) throws RemoteException {
        return side1*side2;
    }

    public double cubeArea(double side) throws RemoteException {
        return side*side*side;
    }

    public double ellipseArea(double majoraxis, double minoraxis) throws RemoteException {
        return 3.14*majoraxis*minoraxis;
    }

    public static void main(String args[]) throws RemoteException, IOException {
        try {
            try {
                
            }
        }
    }
RMIServer server=new RMIServer();
Naming.rebind("area",server);
System.out.println("Methods are registred to RMI REGISTRY.......");
}
catch(Exception e)
{
    System.out.println("Exception in main() function: "+e);
}
}

RMIClient Program :
import java.rmi.*;
public class AddClient
{
    public static void main(String args[])
    {
        try
        {
            RMIMethods rmimethods=(RMIMethods)
            Naming.lookup("rmi://localhost/area"); //area is the name bounded by the server
            double a1=rmimethods.circleArea(5.0);
            System.out.println("\nRadius of circle is : 5.0");
            System.out.println("Area of Circle is : "+a1);
            System.out.println("\nSide of Cube is : 5.0");
            double a3=rmimethods.cubeArea(5.0);
System.out.println("Area of Cube is : "+a3);
System.out.println("\nSide of Rectangle are : 5.0 and 5.0");
double a2=rmiMethods.rectangleArea(5.0,5.0);
System.out.println("Area of Rectangle is : "+a2);
System.out.println("\nMajor and Minor axis of Ellipse are : 5.0 and 5.0");
double a4=rmiMethods.ellipseArea(5.0,5.0);
System.out.println("Area of Ellipse is : "+a4);
}
catch(Exception e)
{
    System.out.println("Exception in main(): "+e);
}
}

MATHServer Program :
import java.rmi.server.*;
import java.rmi.*;
interface mathFuns extends Remote
{
    double sin(double a) throws RemoteException;
    double cos(double a) throws RemoteException;
}
class MathServer extends UnicastRemoteObject implements mathFuns
{
    public MathServer() throws RemoteException
public double sin(double angle) throws RemoteException
{
    return Math.sin(3.14*angle/180);
}

public double cos(double angle) throws RemoteException
{
    return Math.cos(3.14*angle/180);
}

public static void main(String a[]) throws RemoteException
{
    try
    {
        MathServer ms = new MathServer();
        Naming.rebind("maths",ms);
        System.out.println("Methods are registred to RMI REGISTRY.......");
    }
    catch(Exception e)
    {
        System.out.println("Exception in main() : "+e);
    }
}

MATHClient Program:

import java.rmi.*;

class MathClient
{
    public static void main(String a[])
    {
        try
        {
            mathFuns maths = (mathFuns) Naming.lookup("rmi://localhost/maths");
            double angle = 90;
            System.out.println("\n\nSin(\n+angle\n) = \n\n+maths.sin(angle));
            System.out.println("\nCos(\n+angle\n) = \n\n+maths.cos(angle));
        }
        catch(Exception e)
        {
            System.out.println("Exception at Client Side main(): \n\n+e);}
        }
    }
}
RMI OUTPUT:

Server:

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\user>cd 
C:\>cd C:\jdk1.6.0_17\bin
C:\jdk1.6.0_17\bin>set path=C:\Program Files\Java\jdk1.6.0_17\bin
C:\jdk1.6.0_17\bin>javac RMIServer.java
C:\jdk1.6.0_17\bin>javac MathServer.java
C:\jdk1.6.0_17\bin>start rmiregistry
C:\jdk1.6.0_17\bin>java RMIServer
Methods are registered to RMI REGISTRY....... 
C:\jdk1.6.0_17\bin>java MathServer
Methods are registered to RMI REGISTRY....... 

Client:

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\user>cd 
C:\>cd C:\jdk1.6.0_17\bin
C:\jdk1.6.0_17\bin>set path=C:\Program Files\Java\jdk1.6.0_17\bin
C:\jdk1.6.0_17\bin>javac Addclient.java
C:\jdk1.6.0_17\bin>javac MathClient.java
C:\jdk1.6.0_17\bin>java AddClient
Radius of circle is : 5.0
Area of Circle is : 78.5
Side of Cube is : 5.0
Area of Cube is : 125.0
Side of Rectangle are : 5.0 and 5.0
Area of Rectangle is : 25.0
Major and Minor axis of Ellipse are : 5.0 and 5.0
Area of Ellipse is : 78.5

C:\jdk1.6.0_17\bin>java MathClient
Sin(90.0) = 0.9999996829318346
Cos(90.0) = 7.963267107332633E-4

CONCLUSION : Hence the remote method is invoked of server side by client.
EXPERIMENT NO. - 7

**TITLE:** - Implementation of TCP/IP.

**AIM:** - To implement the TCP/IP protocol.

**OBJECTIVE:** - To understand network programming using socket programming.

**THEORY:** -

TCP is a transport layer protocol used by applications that require guaranteed delivery. It is a sliding window protocol that provides handling for both timeouts and retransmissions.

TCP establishes a full duplex virtual connection between two endpoints. Each endpoint is defined by an IP address and a TCP port number. The operation of TCP is implemented as a finite state machine.

The byte stream is transferred in segments. The window size determines the number of bytes of data that can be sent before an acknowledgement from the receiver is necessary.

![TCP header diagram](image)

**TCP header:**

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence Number</td>
<td></td>
</tr>
<tr>
<td>Acknowledgment Number</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Offset</th>
<th>reserved</th>
<th>ECN</th>
<th>Control Bits</th>
<th>Window</th>
</tr>
</thead>
</table>
Source Port. 16 bits.

Destination Port. 16 bits.

Sequence Number. 32 bits.
The sequence number of the first data byte in this segment. If the SYN bit is set, the sequence number is the initial sequence number and the first data byte is initial sequence number + 1.

Acknowledgment Number. 32 bits.
If the ACK bit is set, this field contains the value of the next sequence number the sender of the segment is expecting to receive. Once a connection is established this is always sent.

Data Offset. 4 bits.
The number of 32-bit words in the TCP header. This indicates where the data begins. The length of the TCP header is always a multiple of 32 bits.

Reserved. 3 bits.
Must be cleared to zero.

ECN, Explicit Congestion Notification. 3 bits.
Added in RFC 3168.

<table>
<thead>
<tr>
<th>00</th>
<th>01</th>
<th>02</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>C</td>
<td>E</td>
</tr>
</tbody>
</table>

N, NS, Nonce Sum. 1 bit.
Added in RFC 3540. This is an optional field added to ECN intended to protect against accidental or malicious concealment of marked packets from the TCP sender.
C, CWR. 1 bit.

E, ECE, ECN-Echo. 1 bit.

**Control Bits.** 6 bits.

<table>
<thead>
<tr>
<th>00</th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>U</strong></td>
<td><strong>A</strong></td>
<td><strong>P</strong></td>
<td><strong>R</strong></td>
<td><strong>S</strong></td>
<td><strong>F</strong></td>
</tr>
</tbody>
</table>

U, URG. 1 bit.
Urgent pointer valid flag.

A, ACK. 1 bit.
Acknowledgment number valid flag.

P, PSH. 1 bit.
Push flag.

R, RST. 1 bit.
Reset connection flag.

S, SYN. 1 bit.
Synchronize sequence numbers flag.

F, FIN. 1 bit.
End of data flag.

Window. 16 bits, unsigned.
The number of data bytes beginning with the one indicated in the acknowledgment field which the sender of this segment is willing to accept.

Checksum. 16 bits.
This is computed as the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the TCP header, and the data, padded as needed with zero bytes at the end to make a multiple of two bytes. The pseudo header contains the following fields:
### Source IP address

### Destination IP address

<table>
<thead>
<tr>
<th>IP Protocol</th>
<th>Total length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Urgent Pointer.** 16 bits, unsigned.  
If the URG bit is set, this field points to the sequence number of the last byte in a sequence of urgent data.

**Options.** 0 to 40 bytes.  
Options occupy space at the end of the TCP header. All options are included in the checksum. An option may begin on any byte boundary. The TCP header must be padded with zeros to make the header length a multiple of 32 bits.

**TCP State machine:**

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOSE-WAIT</td>
<td>Waits for a connection termination request from the remote host.</td>
</tr>
<tr>
<td>CLOSED</td>
<td>Represents no connection state at all.</td>
</tr>
<tr>
<td>CLOSING</td>
<td>Waits for a connection termination request acknowledgment from the remote host.</td>
</tr>
<tr>
<td>ESTABLISHED</td>
<td>Represents an open connection, data received can be delivered to the user. The normal state for the data transfer phase of the connection.</td>
</tr>
<tr>
<td>FIN-WAIT-1</td>
<td>Waits for a connection termination request from the remote host or an acknowledgment of the connection termination request previously sent.</td>
</tr>
<tr>
<td>FIN-WAIT-2</td>
<td>Waits for a connection termination request from the remote host.</td>
</tr>
</tbody>
</table>
The CLOSED state is the entry point to the TCP state machine.

