A project report on

Web Authentication & Central Login Management Using NIS Server in Linux

BY

K R I S H N A N A N D

S H R E Y A S U Y O G

S A N G E E T A M O I

HALDIA INSTITUTE OF TECHNOLOGY

Dept of Information Technology

University roll no:
Krishn – 10300212023
Shreya – 10300212051
Sangeeta – 10300212045

B.Tech
Information Technology

Mentor:
Mr Ramkrishna Ghosh

CERTIFICATE

This is to ceritify that the report entitled "Web Authentication and Central Login Management Using NIS and LDAP Server in Linux" submitted by Mr. Krishn Anand (10300212023), Ms. Shreya Suyog (10300212051) and Ms. Sangeeta Moi (10300212045) to Haldia Institute of Technology for the fulfilment of the requirements of the degree of Bachelor of Technology is a record of final year project work carried out by him or her in the Department of Information Technology. The results or findings contained in this thesis have not been submitted in part or full to any other university or institute for the award of any other Degree/Diploma.

Head Of Dapartment
Dr. Soumen Paul

Project Mentor
Mr. Ramkrishn Ghosh

Place: Haldia

CONTENTS

ACKNOWLEDGEMENT

It is a pleasure to acknowledge the help that we received from the project coordinator during the compilation of this project. We are particularly indebted to our mentor **Mr. Ramkrishna Ghosh** for his guidance and constant supervision as well as for providing necessary information regarding the project & also for his support in completing the project. Our thanks and appreciations also goes to our team in developing the project and people who have willingly helped us out with their abilities.

Shreya Suyog
Krishn Anand
Sangeeta Moi

ABSTRACT

Linux is a **multiuser, multitasking** operating system from the ground up, and in this regard the system administrator has flexibility and responsibility far beyond those of other operating systems. Now, Red Hat has employed innovations that extend these duties even for the experienced Linux user.

By definition, the Linux system administrator is the person who has "root" access, which is to say the one who is the system's "**super user**" (or root user). A standard Linux user is limited as to the things he or she can do with the underlying engine of the system. But the "root" user has unfettered access to everything — all user accounts, their home directories, and the files therein; all system configurations; and all files on the system.

Web Authentication

We can also configure Apache to allow access only to specific users. To do this, we can configure our server to use authentication. We will configure authentication based on a flat file that contains usernames and hashed passwords.

LDAP

LDAP (Lightweight Directory Access Protocol) is a set of open protocols used to access centrally stored information over a network. It is based on the X.500 standard for directory sharing. LDAP is sometimes referred to as "X.500 Lite".
LDAP organizes information in a hierarchical manner using directories, enabling anyone to access users account from any machine on the LDAP enabled network.

NIS

A common challenge facing administrators charged with maintaining a network of Linux machines is how to share information across the network while maintaining that information centrally. The Network Information Service (NIS) is one solution to such a challenge.

HISTORY AND ORIGIN

The foundation for linux was laid with programming language **C by Dennis Richie** at Bell Telephone Laboratories in **1969**. This language was developed for use with the UNIX operating system. The UNIX operating system was the first operating system where people from different companies tried to work together to build instead of competing with each other, keeping their efforts secret.

Because of the huge success of UNIX, companies started claiming parts of this operating system in the **1970s**. They succeeded fairly well, and that was the beginning of the development of different **flavours of UNIX.**

As a reaction to the closing of UNIX, **Richard Stallman** of MIT announced in **1984** the **GNU** operating system project. During the 1980s, many common Unix commands, tools, and applications were developed until, in **1991**, the last gap was filled in with the launch of the **Linux kernel** by a student at the University of Helsinki in Finland, **Linus Torvalds.**

Some initiatives started soon to provide ready-to-install Linux distributions. Among the first was MCC Interim Linux, a distribution made available for public download in February 1992, shortly after the release of the Linux kernel itself. In 1993, **Patrick Volkerding** released a distribution called **Slackware**, a distribution that could be downloaded to floppy disk images in the early days.

In 1993, **Marc Ewing and Bob Young** founded **Red Hat**, the first Linux distributor operating as a business. Since then, Red Hat has acquired other companies to integrate specific Linux-related technologies**. Red Hat went public** in **1999**, thus becoming the first Linux-based company on Wall Street.