**Implementation of TCP**

**CLIENT PROGRAM**

```java
import java.io.*;
import java.net.*;
class tcpclient
{
    public static void main(String ar[])
    throws Exception
    {
        Socket s = new Socket("localhost", 4546);
    }
```
BufferedReader br=new BufferedReader(new InputStreamReader(s.getInputStream()));
PrintWriter pr=new PrintWriter(s.getOutputStream(),true);
BufferedReader br1=new BufferedReader(new InputStreamReader(System.in));
PrintWriter pr1=new PrintWriter(System.out,true);
boolean a=true;

while(a)
{
    try
    {
        pr1.println("Enter msg for server=");
        String s1=br1.readLine();
        pr.println(s1);
        s1=br.readLine();
        System.out.println(""+s1);
    }
    catch(Exception e)
    {
    }
}
br.close();
br1.close();
pr.close();
pr1.close();
import java.net.*;
import java.io.*;
import java.util.*;

public class tcpserver extends Thread
{
    public static void main(String a[])throws Exception
    {
        Socket cli;
        int counter=0;
        ServerSocket serv=new ServerSocket(4546);
        System.out.println("nWaiting for new connection... ");

        while(true)
        {
            cli=serv.accept();
            System.out.println("n\nNew Client No : "+(++counter));
            Thread th=new ThreadServer(cli,counter);
            th.start();
        }
    }
}

class ThreadServer extends Thread
{
    Socket sock;
    int ID,clictr;
    public ThreadServer(Socket s,int no)
{
    sock=s;
    ID=no;
}

public void run()
{
    try{
        BufferedReader br=new BufferedReader(new InputStreamReader(sock.getInputStream()));
        PrintWriter pr=new PrintWriter(sock.getOutputStream(),true);
        BufferedReader br1=new BufferedReader(new InputStreamReader(System.in));
        PrintWriter pr1=new PrintWriter(System.out,true);

        boolean a=true;

        while(a)
        {
            String str=br.readLine();

            pr1.println("msg from client="+str);
            pr1.println("msg for client=");
            str=br1.readLine();
            pr.println(str);
        }
    }
    catch(Exception e)
    {
    }
}

CONCLUSION : - Thus we have implemented TCP/IP Protocol.
EXPERIMENT NO. - 8

TITLE :- Implementation of UDP.

AIM :- To implement UDP.

OBJECTIVE :- To understand network programming using socket programming.

THEORY :-

Both clients & servers can run in concurrent mode :

- **Concurrent in Clients :-**
  Clients can run on a machine either iteratively or concurrently. Running clients iteratively means running them one by one, one client must start, run & terminate before the m/c can start another client. Most computers today, however, allow concurrent clients, i.e two or more clients can run at the same time.

- **Concurrency in servers :-**
  An iterative server can process only one request at a time, it receive a request, processes it, & sends the response to the requestor before it handles another request. A concurrent server, on the other hand, can process many requests at the same time & thus can share its time between many respects. The servers use either UDP, a connectionless transport layer protocol or TCP, a connection-oriented transport layer protocol & the service method.

Theoretically, We can have four types of servers : Connectionless iterative, Connectionless Concurrent, Connection-oriented iterative & connection-oriented concurrent.

![Types of Server Diagram]

Fig-Types of Server
● **Connectionless Iterative Server** :-
The servers that use UDP are normally iterative, which as we have said, means that
the server processes one request at a time. A server gets the request in a datagram
from UDP, processes the request, & gives the response to UDP to send to the client.

The server plays no attention to the other datagram’s. Theses datagram’s are stored
in a queue, waiting for service. They could all be from many clients. In either case
they are processed one by one in order of arrival.

The server uses one single port for this purpose, the well-known port. All the
datagram’s arriving at this port wait in line to be served.

**Implementation of UDP**

**UDPServer Program** :

```java
import java.net.*;
import java.io.*;
class UdpServer {
    public static void main(String a[])
    {
        try
        {
            DatagramSocket d=new DatagramSocket(2345);
            InetAddress i=InetAddress.getByName("localhost");
            BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
            while(true)
            {
                System.out.println("Msg For Client");
                String msg=br.readLine();
                byte b[]=msg.getBytes();
            }
        }
    }
}
```

DatagramPacket p=new DatagramPacket(b,b.length,i,3456);
d.send(p);
byte b1[]=new byte[100];
DatagramPacket p1=new DatagramPacket(b1,b1.length);
d.receive(p1);
System.out.println("Msg From Client"+new String(p1.getData()));
}
}
catch(Exception e )
{
}
}

UDPClient Program :
import java.net.*;
import java.io.*;
class UdpClient
{
    public static void main(String a[])
    {
        try
        {
            DatagramSocket d=new DatagramSocket(3456);
            InetAddress i=InetAddress.getByName("localhost");
            BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
            while(true)
            {
                ...
byte b1[]=new byte[100];
DatagramPacket p1=new DatagramPacket(b1,b1.length);
d.receive(p1);
System.out.println("Msg From Server"+new String(p1.getData()));
System.out.println("Msg For Server");
String msg=br.readLine();
byte b[]=msg.getBytes();
DatagramPacket p=new DatagramPacket(b,b.length,i,2345);
d.send(p);
}
}
catch(Exception e )
{
}
}
UDP OUTPUT:

server:
C:\Program Files\Java\jdk1.5.0_01\bin>javac UdpServer.java
C:\Program Files\Java\jdk1.5.0_01\bin>java UdpServer
Msg For Client
Hello Client! How are you?
Msg From Client Hello Server! I am fine!!
Msg For Client
Good Morning Client
Msg From Client Good Morning & Have a Nice Day!!

client:
C:\Program Files\Java\jdk1.5.0_01\bin>javac UdpClient.java
C:\Program Files\Java\jdk1.5.0_01\bin>java UdpClient
Msg From Server Hello Client! How are you?
Msg For Server Hello Server! I am fine!!
Msg From Server Good Morning Client
Msg For Server Good Morning & Have a Nice Day!!

CONCLUSION: Thus we have implemented UDP Protocol.
EXPERIMENT NO. - 9

**TITLE:** - Study of windows 2003/2008 system architecture.

**AIM:** - To learn about system architecture of windows 2003/2008

**OBJECTIVE:** - Understanding Windows 2003/2008 simplified & detailed architecture.

**THEORY:**

**Requirements and Design Goals**-

1. Provide a true 32-bit, preemptive, reentrant, virtual memory operating system
2. Run on multiple hardware architectures and platforms
3. Run and scale well on symmetric multiprocessing systems
4. Be a great distributed computing platform, both as a network client and as a server
5. Run most existing 16-bit MS-DOS and Microsoft Windows 3.1 applications
6. Meet government requirements for POSIX 1003.1 compliance
7. Meet government and industry requirements for operating system security
8. Be easily adaptable to the global market by supporting Unicode

To guide the thousands of decisions that had to be made to create a system that met these requirements, the Windows NT design team adopted the following design goals at the beginning of the project:

9. Extensibility The code must be written to comfortably grow and change as market Requirements change.
10. Portability The system must be able to run on multiple hardware architectures and must be able to move with relative ease to new ones as market demands dictate.
11. Reliability and robustness The system should protect itself from both internal malfunction and external tampering. Applications should not be able to harm the operating system or other applications.
12. Compatibility Although Windows NT should extend existing technology, its user interface and APIs should be compatible with older versions of Windows and with MS-DOS. It should also Interoperate well with other systems, such as UNIX, OS/2, and NetWare.
13. Performance Within the constraints of the other design goals, the system should be as fast and responsive as possible on each hardware platform.
Architecture Overview

A simplified version of this architecture is shown in Figure.

In Figure first notice the line dividing the user-mode and kernel-mode parts of the Windows operating system. The boxes above the line represent user-mode processes, and the components below the line are kernel-mode operating system services. User-mode threads execute in a protected process address space (although while they are executing in kernel mode, they have access to system space). Thus, system support processes, service processes, user applications, and environment subsystems each have their own private process address space.

The four basic types of user-mode processes are described as follows:

■ Fixed (or hardwired) system support processes, such as the logon process and the Session Manager, that are not Windows services. (That is, they are not started by the service control manager

■ Service processes that host Windows services, such as the Task Scheduler and Print Spooler services. Services generally have the requirement that they run independently of user logons. Many Windows server applications, such as Microsoft SQL Server and Microsoft Exchange Server, also include components that run as services.
User applications, which can be one of five types: Windows 32-bit, Windows 64-bit, Windows 3.1 16-bit, MS-DOS 16-bit, or POSIX 32-bit.

Environment subsystem server processes, which implement part of the support for the operating system environment, or personality presented to the user and programmer.

Windows NT originally shipped with three environment subsystems: Windows, POSIX, and OS/2. However, OS/2 last shipped with Windows 2000. Windows Vista Ultimate and Enterprise include support for an enhanced POSIX subsystem called Subsystem for Unix-based Applications (SUA).

"Subsystem DLLs" box below the “Service processes" and “User applications" boxes. Under Windows, user applications don't call the native Windows operating system services directly; rather, they go through one or more subsystem dynamic link libraries (DLLs). The role of the subsystem DLLs is to translate a documented function into the appropriate internal (and generally undocumented) native system service calls. This translation might or might not involve sending a message to the environment subsystem process that is serving the user application.

**Key System Components**
Now that we've looked at the high-level architecture of Windows, let's delve deeper into the internal structure and the role each key operating system component plays.
CONCLUSION: - Thus we have studied windows 2003/2008 System Architecture.
EXPERIMENT NO. - 10

TITLE  :- Installation of windows 2003/2008 server.

AIM  :- To install windows 2003/2008 server


THEORY  :-

Steps to install windows 2003 server:

1. Insert bootable CD of Windows 2003/2008 server in CDROM
2. After setup scanning file system, select the option "install Windows 2003/2008 server". Select the drive to which OS installs. If you have planned to install dual boot then, select another drive e.g. d: else select c: and install fresh copy of OS.
3. Next file system menu will be displayed to ask format, convert and leave file system. Select appropriate option and press “Enter”.
4. Next menu is License agreement and CD-Key input dialog box.
5. After that setup will ask you for regional settings, computer name and administrators password. Here it finalization of setup starts and setup ends.

Windows Server 2003 operating systems take the best of Windows 2000 Server technology and make it easier to deploy, manage, and use. The result: a highly productive infrastructure that helps make your network a strategic asset for your organization. Windows Server 2003 SP2 provides enhanced security, increased reliability, and a simplified administration to help enterprise customers across all industries.

Basic Windows 2003 Server Installation

1. Disconnect Server from Network.
2. Format/Install OS
3. It is recommended that there be at least two partitions; a system partition (8GB or better preferred) and a data partition.
4. Install without IIS if you are not going to run a website.
5. Join Server to a workgroup (any name is fine).
6. Make sure both Administrator and Guest have passwords set (guest disabled).
   Eight characters minimum with at least one number and one special character preferred.
7. Install any Security updates from CD if available (RPC patches such as Blaster and Sasser especially).
8. Install passprop.exe from the NT Resource Kit. This will allow the Administrator Account to be locked out over network connections only. Interactive logins of the Administrator account will not be locked out.
9. At the C:\ command prompt, type `passprop /adminlockout`.
10. Enter IP addresses of server, DNS (129.118.1.3 & 129.118.1.9), and WINS (129.118.1.152 & 129.118.1.151).
11. Uncheck File and Printer Sharing if you are not going to share files or printers.
12. Set auditing to track success/failure of Account Management, Logon, Account Logon, and Policy Change events at a minimum.
13. Install Norton Antivirus from CD if available.
14. If the CD is not available, connect to the Network and immediately install Norton anti-virus from http://www.eraider.ttu.edu website.
15. Run Live Update.
16. Apply the latest Critical Updates from Windows Update site (http://windowsupdate.microsoft.com).
17. Disable any services not needed for the particular installation.

**Computer and processor**

PC with a 133-MHz processor required; 550-MHz or faster processor recommended; support for up to four processors on one server

**Memory**

128 MB of RAM required; 256 MB or more recommended; 4 GB maximum

**Hard disk**

1.2 GB for network install; 2.9 GB for CD install

**Drive**

CD-ROM or DVD-ROM drive
Display

VGA or hardware that supports console redirection required; Super VGA supporting 800 x 600 or higher-resolution monitor recommended

Plan your installation

When you run the Windows Server 2003 Setup program, you must provide information about how to install and configure the operating system. Thorough planning can make your installation of Windows Server 2003 more efficient by helping you to avoid potential problems during installation.

when planning for your Windows Server 2003 installation

- Check System Requirements
- Check Hardware and Software Compatibility
- Determine Disk Partitioning Options
- Choose the Appropriate File System: FAT, FAT32, NTFS
- Decide on a Workgroup or Domain Installation
- Complete a Pre-Installation Checklist

After you made sure you can go on, start the installation process.

Windows 2008 Server-

To use Windows Server 2008 you need to meet the following hardware requirements:

Component Requirement

Processor
- Minimum: 1GHz (x86 processor) or 1.4GHz (x64 processor)
- Recommended: 2GHz or faster Note: An Intel Itanium 2 processor is required for Windows Server 2008 for Itanium-based Systems
Memory
  • Minimum: 512MB RAM • Recommended: 2GB RAM or greater
  • Maximum (32-bit systems): 4GB (Standard) or 64GB (Enterprise and Datacenter)
  • Maximum (64-bit systems): 32GB (Standard) or 2TB (Enterprise, Datacenter and Itanium-based Systems)

Available Disk Space
  • Minimum: 10GB
  • Recommended: 40GB or greater Note: Computers with more than 16GB of RAM will require more disk space for paging, hibernation, and dump files

Drive DVD-ROM drive

Display and Peripherals
  • Super VGA (800 x 600) or higher-resolution monitor • Keyboard • Microsoft Mouse or compatible pointing device

Follow this procedure to install Windows Server 2008:

1. Insert the appropriate Windows Server 2008 installation media into your DVD drive.

2. Reboot the computer.

3. When prompted for an installation language and other regional options, make your selection and press Next.

4. Next, press Install Now to begin the installation process.

5. Product activation is now also identical with that found in Windows Vista. Enter your Product ID in the next window, and if you want to automatically activate Windows the moment the installation finishes, click Next.
If you do not have the Product ID available right now, you can leave the box empty, and click Next. You will need to provide the Product ID later, after the server installation is over. Press No.

6. Because you did not provide the correct ID, the installation process cannot determine what kind of Windows Server 2008 license you own, and therefore you will be prompted to select your correct version in the next screen, assuming you are telling the truth and will provide the correct ID to prove your selection later on.

7. If you did provide the right Product ID, select the Full version of the right Windows version you're prompted, and click Next.

8. Read and accept the license terms by clicking to select the checkbox and pressing Next.

9. In the "Which type of installation do you want?" window, click the only available option – Custom (Advanced).

10. In the "Where do you want to install Windows?", if you're installing the server on a regular IDE hard disk, click to select the first disk, usually Disk 0, and click Next.

If you're installing on a hard disk that's connected to a SCSI controller, click Load Driver and insert the media provided by the controller's manufacturer.

If you're installing in a Virtual Machine environment, make sure you read the "Installing the Virtual SCSI Controller Driver for Virtual Server 2005 on Windows Server 2008"

If you must, you can also click Drive Options and manually create a partition on the destination hard disk.

11. The installation now begins, and you can go and have lunch. Copying the setup files from the DVD to the hard drive only takes about one minute. However, extracting and uncompressing the files takes a good deal longer. After 20 minutes, the operating system is installed. The exact time it takes to install server core depends upon your hardware specifications. Faster disks will perform much faster installs… Windows Server 2008 takes up approximately 10 GB of hard drive space.

The installation process will reboot your computer, so, if in step #10 you inserted a floppy disk (either real or virtual), make sure you remove it before going to lunch, as you'll find the server hanged without the ability to boot (you can bypass this by
configuring the server to boot from a CD/DVD and then from the hard disk in the booting order on the server's BIOS)

12. Then the server reboots you'll be prompted with the new Windows Server 2008 type of login screen. Press CTRL+ALT+DEL to log in.

13. Click on Other User.

14. The default Administrator is blank, so just type Administrator and press Enter.

15. You will be prompted to change the user's password. You have no choice but to press Ok.

16. In the password changing dialog box, leave the default password blank (duh, read step #15...), and enter a new, complex, at-least-7-characters-long new password twice. A password like "top-secret" is not valid (it's not complex), but one like "T0pSecreT!" sure is. Make sure you remember it.

17. Someone thought it would be cool to nag you once more, so now you'll be prompted to accept the fact that the password had been changed. Press Ok.

18. finally, the desktop appears and that's it, you're logged on and can begin working. You will be greeted by an assistant for the initial server configuration, and after performing some initial configuration tasks, you will be able to start working.

**CONCLUSION:** - Installation of Windows 2003/2008 Server is completed with simple steps.
EXPERIMENT NO. - 11


AIM  :-To implement Active Directory on Windows 2003 Server.


THEORY:-

Active Directory stores information about objects on the network and makes this information easy for administrators and users to find and use. Active Directory directory service uses a structured data store as the basis for a logical, hierarchical organization of directory information.

Security is integrated with Active Directory through logon authentication and access control to objects in the directory. With a single network logon, administrators can manage directory data and organization throughout their network, and authorized network users can access resources anywhere on the network. Policy-based administration eases the management of even the most complex network.

The Active Directory service has the following features:

- A data store, also known as the directory, which stores information about Active Directory objects. These objects typically include shared resources such as servers, files, printers, and the network user and computer accounts. For more information about the Active Directory data store, see Directory data store.
- A set of rules, the schema, that defines the classes of objects and attributes contained in the directory, the constraints and limits on instances of these objects, and the format of their names. For more information about the schema, see Active Directory schema overview.
- A global catalog that contains information about every object in the directory. This allows users and administrators to find directory information regardless of which domain in the directory actually contains the data. For more information about the global catalog, see Global catalog.
- A query and index mechanism, so that objects and their properties can be published and found by network users or applications. For more information about querying the directory, see finding directory information.
• A replication service that distributes directory data across a network. All domain controllers in a domain participate in replication and contain a complete copy of all directory information for their domain. Any change to directory data is replicated to all domain controllers in the domain. For more information about Active Directory replication, see Replication goals and strategies.