BASICS ABOUT LINUX

Compared to any other Operating System, Linux can practically be installed on any system with very basic system requirements. There is a version of RHEL Server for almost any hardware platform. That means we can install it on a mainframe computer, a mid-range system, or PC-based server hardware using a 64- or 32-bit architecture. The version we used is Red Hat Enterprise
Linux Server 6.1 for 32-bit. Which can be downloaded from www.redhat.com.

- A CPU capable of handling 32-bit instructions
- 1GB of RAM
- 20GB of available hard disk space
- A DVD drive
- A network card

We can run Red Hat Enterprise Linux with less than this, but we'll miss certain functionality. For instance, we can install RHEL on a machine that has 512MB of
RAM, but we'll lose the graphical user interface.

OPEN SOURCE

Torvalds just needed a license to ensure that the Linux kernel would be free software forever, and he chose to use the GNU General Public License (GPL) for
this purpose. The GPL is a *copyleft license*, which means that derived works can be distributed only under the same license terms. Using GPL made it possible to publish open source software where others could freely add to or modify lines of code.

UNIX PRINCIPLES

UNIX has five principles and every linux/unix distribution must follow them:

1. *Everything is a file*
   There are no drives in Linux like C: or D: as in windows operating systems. In UNIX the hard disk are portioned in the format of dev/sda, dev/sdb, dev/sdb1, dev/sdb2.

2. *Configuration data is stored in text format.*
   In UNIX all the configuration files are stored in text format which can easily be viewed and edited using vi editor or any other text editor. The configuration files are stored in text format so that Open Source meaning persists and user (administrator) can make changes as per need to the .conf files.

3. *Small single purpose programs*
   UNIX provides many small utility that performs a task in many ways. For example touch, vi, cat, nano, gedit are all different inbuilt commands or programs for performing a task that is text editing or creating a new file.

4. *Ability to chain programs*
   In UNIX, we can chain programs together to perform complex task with ease, for example output of one program/command can be the input of other command.

5. *Avoid captive user interface*
   In UNIX, if a command is wrong then a user friendly error is shown and UNIX also provides the flexibility of rectifying the errors by showing the help options.

BOOT SEQUENCE IN LINUX

1. **POST**
   - Power OnShell Test
   - Keyboard, mouse and other hardware are checked at this stage whether they are responding properly or not.
2. **MBR**
   - Master Boot Record
   - It searches for active partitions is system.
3. **Boot Loader**
   - It is loaded in **/boot**
   - Name of the bootloader in RHEL1 to RHEL4 was *LiLo (Linux Loader).*
   - In RHEL5 and RHEL6 it is known as *grub (grand unified bootloader)* whose configuration file is *grub.conf*.
   - In RHEL7 it changed to *grub2.cfg*.
4. **Kernel**
   - It is also found in /boot.
   - The filename of the kernel starts with *vmlinuz* version number.
5. **Initrd**
   - Initrd stands for *initialisation Ram Disk.*
   - It is also found in /boot and it checks the information for the drivers of mouse, keyboards, graphics etc.
   - The filename is *initramfs 2.6.42.6.img*, it is an image based file.
6. **Init**
   - init was found till RHEL5 only.
   - The directory is /etc/inittab.
   - It checks whether the installation is done using GUI(Graphical User Interface) mode or the CLI(Command Line Argument) mode.
   - It is a runnable selection file.
   - If the init file is not found then system can not boot.

TABLE: Various init commands that perform different tasks.

Init	***Action***
0	Shutdown
1	Single User Mode
2	Multiuser mode with CLI and no NFS support
3	Multiuser mode with NFS support
4	(UNUSED)
5	GUI Mode
6	Restart / Reboot

7. Upstart

- Upstart was introduced in RHEL6.
- The system firsl calls the inittab, if it is not found then default boot takes place. So the upstart rescues from crashing our system.

8. Login

- Now the user or the superuser can login successfully into the system.

BASIC CONFIGURATION

Firewall Configuration

The firewall used in linux is known as SELINUX and we need to change its configuration file using the following command.