• Integration with the security subsystem for a secure logon process to a network, as well as access control on both directory data queries and data modifications. For more information about Active Directory security, see Security model.

• To gain the full benefits of Active Directory, the computer accessing the Active Directory over the network must be running the correct client software. To computers not running Active Directory client software, the directory will appear just like a Windows NT directory. For more information about client software.

The Active Directory administrative tools that are included with Windows Server simplify directory service administration. You can use the standard tools or, using Microsoft Management Console (MMC), create custom tools that focus on single management tasks. You can combine several tools into one console. You can also assign custom tools to individual administrators with specific administrative responsibilities. For information about MMC, see Creating and opening MMC consoles. The Active Directory administrative tools can only be used from a computer with access to a Windows domain. The following Active Directory administrative tools are available on the Windows Server Administrative Tools menu of all Windows domain controllers:

• Active Directory Users and Computers
• Active Directory Domains and Trusts
• Active Directory Sites and Services
Practical Work:

Following are the steps used for installation of active directory:

1. Open “Manage your server” from control panel.

2. Click on Add or Remove roles.
3. Select the option “Active directory(Domain Controller)”.
4. Click “Next” to active directory installation wizard.
5. In directory installation wizard first it will ask type of domain controller. Select “Domain Controller for a New Domain.” Click on Next.
6. Select “A New Domain”. Click on “Next”.
7. Enter “DNS full name for domain” in New Domain Name Window. e.g. cse.rit or domain1.com. Click on “Next”.
8. Enter NetBIOS name for Domain. Click on “Next”.
9. Next few dialogs are related with files to be used by Active Directory. Click on “Next”.
10. Finally, it will prompts you to enter AD restore mode password. Enter password and Click on “Next”.

![Manage Your Server](image-url)
Managing Active Directory:

You can manage your AD using “Manage your server wizard”.

Or you will find same functionalities in “Administrative Tools”.

These are:

1. Manage users and computers in Active Directory.
2. Manage domains and trusts.
3. Manage sites and services.

CONCLUSION :-Thus we implemented Windows 2003/2008 configuration of Active Directory.
EXPERIMENT NO. - 12

**TITLE:** - Windows 2003/2008- Study and configuration of DHCP service with static as well as dynamic address binding.

**AIM:** - To study & configure DHCP service with static as well as dynamic address binding in Windows 2003/2008 server

**OBJECTIVE:** - Understanding of automatic IP address allocation.

**THEORY:-**

DHCP is a TCP/IP standard that reduces the complexity and administrative overhead of managing network client IPv4 addresses and other configuration parameters. A properly configured DHCP infrastructure eliminates the configuration problems associated with manually configuring TCP/IP. A DHCP infrastructure consists of the following elements:

**DHCP servers**

Computers that offer dynamic configuration of IPv4 addresses and related configuration parameters to DHCP clients.

**DHCP clients**

Network nodes that support the ability to communicate with a DHCP server to obtain a dynamically leased IPv4 address and related configuration parameters.

**DHCP relay agents**

Network nodes, typically routers, that listen for broadcast and unicast DHCP messages and relay them between DHCP servers and DHCP clients. Without DHCP relay agents, you would have to install a DHCP server on each subnet that contains DHCP clients.
Each time a DHCP client starts, it requests IPv4 addressing information from a DHCP server, including:

- IPv4 address
- Subnet mask
- Additional configuration parameters, such as a default gateway address, Domain Name System (DNS) server addresses, a DNS domain name, and Windows Internet Name Service (WINS) server addresses. When a DHCP server receives a request, it selects an available IPv4 address from a pool of addresses defined in its database (along with other configuration parameters) and offers it to the DHCP client. If the client accepts the offer, the IPv4 addressing information is leased to the client for a specified period of time. The DHCP client will typically continue to attempt to contact a DHCP server if a response to its request for an IPv4 address configuration is not received, either because the DHCP server cannot be reached or because no more IPv4 addresses are available in the pool to lease to the client. For DHCP clients that are based on Microsoft Windows XP or Windows Server 2003 operating systems, the DHCP Client service uses the alternate configuration when it cannot contact a DHCP server. The alternate configuration can be either an Automatic Private IP Addressing [APIPA] address or an alternate configuration that has been configured manually.

Requests for Comments (RFCs) 2131 and 2132 define the operation of DHCP clients and servers. RFC 1542 defines the operation of DHCP relay agents. All DHCP messages are sent using the User Datagram Protocol (UDP). DHCP clients listen on UDP port 67. DHCP servers listen on UDP port 68. DHCP relay agents listen on both UDP ports.
Practical Work:

Before you install a Windows-based DHCP server, ask yourself these questions:

Q.1. What IPv4 configuration options will DHCP clients obtain from a DHCP server (such as default gateway, DNS servers, a DNS domain name, or WINS servers)?

The IPv4 configuration options determine how you should configure the DHCP server and whether the options should be created for all clients in the entire network, clients on a specific subnet, or individual clients.

Q.2. Will all computers become DHCP clients? If not, consider that non-DHCP clients have static IPv4 addresses, and you might have to exclude those addresses from the scopes that you create on DHCP servers. If a specific DHCP client requires a specific IPv4 address, you must reserve the address.

Q.3. Will a DHCP server supply IPv4 addresses to multiple subnets?

If so, each subnet must contain a DHCP relay agent. If a subnet does not have a DHCP relay agent, you must install a separate DHCP server on the subnet.

Q.4. How many DHCP servers do you require?

To ensure fault tolerance for DHCP configuration, you should use at least two DHCP servers. You might need additional DHCP servers for branch offices of a large organization.

- Steps for configuring DHCP Server-:
  
  On server side-
  1. Go to server manager – Click on Add Roll – Click on DHCP server option.
  2. On Add rolls wizards click on Dhcp scop – Click on Add button.
  3. Enter the fields- 1. Scope name-
     1. Starting IP address.
     2. Ending IP address.
     3. Subnet mask.
     4. Click ok.
  4. Disable DHCPv6 stateless mode for this server.
5. Click next.

On client side

1. Go to control panel
2. Go to network connection
3. Select properties of Local Area Network.
4. In general tab, select tcp/ip internet protocol and click on properties.
5. Here select “obtain ip address automatically”.
6. Set preferred DNS as server’s ip address.
7. Click ok.

For checking:

Go to run, type cmd, give command ipconfig.

Here you can see your ip address, subnet mask and gateway.

CONCLUSION: - Configuration of DHCP service is completed on windows 2008 server
EXPERIMENT NO. - 13


AIM : - To Study & Configure Print Server

OBJECTIVE:-

THEORY:-

To Add Printer service:-

- Click on Start
- Open Server Manager
- click on Add and share a printer on the network
- Click next
- Add a new printer using an existing port:
  - click next
  - use an existing printer driver on the computer
  - click next
- Select share this Printer then click next
- Click on finish

To deploy printer connections by using Group Policy on computers that are running versions of Windows released prior to Windows Vista, you must add the PushPrinterConnections.exe utility to a computer startup script or to a user logon script. The PushPrinterConnections.exe utility reads the printer connection settings from Group Policy and adds the appropriate printer connections to the computer or user account (or updates existing connections).

The PushPrinterConnections.exe file automatically detects and exits on computers running Windows Vista or Windows Server 2008. These computers have
built-in support for printer connections that are deployed using Group Policy, so you can safely deploy this file to all the client computers in your organization.

To add the PushPrinterConnections.exe file to startup or logon scripts

Open the GPMC.

In the GPMC console tree, navigate to the domain or organizational unit (OU) that stores the computer or user accounts where you want to deploy the PushPrinterConnections.exe utility.

Right-click the GPO that contains the printer connections that you want to deploy using Group Policy, and then click Edit.

Navigate to one of the following locations:

1. If the printer connections are deployed per-computer, go to Computer Configuration, Policies, Windows Settings, Scripts (Startup/Shutdown).

2. If the printer connections are deployed per-user, go to User Configuration, Policies, Windows Settings, Scripts (Logon/Logoff).


4. Right-click Startup or Logon, and then click Properties.

5. In the Startup Properties or Logon Properties dialog box, click Show Files. The Startup or Logon window appears.

6. Copy the PushPrinterConnections.exe file from the %WINDIR%\System32 folder to the Startup or Logon window. This adds the utility to the GPO, where it will replicate to the other domain controllers with the Group Policy settings.
7. In the Startup Properties or Logon Properties dialog box, click Add. The Add Script dialog box appears.

8. In the Script Name box, type: PushPrinterConnections.exe


10. Log files are written to %WINDIR%\temp\ppcMachine.log (for per-computer connections) and to %temp%\ppcUser.log (for per-user connections) on the computer where the policy is applied.

11. In the Add Script dialog box, click OK.

12. In the Startup Properties or Logon Properties dialog box, click OK.

13. Use GPMC to link the GPO to other OUs or domains to which you want to deploy the PushPrinterConnections.exe utility.

**CONCLUSION:** Thus we configured Print Server on windows 2008 server.
EXPERIMENT NO. - 14

**TITLE:** - Linux- Study and management of users and groups.

**AIM** :: To Create groups, users and understanding the concepts of Linux operating system.

**OBJECTIVE:** - Creating groups, users and understanding the concepts of Linux operating system.

**THEORY:-**

User accounts are used within computer environments to verify the identity of the person using a computer system. By checking the identity of a user, the system is able to determine if the user is permitted to log into the system and, if so, which resources the user is allowed to access.

Groups are logical constructs that can be used to cluster user accounts together for a specific purpose. For instance, if a company has a group of system administrators, they can all be placed in a system administrator group with permission to access key resources and machines. Also, through careful group creation and assignment of privileges, access to restricted resources can be maintained for those who need them and denied to others.

After a normal user account is created, the user can log into the system and access any applications or files they are permitted to access. Red Hat Linux determines whether or not a user or group can access these resources based on the permissions assigned to them.

There are three permissions for files, directories, and applications. The following lists the symbols used to denote each, along with a brief description:

- **r** — Indicates that a given category of user can read a file.
- **w** — Indicates that a given category of user can write to a file.
- **x** — Indicates that a given category of user can execute the file.

A fourth symbol (-) indicates that no access is permitted.
Each of the three permissions are assigned to three defined categories of users. The categories are:

- **owner** — The owner of the file or application.
- **group** — The group that owns the file or application.
- **everyone** — All users with access to the system.

One can easily view the permissions for a file by invoking a long format listing using the command `ls -l`. For instance, if the user **juan** creates an executable file named **foo**, the output of the command `ls -l foo` would look like this:

```
-rwxrwxr-x 1 juan juan 0 Sep 26 12:25 foo
```

The permissions for this file are listed at the start of the line, starting with `rwx`. This first set of symbols define owner access. The next set of `rwx` symbols define group access, with the last set of symbols defining access permitted for all other users.

This listing indicates that the file is readable, writable, and executable by the user who owns the file (user **juan**) as well as the group owning the file (which is a group named **juan**). The file is also world-readable and world-executable, but not world-writable.

One important point to keep in mind regarding permissions and user accounts is that every application run on Red Hat Linux runs in the context of a specific user. Typically, this means that if user **juan** launches an application, the application runs using user **juan**'s context. However, in some cases the application may need more access in order to accomplish a task. Such applications include those that edit system settings or log in users. For this reason, special permissions have been created.

There are three such special permissions within Red Hat Linux. They are as follows:

- **setuid** — used only for applications, this permission indicates that the application runs as the owner of the file and not as the user executing the application. It is indicated by the character **s** in place of the **x** in the owner category. If the owner of the file does not have execution permissions, the **S** is capitalized.

- **setgid** — used primarily for applications, this permission indicates that the application runs as the group owning the file and not as the group executing the application. If applied to a directory, all files a user creates within the directory
are owned by the group who owns the directory, rather than by the user's private group. It is indicated by the character s in place of the x in the group category. If the group owner of the file or directory does not have execution permissions, the S is capitalized.

**sticky bit** — used primarily on directories, this bit dictates that a file created in the directory can be removed only by the user who created the file. It is indicated by the character t in place of the x in the everyone category. In Red Hat Linux the sticky bit is set by default on the /tmp/ directory for exactly this reason.

Another point worth noting is that user account and group names are primarily for people's convenience. Internally, the system uses numeric identifiers. For users, this identifier is known as a UID, while for groups the identifier is known as a GID. Programs that make user or group information available to users translate the UID/GID values into their more human-readable counterparts.

**Practical Work:**

Add a new user and assign them to be members of the group "accounting":

```
useradd -m -g accounting user2
```

Add a new user and assign them to be members of the initial group "accounting" and supplementary group "floppy":

```
useradd -m -g accounting -G floppy user1
```

**Group Commands:**

- `gpasswd`: administer the /etc/group file
- `groupadd`: Create a new group
  
  Format: `groupadd [-g gid [-o]] [-f] [-K KEY=VALUE] group`
  
  Example: `groupadd accounting`
- `groupmod`: Modify a group
  
  Format: `groupmod [-g gid [-o ]] [-n new_group_name] group`
  
  Example: `- Change name of a group: groupmod -n accounting nerdyguys`
- `groupdel`: Delete a group
  
  Example: `groupdel accounting`
- `vigr`: Edit the group file /etc/group with vi. No arguments specified.

Permissions may be viewed by issuing the command: Is -l file-name
- File can be written by yourself and members of the group. Others may only view it.
  -rw-rw-r-- user group file-size date file-name
- Directory is completely open for read/write:
  drwxrwxrwx user group file-size date directory-name
- File can only be accessed by owner (user):
  -rwx------ user group file-size date file-name

Where the first block of "rwx" represents the permissions for the user (u), the second is for the group (g) and the third is for others (o). The "-" represents no access for that access placeholder for user, group or other.

Permissions may be granted using human readable assignments "rwx" or octal codes.

<table>
<thead>
<tr>
<th>Description</th>
<th>Abbreviation</th>
<th>Octal code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read access</td>
<td>R</td>
<td>4</td>
</tr>
<tr>
<td>Write (change) permission</td>
<td>W</td>
<td>2</td>
</tr>
<tr>
<td>Execute script of binary executable</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>Read and Execute</td>
<td>Rx</td>
<td>5</td>
</tr>
<tr>
<td>Read and Write</td>
<td>Rw</td>
<td>6</td>
</tr>
<tr>
<td>Read, Write and Execute</td>
<td>Rwx</td>
<td>7</td>
</tr>
</tbody>
</table>

Use of octal assignment does not add or remove permission, but assigns the permission explicitly.

**CONCLUSION** :- Thus we studied Linux- management of users and groups.
EXPERIMENT NO. - 15

TITLE: Configuration of FTP and Secure Shell in Linux

AIM: To Create FTP and Secure Shell in Linux operating system.

OBJECTIVE: Understanding the FTP and Secure Shell the concepts of Linux operating system.

THEORY:

A. Ubuntu Linux comes with various ftp servers to setup FTP service such as:

1. proftpd - Versatile, virtual-hosting FTP daemon
2. vsftpd - The Very Secure FTP Daemon
3. ftpd - FTP server
4. wu-ftpd - powerful and widely used FTP server
5. wzdftp - A portable, modular, small and efficient ftp server
6. pure-ftpd - Pure-FTPD FTP server

But vsftpd is simple and quite secure FTP server

Vsftpd is the Very Secure File Transfer Protocol Daemon. The server can be launched via a super-server such as inetd or xinetd. Alternatively, vsftpd can be launched in standalone mode, in which case vsftpd itself will listen on the network.

=> Default ftp port : 21
=> Default configuration file : /etc/vsftpd.conf

The configuration of the vsftpd FTP service (read as daemon) simply requires three steps.

Step # 1: Install vsftpd

Type apt-get command to install vsftpd

$ sudo apt-get install vsftpd

Step # 2: Configure /etc/vsftpd.conf

The default vsftpd configuration file is /etc/vsftpd.conf. You need to edit this file using text editor such as vi:
$ sudo vi /etc/vsftpd.conf

Add the following line (uncomment line) to the vsftpd configuration file:

local_enable=YES

This config directive will allow local users to log in via ftp

If you would like to allow users to upload file, add the following to the file:

write_enable=YES

For security you may restrict local users to their home directories. Add the following to the file:

chroot_local_user=YES

Save and close the file.

**Step # 3: Restart vsftpd**

To restart vsftpd type the command:

$ sudo /etc/init.d/vsftpd restart

**Secure Shell:**

SSH (Secure SHell) is the secure way to connect over the Internet. A free version of SSH called OpenSSH is available as the ssh package in Debian.

**Basics of SSH**

First install the OpenSSH server and client.

```
# apt-get update
# apt-get install ssh
```

`/etc/ssh/sshd_not_to_be_run` must not be present if one wishes to run the OpenSSH server.

SSH has two authentication protocols:

- **SSH protocol version 1:**
  - Potato version only supports this protocol.
  - available authentication methods:
    - RSAAuthentication: RSA identity key based user authentication
    - RhostsAuthentication: .rhosts based host authentication (insecure, disabled)
    - RhostsRSAAuthentication: .rhosts authentication combined with RSA host key (disabled)
    - ChallengeResponseAuthentication: RSA challenge-response authentication
    - PasswordAuthentication: password based authentication

- **SSH protocol version 2:**
  - post-Woody versions use this as the primary protocol.
  - available authentication methods:
- PubkeyAuthentication: public key based user authentication
- HostbasedAuthentication: .rhosts or /etc/hosts.equiv authentication combined with public key client host authentication (disabled)
- ChallengeResponseAuthentication: challenge-response authentication
- PasswordAuthentication: password based authentication

**Installation steps:**

1. goto configuration file and edit
   `vim /etc/ssh/sshd_config`
2. check the Authentication is on.
3. service sshd restart
4. chkconfig sshd on
5. goto console and connect to server using command
   `ssh ip of machine`

**Conclusion:**

Thus, We configured ftp and ssh.
EXPERIMENT NO. - 16

TITLE: - Configuration of Mail Server in Linux

AIM: - To Configure Mail server in Linux operating system.