#vi /etc/sysconfig/selinux

set SELINUX to disabled from enforcing, it disables the selinux services. We need to disable firewall on both server as well as client machine in order to connect to each other without any glitches.

```
1
2 # This file controls the state of SELinux on the system.
3 # SELINUX= can take one of these three values:
4 #     enforcing - SELinux security policy is enforced.
5 #     permissive - SELinux prints warnings instead of enforcing.
6 #     disabled - No SELinux policy is loaded.
7 SELINUX=enforcing
8 # SELINUXTYPE= can take one of these two values:
9 #     targeted - Targeted processes are protected,
10 #     mls - Multi Level Security protection.
11 SELINUXTYPE=targeted
```

Next we need to disable the firewall by going to setup.
#setup

A popup windows opens where we select firewall configuration and press next. Then we remove the mark from [*] enabled by pressing the spacebar. And exit and then restart the system.

Installing YUM and understanding RPM

RPM stands for RedHat Package Manager, it is used to install packages in redhat, but there is a problem using RPM for packages installations as it can not resolve the dependencies witihin. For example we need to install gcc and gcc needs several packages that needs to be installed before hand before installing gcc then RPM can not solve that dependency.
So we require YUM which is Yellowdog Updater Modified, It can resolve the dependencies and automatically install the packages which needs to be installed.
So first we neeed to configure our machine with YUM.

The steps required for doing the same is as follows:

#cp -avf /media/RHEL_6.1\ i386\ Disc\ 1/Packages/ /repo
This command copies all the packages from the installation disc to a directory named as /repo on our hard disk, so we can use it to install packages even if the installation disc is not there.

Next we create a file named as as.repo which is a blank file and we add the lines in it. In the baseurl we provide the path where our packages are saved.

#vi /etc/yum.repos.d/as.repo
[as]
name=as
baseurl=:///repo
enabled=1
gpgcheck=0

We need to install some packages using RPM to configure our machine so that it can start using yum installations.

#cd /repo
#rpm -ivf deltarpm-3.5-0.5.20090913git.el6.i686.rpm
#rpm -ivf python-deltarpm-3.5-0.5.20090913git.el6.i686.rpm
#rpm -ivf createrepo-0.9.8-4.el6.noarch.rpm

Note: In RHEL5 we just need to install only one package which is createrepo instead all the three packages.

After the three packages has been installed we need to create the parent and family of the dependencies using the createrepo command.

#createrepo –v /repo/

To check whetehr there are any bugs or not we run the clean all command as a parameter to yum.

#yum clean all

Once we verified that there is no bug in our process then in order to create a listing or a hierarchy of the dependencies we run the list all command with yum.

#yum list all

Now our server and client is ready to install packages using yum.
We need to do the above procedure on both server as well as client machines.

In redhat 5 the Packages are found in a different location which is
/media/RHEL5…../Servers

UNDERSTANDING APACHE

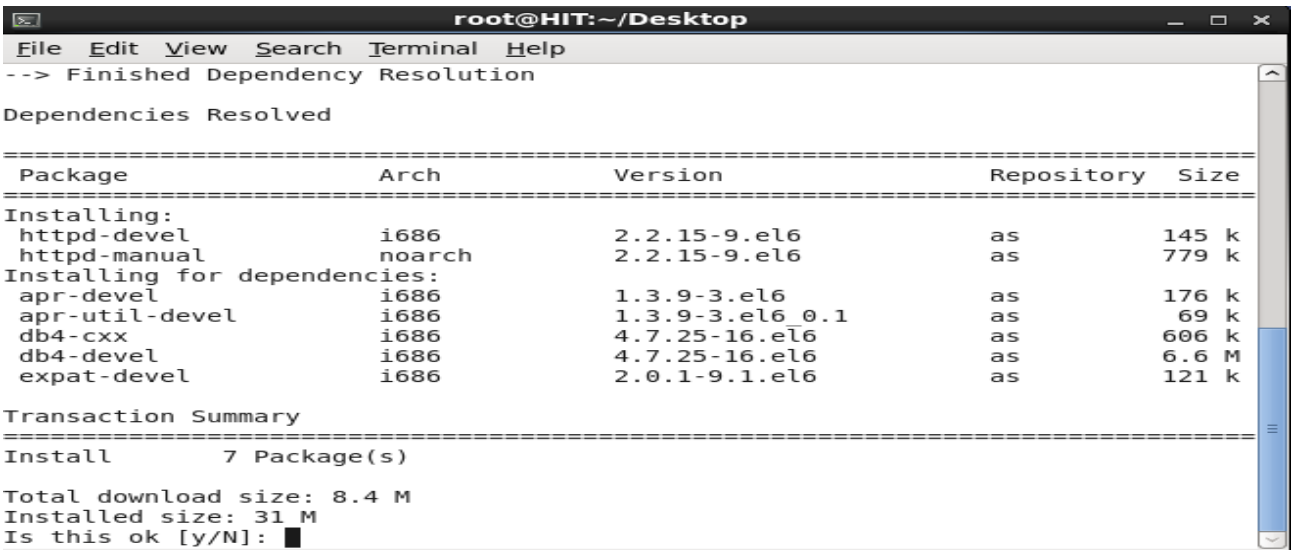
The Apache Web server began life as the NCSA (National Center for Supercomputing Applications) HTTP server. The name "*Apache*" came from the fact that it is "*A Patchy server*." It was based on some existing code and a series of "patch files."