OBJECTIVE: Understanding the Mail Server in of Linux operating system.

THEORY:-

There are many reasons to host your own email. Perhaps you don't like the limits placed on you by your current ISP. Maybe they aren't willing to host the domain you want, or give you the access you want. And if they do fit your needs, they want to charge a small fortune. Maybe you want complete privacy. Or perhaps you just want to access your email from anywhere using a web-based frontend.

Installation Steps:-

Configuring sendmail:

Sendmail is a general purpose internetwork email routing facility that supports many kinds of mail-transfer and -delivery methods, including the Simple Mail Transfer Protocol (SMTP) used for email transport over the Internet.

1. yum install sendmail-cf
2. Edit the file /etc/mail/sendmail.mc and add the following lines. Make sure you set your mail server domain name where it's bolded:

   MASQUERADE_AS(yourdomain.com)dnl
   MASQUERADE_DOMAIN(yourdomain.com)dnl

   In the same file /etc/mail/sendmail.mc remove the "dnl" from the beginning of the lines so it will look like this:

   LOCAL_DOMAIN('localhost.localdomain')dnl
   FEATURE(masquerade_envelope)dnl
   FEATURE(masquerade_entire_domain)dnl

3. Save the file and compile it using m4:
m4 /etc/mail/sendmail.mc > /etc/sendmail.cf
4. Send Sendmail a -HUP signal using kill or simply restart the daemon for the configuration changes to take effect:

5. service sendmail restart
6. `chkconfig sendmail on`

**Configuring Dovecot:**

To make remote users able to send and receive emails, we need to install and configure POP or IMAP server. Our choice here will be Dovecot.

1. First install it with the following command

```
yum install dovecot
```

2. Open nano dovecot.conf

```
nano /etc/dovecot.conf
```

3. And uncomment the following line

```
protocols = imap imaps pop3 pop3s
```

4. Save it & Start dovecot

```
/etc/init.d/dovecot start
```

**Testing your configuration using mutt:**

```
mutt -s "Test Email" you@somewhere.com < /dev/null
```

Mutt provides graphical user interface for sending and receiving mails.

**Conclusion:**- Thus, we studied how to implement mail server in linux.
EXPERIMENT NO. - 17

**TITLE:** - Configuration of DNS and DHCP in Linux

**AIM:** - To Configure DNS and DHCP in Linux operating system.

**OBJECTIVE:** Understanding the DNS and DHCP in the Linux operating system.

**THEORY:**

Domain Name System (DNS) converts the name of a Web site to an IP address. This step is important, because the IP address of a Web site's server, not the Web site's name, is used in routing traffic over the Internet.

BIND is an acronym for the Berkeley Internet Name Domain project, which is a group that maintains the DNS-related software suite that runs under Linux. The most well known program in BIND is named, the daemon that responds to DNS queries from remote machines.

A DNS client doesn't store DNS information; it must always refer to a DNS server to get it. The only DNS configuration file for a DNS client is the /etc/resolv.conf file, which defines the IP address of the DNS server it should use. You shouldn't need to configure any other files. You'll become well acquainted with the /etc/resolv.conf file.

DNS resolution maps a fully qualified domain name (FQDN), such as www.linuxhomelnetworking.com, to an IP address. This is also known as a forward lookup. The reverse is also true: By performing a reverse lookup, DNS can determining the fully qualified domain name associated with an IP address.

**Setting up your DNS server:**

1. Get Bind Package from Internet by using command Bind version no.

You can use the **chkconfig** command to get BIND configured to start at boot

```
[root@bigboytmp]# chkconfig named on
```

To start, stop, and restart BIND after booting, use:

```
[root@bigboytmp]# /etc/init.d/named start
[root@bigboytmp]# /etc/init.d/named stop
[root@bigboytmp]# /etc/init.d/named restart
```

Remember to restart the BIND process every time you make a change to the configuration file for the changes to take effect on the running process.
Configuring resolv.conf:

You'll have to make your DNS server refer to itself for all DNS queries by configuring the /etc/resolv.conf file to reference localhost only.

```bash
nameserver 127.0.0.1
```

Creating a named.conf Base Configuration

The /etc/named.conf file contains the main DNS configuration and tells BIND where to find the configuration, or zone files for each domain you own. This file usually has two zone areas:

- Forward zone file definitions list files to map domains to IP addresses.
- Reverse zone file definitions list files to map IP addresses to domains.

Some versions of BIND will come with a /etc/amd.conf file configured to work as a caching nameserver which can be converted to an authoritative nameserver by adding the correct references to your zone files. Please proceed to the next section if this is the case with your version of BIND.

In other cases the named.conf configuration file may be hard to find. Some versions of Linux install BIND as a default caching nameserver using a file names /etc/named.caching-nameserver.conf for its configuration. In such cases BIND becomes an authoritative nameserver when a correctly configured /etc/named.conf file is created.

Fortunately BIND comes with samples of all the primary files you need.

The first task is to make sure your DNS server will listening of requests on all the required network interfaces. The options section of named.conf may be configured to listen exclusively on its internal hidden localhost interface with an IP address of 127.0.0.1 as we see in this example.

```bash
# File: /etc/named.conf

options {
    listen-on port 53 { 127.0.0.1; };
};
```

If other devices are going to rely on your server for queries, then you’ll need to either change this or add a selected number of IP addresses on your server. In this example, we allow queries on any interface.

```bash
listen-on port 53 { any; };
```

In this example, we allow queries on localhost and address 192.168.1.100.

```bash
listen-on port 53 { 127.0.0.1; 192.168.1.100; };
```
Note: Always make sure localhost, 127.0.0.1 is included.

Though it is not required, it is a good practice to configure your DNS server's named.conf file to support BIND views.

**Forward Zone File References in named.conf**

Let’s describe how we point to forward zone files in a typical named.conf file.

In this example the zone file is named my-site.zone, and, although not explicitly stated, the file my-site.zone should be located in the default directory of /var/named/chroot/var/named in a chroot configuration or in /var/named in a regular one. With Debian / Ubuntu, references to the full file path will have to be used. Use the code:

```plaintext
zone "my-web-site.org" {
    type master;
    notify no;
    allow-query { any; };
    file "my-site.zone";
};
```

In addition, you can insert more entries in the named.conf file to reference other Web domains you host. Here is an example for another-site.com using a zone file named another-site.zone.

```plaintext
zone "another-site.com" {
    type master;
    notify no;
    allow-query { any; };
    file "another-site.zone";
};
```

Note: The allow-query directive defines the networks that are allowed to query your DNS server for information on any zone. For example, to limit queries to only your 192.168.1.0 network, you could modify the directive to:

```plaintext
allow-query { 192.168.1.0/24; }
```
Configuring The Zone Files

You need to keep a number of things in mind when configuring DNS zone files:

- In all zone files, you can place a comment at the end of any line by inserting a semi-colon character then typing in the text of your comment.
- By default, your zone files are located in the /var/named or /var/named/chroot/var/named or /etc/bind directories depending on your Linux distribution.
- Each zone file contains a variety of records (SOA, NS, MX, A, and CNAME) that govern different areas of BIND.
- The purpose of a TTL is to reduce the number of DNS queries the authoritative DNS server has to answer. If the TTL is set to three days, then caching servers use the original stored response for three days before making the query again.
- $TTL 3D
- BIND recognizes several suffixes for time-related values. A D signifies days, a W signifies weeks, and an H signifies hours. In the absence of a suffix, BIND assumes the value is in seconds.

Loading Your New Configuration Files

Make sure your configuration files are in the correct locations and the serial numbers of the zone files you may have modified have been updated. If all seems correct, restart BIND named daemon for the configuration to become active.

[root@bigboytmp]# /etc/init.d/named restart

DHCP:

A DHCP Server assigns IP addresses to client computers. This is very often used in enterprise networks to reduce configuration efforts. All IP addresses of all computers are stored in a database that resides on a server machine.

A DHCP server can provide configuration settings using two methods

Address Pool

This method entails defining a pool (sometimes also called a range or scope) of IP addresses from which DHCP clients are supplied their configuration properties dynamically and on a first come first serve basis. When a DHCP client is no longer on the network for a specified period, the configuration is expired and released back to the address pool for use by other DHCP Clients.

MAC Address
This method entails using DHCP to identify the unique hardware address of each network card connected to the network and then continually supplying a constant configuration each time the DHCP client makes a request to the DHCP server using that network device.

1. Install DHCP server in ubuntu

```
sudo apt-get install dhcp3-server
```

This will complete the installation.

2. Configuring DHCP server

If you have two network cards in your Ubuntu server you need to select which interface you want to use for DHCP server listening. By default it listens to eth0.

You can change this by editing /etc/default/dhcp3-server file

```
sudo vi /etc/default/dhcp3-server
```

Find this line

```
INTERFACES="eth0?"?
```

Replace with the following line

```
INTERFACES="eth1?"
```

Save and exit. This is optional.

3. Next you need to make a backup copy of /etc/dhcp3/dhcpd.conf file

```
cp /etc/dhcp3/dhcpd.conf /etc/dhcp3/dhcpd.conf.back
```

Edit /etc/dhcp3/dhcpd.conf file using the following command

```
sudo vi /etc/dhcp3/dhcpd.conf
```

Using address pool method

4. You need to change the following sections in /etc/dhcp3/dhcpd.conf file

```
default-lease-time 600;
max-lease-time 7200;
option subnet-mask 255.255.255.0;
```
option broadcast-address 192.168.1.255;

option routers 192.168.1.254;

option domain-name-servers 192.168.1.1, 192.168.1.2;

option domain-name “yourdomainname.com”;

subnet 192.168.1.0 netmask 255.255.255.0 {
  range 192.168.1.10 192.168.1.200;
}

save and exit the file

This will result in the DHCP server giving a client an IP address from the range 192.168.1.10-192.168.1.200. It will lease an IP address for 600 seconds if the client doesn’t ask for a specific time frame. Otherwise the maximum (allowed) lease will be 7200 seconds. The server will also “advise” the client that it should use 255.255.255.0 as its subnet mask, 192.168.1.255 as its broadcast address, 192.168.1.254 as the router/gateway and 192.168.1.1 and 192.168.1.2 as its DNS servers.

5. Now you need to restart dhcp server using the following command

sudo /etc/init.d/dhcp3-server restart

Configure Ubuntu DHCP Client

If you want to configure your ubuntu desktop as DHCP client following this procedure

You need to open /etc/network/interfaces file

sudo vi /etc/network/interfaces

make sure you have the following lines (eth0 is an example)

auto lo eth0

iface eth0 inet dhcp

iface lo inet loopback

Save and exit the file

You need to restart networking services using the following command
sudo /etc/init.d/networking restart

How to find DHCP server IP address

You need to use the following commands

sudodhcclient

**Conclusion:** Thus, We configured the DHCP and DNS Server in linux.
EXPERIMENT NO. - 18

TITLE:--Configuration of HTTP server in Linux.

AIM:--To Configure HTTP Server in Linux operating system.

OBJECTIVE: Understanding the HTTP Server in Linux operating system.

THEORY:

The Apache HTTP Server is a “heavy-duty” network server that Subversion can leverage. Via a custom module, httpd makes Subversion repositories available to clients via the WebDAV/DeltaV protocol, which is an extension to HTTP 1.1. This protocol takes the ubiquitous HTTP protocol that is the core of the World Wide Web, and adds writing—specifically, versioned writing—capabilities. The result is a standardized, robust system that is conveniently packaged as part of the Apache 2.0 software.

The Apache HTTP Server, commonly referred to as Apache is web server software notable for playing a key role in the initial growth of the World Wide Web. Apache supports a variety of features, many implemented as compiled modules which extend the core functionality. These can range from server-side programming language support to authentication schemes. Some common language interfaces support Perl, Python, Tcl, and PHP.

Installation Steps:--

1. Install Apache HTTP Server
   `yum install httpd` OR `rpm -qa httpd`

2. Goto document root

   `Cd /var/www/html`

3. In that create sample html file and any content.
4. Open the httpd configuration file located at `/etc/httpd/conf/httpd.conf`
5. Un-comment the line containing the text `NameVirtualHost *:80`
6. `<VirtualHost IP of server:80>
    ServerAdmin root@example.com
    DocumentRoot /var/www/html/
    ServerName IP of Server OR Name
     7. service httpd -t
     8. service httpd restart`
Conclusion: we studied how to implement HTTP server in Linux.

EXPERIMENT NO. - 19

TITLE :- Study of Windows 2008 Security system.

AIM :- To learn about Windows 2003/2008 Security system.

OBJECTIVE :-

THEORY :- Preventing unauthorized access to sensitive data is essential in any environment in which multiple users have access to the same physical or network resources. Operating system security includes obvious mechanisms such as accounts, passwords, and file protection. It also includes protecting the operating system from corruption, preventing less privileged users from performing actions (rebooting the computer, for example), and not allowing user programs to adversely affect the programs of other users or the operating system.

These are the core components and databases that implement Windows security:

- Security reference monitor (SRM) A component in the Windows executive (%SystemRoot%\System32\Ntoskrnl.exe) that is responsible for defining the access token data structure to represent a security context, performing security
access checks on objects, manipulating privileges (user rights), and generating any resulting security audit messages.

- **Local Security Authority subsystem (Lsass)** A user-mode process running the image `%SystemRoot%\System32\Lsass.exe` that is responsible for the local system security policy (such as which users are allowed to log on to the machine, password policies, privileges granted to users and groups, and the system security auditing settings), user authentication, and sending security audit messages to the Event Log. The Local Security Authority service (Lsasrv—%SystemRoot%\System32\Lsasrv.dll), a library that Lsass loads, implements most of this functionality.

- **Lsass policy database** A database that contains the local system security policy settings. This database is stored in the registry under HKLM\SECURITY. It includes such information as what domains are entrusted to authenticate logon attempts, who has permission to access the system and how (interactive, network, and service logons), who is assigned which privileges, and what kind of security auditing is to be performed. The Lsass policy database also stores “secrets” that include logon information used for cached domain logons and Windows service user-account logons.

- **Security Accounts Manager (SAM) service** A set of subroutines responsible for managing the database that contains the user names and groups defined on the local machine. The SAM service, which is implemented as %SystemRoot%\System32\Samsrv.dll, runs in the Lsass process.

- **SAM database** A database that on systems not functioning as domain controllers contains the defined local users and groups, along with their passwords and other attributes. On domain controllers, the SAM stores the system’s administrator recovery account definition and password. This database is stored in the registry under HKLM\SAM.

- **Active Directory** A directory service that contains a database that stores information about objects in a domain. A domain is a collection of computers and their associated security groups that are managed as a single entity. Active Directory stores information about the objects in the domain, including users, groups, and computers. Password information and privileges for domain
users and groups are stored in Active Directory, which is replicated across the computers that are designated as domain controllers of the domain. The Active Directory server, implemented as %SystemRoot%\System32\Ntdsa.dll, runs in the Lsass process. For more information on Active Directory.

- Authentication packages These include dynamic-link libraries (DLLs) that run both in the context of the Lsass process and client processes and that implement Windows authentication policy. An authentication DLL is responsible for checking whether a given user name and password match, and if so, returning to the Lsass information detailing the user’s security identity, which Lsass uses to generate a token.

- Interactive logon manager (Winlogon) A user-mode process running %System-Root%\System32\Winlogon.exe that is responsible for responding to the SAS and for managing interactive logon sessions.

- Logon user interface (LogonUI) A user-mode process that presents users with the user interface they can use to authenticate themselves on the system. Uses credential providers to query user credentials through various methods.

- Credential providers (CPs) In-process COM objects that run in the LogonUI process and used to obtain a user’s name and password, smartcard PIN, or biometric data (such as a fingerprint). The standard CPs are,

  %SystemRoot%\System32\authui.dll and %SystemRoot%\System32\SmartcardCredentialProvider.dll

- Network logon service (Netlogon) A Windows service (%SystemRoot%\System32\Netlogon.dll) that sets up the secure channel to a domain controller, over which security requests—such as an interactive logon (if the domain controller is running Windows NT 4) or LAN Manager and NT LAN Manager (v1 and v2) authentication validation—are sent.

**Security Descriptors and Access Control**

Tokens, which identify a user’s credentials, are only part of the object security equation. Another part of the equation is the security information associated with an object, which specifies who can perform what actions on the object. The data structure for this information is called a security descriptor. A security descriptor consists of the following attributes:
- **Revision number**: The version of the SRM security model used to create the descriptor.
- **Flags**: Optional modifiers that define the behavior or characteristics of the descriptor.
- **Owner SID**: The owner’s security ID.
- **Group SID**: The security ID of the primary group for the object (used only by POSIX).
- **Discretionary access control list (DACL)**: Specifies who has what access to the object.
- **System access control list (SACL)**: Specifies which operations by which users should be logged in the security audit log and the explicit integrity level of an object.