Apache's true standout features are its speed, configurability, stability, and rich feature set. Its configuration information resides in plain text files and uses simple English-language directives. Apache is beset with fewer (known) bugs than other Web servers, particularly closed source servers. Apache is an open source software project.

Apache supports virtual hosts, also known as *multi-homed servers*, which enables a single machine to provide Web services for multiple domains or IP addresses (or hostnames). Apache enables administrators to define multiple directory index files, the default page to display when a Web client requests a directory URL. So, for example, the server can return index.html, index.htm, index.php, or execute a script named index.cgi when a client requests a directory URL, depending on what Apache finds in the requested directory.

INSTALLING APACHE

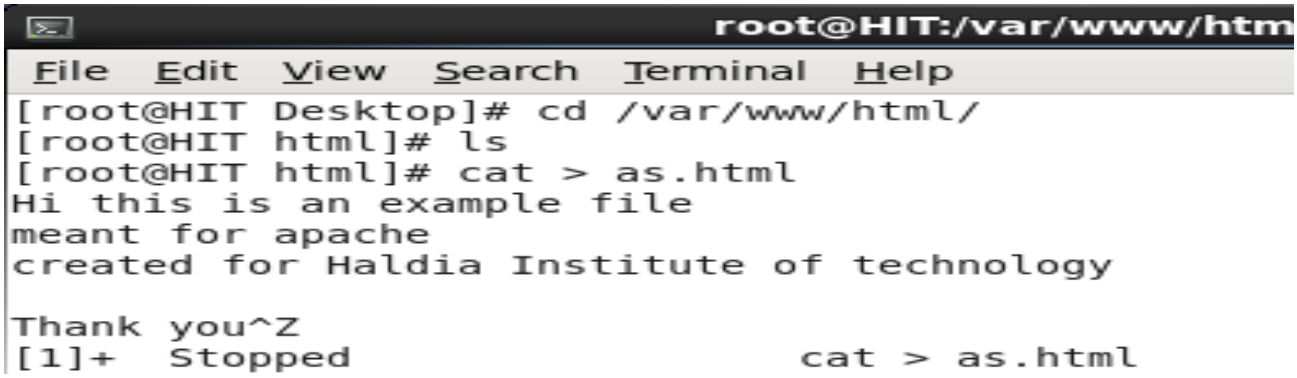
The first and foremost requirement for installing apache is installing the httpd package through yum so that we can later use its services.
To install httpd, we need to run the following command.
]# yum install httpd*
It will install seven packages .We need to press 'y' to continue.