**TABLE 6-5 Security Descriptor Flags**

<table>
<thead>
<tr>
<th>Flag</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SE_OWNER_DEFAULTED</td>
<td>Indicates a security descriptor with a default owner security identifier (SID). Use this bit to find all the objects that have default owner permissions set.</td>
</tr>
<tr>
<td>SE_GROUP_DEFAULTED</td>
<td>Indicates a security descriptor with a default group SID. Use this bit to find all the objects that have default group permissions set.</td>
</tr>
<tr>
<td>SE_DACL_PRESENT</td>
<td>Indicates a security descriptor that has a DACL. If this flag is not set, or if this flag is set and the DACL is NULL, the security descriptor allows full access to everyone.</td>
</tr>
<tr>
<td>SE_DACL_DEFAULTED</td>
<td>Indicates a security descriptor with a default DACL. For example, if an object creator does not specify a DACL, the object receives the default DACL from the access token of the creator. This flag can affect how the system treats the DACL with respect to access control entry (ACE) inheritance. The system ignores this flag if the SE_DACL_PRESENT flag is not set.</td>
</tr>
<tr>
<td>SE_SACL_PRESENT</td>
<td>Indicates a security descriptor that has a system access control list (SACL).</td>
</tr>
<tr>
<td>SE_SACL_DEFAULTED</td>
<td>Indicates a security descriptor with a default SACL. For example, if an object creator does not specify an SACL, the object receives the default SACL from the access token of the creator. This flag can affect how the system treats the SACL with respect to ACE inheritance. The system ignores this flag if the SE_SACL_PRESENT flag is not set.</td>
</tr>
<tr>
<td>SE_DACL_UNTRUSTED</td>
<td>Indicates that the ACL pointed to by the DACL of the security descriptor was provided by an untrusted source. If this flag is set and a compound ACE is encountered, the system will substitute known valid SIDs for the server SIDs in the ACEs.</td>
</tr>
</tbody>
</table>
### Account Rights

Account rights are not enforced by the security reference monitor, nor are they stored in tokens. The function responsible for logon is `LsaLogonUser`. Winlogon, for example, calls the `LogonUser` API when a user logs on interactively to a computer, and `LogonUser` calls `LsaLogonUser`. The function takes a parameter that indicates the type of logon being performed, which includes interactive, network, batch, service, Terminal Server client, and unlock.

In response to logon requests, the Local Security Authority (LSA) retrieves account rights assigned to a user from the LSA policy database at the time that a user attempts to log on to the system. LSA checks the logon type against the account rights assigned to the user account logging on and denies the logon if the account does not have the right that permits the logon type or it has the right that denies the logon type. Table 6-7 lists the user rights defined by Windows.

Windows applications can add and remove user rights from an account by using the `LsaAddAccountRights` and `LsaRemoveAccountRights` functions, and they can

<table>
<thead>
<tr>
<th>Account Right</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SE_SERVER_SECURITY</strong></td>
<td>Requests that the provider for the object protected by the security descriptor should be a server ACL based on the input ACL, regardless of its source (explicit or defaulting). This is done by replacing all the GRANT ACEs with compound ACEs granting the current server access. This flag is only meaningful if the subject is impersonating.</td>
</tr>
<tr>
<td><strong>SE_DAACL_AUTO_INHERIT_REQ</strong></td>
<td>Requests that the provider for the object protected by the security descriptor automatically propagate the DACL to existing child objects. If the provider supports automatic inheritance, the DACL is propagated to any existing child objects, and the SE_DAACL_AUTO_INHERITED bit in the security descriptor of the parent and child objects is set.</td>
</tr>
<tr>
<td><strong>SE_SAACL_AUTO_INHERIT_REQ</strong></td>
<td>Requests that the provider for the object protected by the security descriptor automatically propagate the SACL to existing child objects. If the provider supports automatic inheritance, the SACL is propagated to any existing child objects, and the SE_SAACL_AUTO_INHERITED bit in the security descriptor of the parent and child objects is set.</td>
</tr>
<tr>
<td><strong>SE_DAACL_AUTO_INHERITED</strong></td>
<td>Indicates a security descriptor in which the DACL is set up to support automatic propagation of inheritable ACEs to existing child objects. The system sets this bit when it performs the automatic inheritance algorithm for the object and its existing child objects.</td>
</tr>
<tr>
<td><strong>SE_SAACL_AUTO_INHERITED</strong></td>
<td>Indicates a security descriptor in which the SACL is set up to support automatic propagation of inheritable ACEs to existing child objects. The system sets this bit when it performs the automatic inheritance algorithm for the object and its existing child objects.</td>
</tr>
<tr>
<td><strong>SE_DAACL_PROTECTED</strong></td>
<td>Prevents the DACL of a security descriptor from being modified by inheritable ACEs.</td>
</tr>
<tr>
<td><strong>SE_SAACL_PROTECTED</strong></td>
<td>Prevents the SACL of a security descriptor from being modified by inheritable ACEs.</td>
</tr>
<tr>
<td><strong>SE_RM_CONTROL_VALID</strong></td>
<td>Indicates that the resource control manager bits in the security descriptor are valid. The resource control manager bits are 8 bits in the security descriptor structure that contains information specific to the resource manager accessing the structure.</td>
</tr>
<tr>
<td><strong>SE_SELF_RELATIVE</strong></td>
<td>Indicates a security descriptor in self-relative format, with all the security information in a contiguous block of memory. If this flag is not set, the security descriptor is in absolute format.</td>
</tr>
</tbody>
</table>
determine what rights are assigned to an account with LsaEnumerateAccountRights.

**CONCLUSION** :- Thus we studied Windows 2008 Security system.

**EXPERIMENT NO. - 20**

**TITLE** :- Study of Windows 2008 I/O system.

**AIM** :- To learn about Windows 2008 I/O system.

**OBJECTIVE** :-

**THEORY** :-

The Windows I/O system consists of several executive components that together manage hardware devices and provide interfaces to hardware devices for applications and the system.

The design goals for the Windows I/O system are to provide an abstraction of devices, both hardware (physical) and software (virtual or logical), to applications with the following features:

- Uniform security and naming across devices to protect shareable resources.
■ High-performance asynchronous packet-based I/O to allow for the implementation of scalable applications.
■ Services that allow drivers to be written in a high-level language and easily ported between different machine architectures.
■ Layering and extensibility to allow for the addition of drivers that transparently modify the behavior of other drivers or devices, without requiring any changes to the driver whose behavior or device is modified.
■ Dynamic loading and unloading of device drivers so that drivers can be loaded on-demand and not consume system resources when unneeded.
■ Support for Plug and Play, where the system locates and installs drivers for newly detected hardware, assigns them hardware resources they require, and also allows applications to discover and activate device interfaces.
■ Support for power management so that the system or individual devices can enter low power states.
■ Support for multiple installable file systems, including FAT, the CD-ROM file system (CDFS), the Universal Disk Format (UDF) file system, and the Windows file system (NTFS).
■ Windows Management Instrumentation (WMI) support and diagnosability so that drivers can be managed and monitored through WMI applications and scripts.
The I/O manager is the heart of the I/O system. It connects applications and system components to virtual, logical, and physical devices, and it defines the infrastructure that supports device drivers.

A device driver typically provides an I/O interface for a particular type of device. Device drivers receive commands routed to them by the I/O manager that are directed at devices they manage, and they inform the I/O manager when those commands complete. Device drivers often use the I/O manager to forward I/O commands to other device drivers that share in the implementation of a device’s interface or control.

The PnP manager works closely with the I/O manager and a type of device driver called a bus driver to guide the allocation of hardware resources as well as to detect and respond to the arrival and removal of hardware devices. The PnP manager and bus drivers are responsible for loading a device’s driver when the device is detected. When a device is added to a system that doesn’t have an appropriate device driver, the executive Plug and Play component calls on the device installation services of a user-mode PnP manager.
The power manager also works closely with the I/O manager to guide the system, as well as individual device drivers, through power-state transitions.

Windows Management Instrumentation support routines, called the Windows Driver Model (WDM) WMI provider, allow device drivers to indirectly act as providers, using the WDM WMI provider as an intermediary to communicate with the WMI service in user mode.

The registry serves as a database that stores a description of basic hardware devices attached to the system as well as driver initialization and configuration settings.

INF files, which are designated by the .inf extension, are driver installation files. INF files are the link between a particular hardware device and the driver that assumes primary control of the device. They are made up of script like instructions describing the device they correspond to, the source and target locations of driver files, required driver-installation registry modifications, and driver dependency information. Digital signatures that Windows uses to verify that a driver file has passed testing by the Microsoft Windows Hardware Quality Labs (WHQL) are stored in .cat files.

The hardware abstraction layer (HAL) insulates drivers from the specifics of the processor and interrupt controller by providing APIs that hide differences between platforms. In essence, the HAL is the bus driver for all the devices on the computer's motherboard that aren't controlled by other drivers.

I/O Priorities

The Windows I/O manager internally includes support for five I/O priorities, but only three of the priorities are used. (Future versions of Windows may support High and Low.)

<table>
<thead>
<tr>
<th>I/O Priority</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical</td>
<td>Memory manager</td>
</tr>
<tr>
<td>High</td>
<td>Not used</td>
</tr>
<tr>
<td>Normal</td>
<td>Normal application I/O</td>
</tr>
<tr>
<td>Low</td>
<td>Not used</td>
</tr>
<tr>
<td>Very Low</td>
<td>Scheduled tasks, Superfetch, defragmenting, content indexing, background activities</td>
</tr>
</tbody>
</table>
Internally, these five I/O priorities are divided into two I/O prioritization modes, called strategies. These are the hierarchy prioritization and the idle prioritization strategies. Hierarchy prioritization deals with all the I/O priorities except Very Low. It implements the following strategy:

- All critical-priority I/O must be processed before any high-priority I/O.
- All high-priority I/O must be processed before any normal-priority I/O.
- All normal-priority I/O must be processed before any low-priority I/O.
- All low-priority I/O is processed after all higher priority I/O.

**CONCLUSION**: Thus we studied Windows 2008 I/O system.