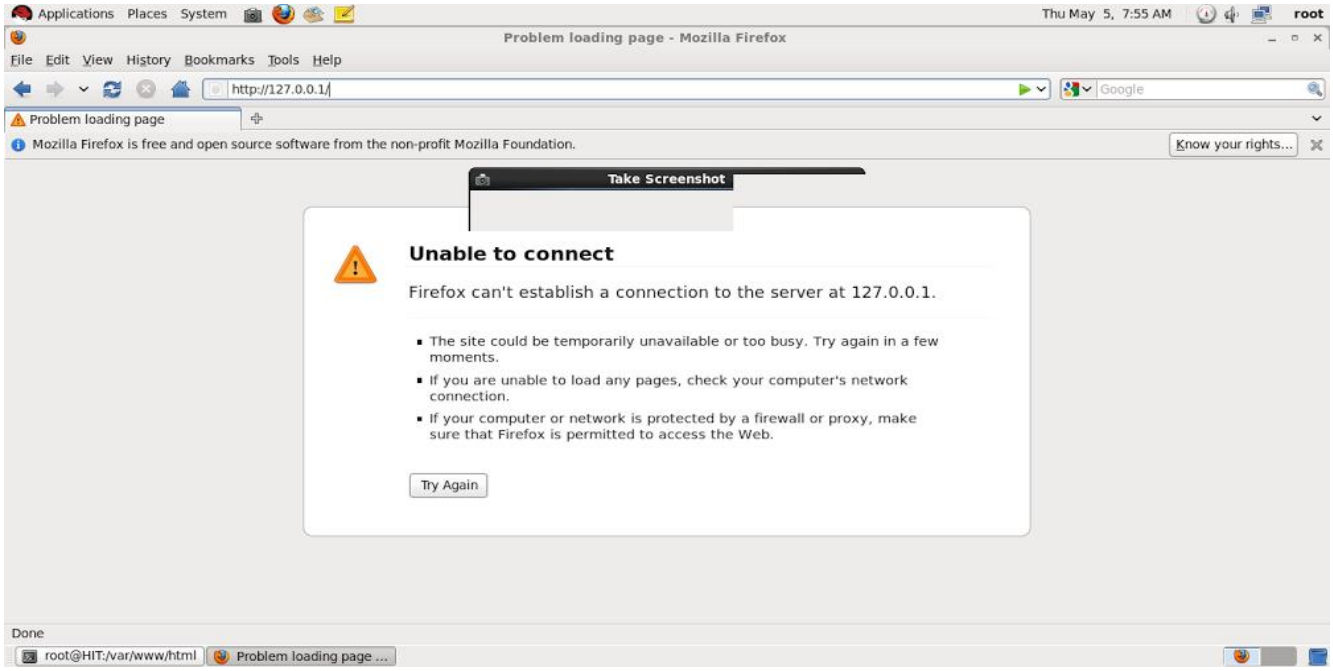
When Apache starts, either during system boot or when invoked after boot using the init script /etc/rc.d/init.d/httpd or the Apache-provided script apachectl, it reads and processes three files, in order: /etc/httpd/conf/httpd.conf, /etc/ httpd/conf/srm.conf, and /etc/httpd/access.conf. All configuration directives can and should be placed in httpd.conf or included from other files specified using the Include directive.Using a single file simplifies maintaining the configuration file. httpd.conf is the primary configuration file.

Now we need to create a file named with .html extension in the DocumentRoot folder */var/www/html*. The DocumentRoot sets the base directory from which all requested documents will be served; document URLs (file names) are interpreted relative to DocumentRoot.

We will be now creating a file "as.html" in /var/www/html/



Now, when we browse to our localhost (httpd://127.0.0.1/), we are unable to connect to our host as the configuration file of httpd is not configured.



We need to specify the page that we created in the configuration file as directory index parameter.

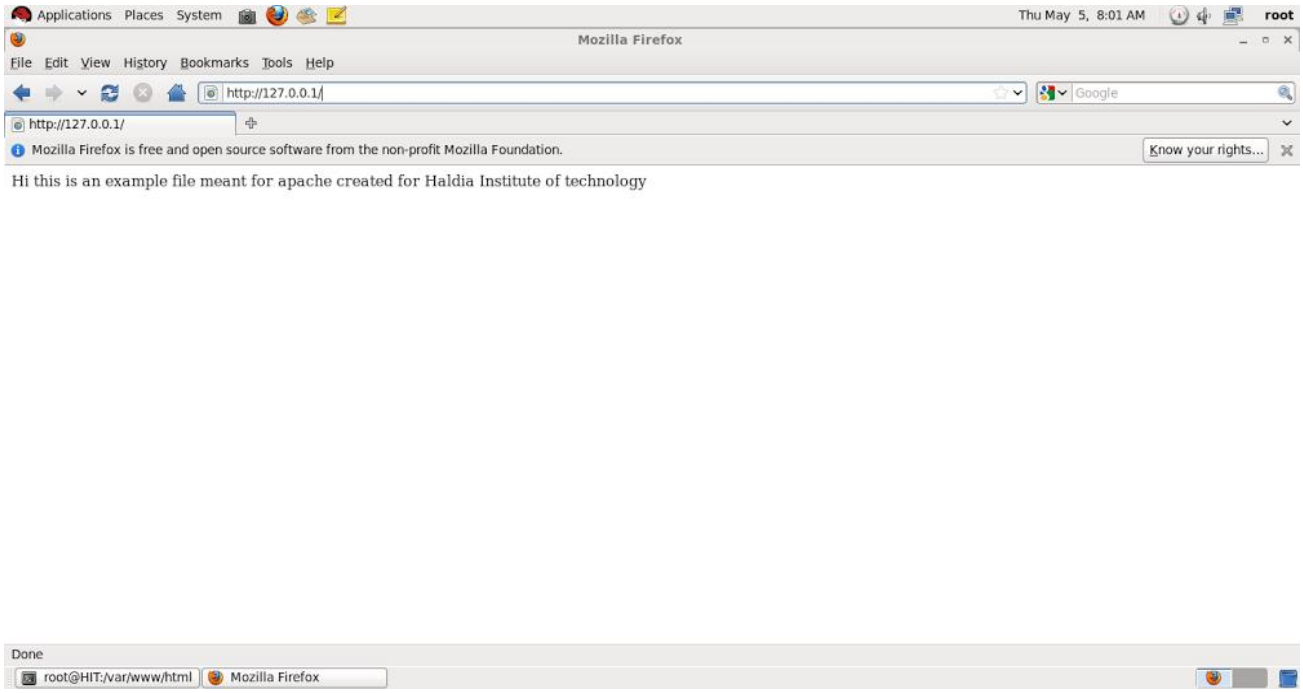
]# vi /etc/httpd/conf/httpd.conf

```
398 # The index.html.var file (a type-map) is used to deliver content-
399 # negotiated documents.  The MultiViews Option can be used for the
400 # same purpose, but it is much slower.
401 #
402 DirectoryIndex           as.html index.html.var
403
404 #
405 # AccessFileName: The name of the file to look for in each directory
406 # for additional configuration directives.  See also the AllowOverride
407 # directive.
```

As we have changed the httpd configuration file, so we need to restart the httpd service.

]# service httpd restart

Now when we browse to our localhost, as.html page opens up.



NOTE: The DocumentRoot /var/www/html directive sets the server's base document directory to /var/www/html, meaning that all URLs are served relative to this directory.

DIRECTORY BROWSING

To demonstrate Directory Browsing in Apache we need to create some files in */var/www/html/* directory. We also need to restore our httpd configuration file as it was.

]# cd /var/www/html/
]# touch 1 2 3 4

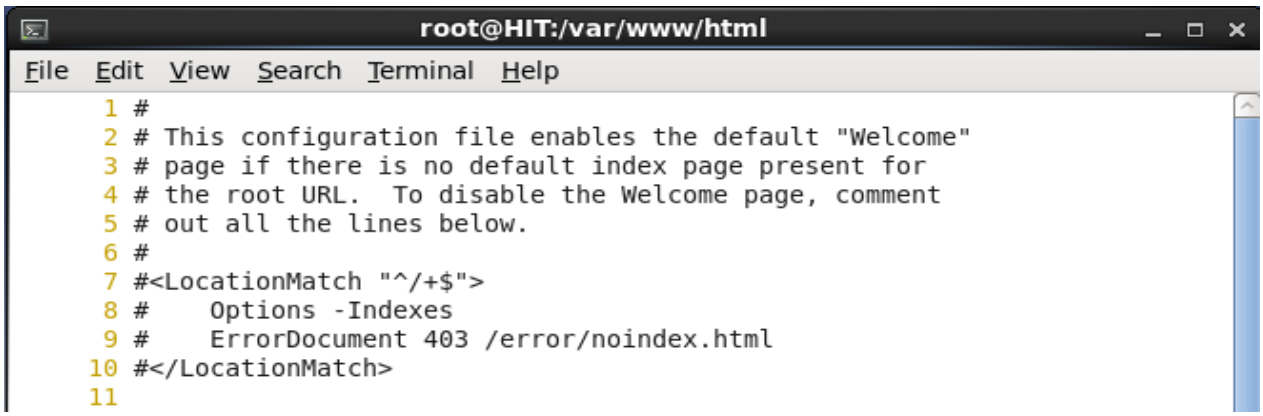
]# vi /etc/httpd/conf/httpd.conf

We remove the entry of as.html from DirectoryIndex parameter.

```
398 # The index.html.var file (a type-map) is used to deliver content-
399 # negotiated documents.  The MultiViews Option can be used for the
400 # same purpose, but it is much slower.
401 #
402 DirectoryIndex index.html index.html.var
403
404 #
405 # AccessFileName: The name of the file to look for in each directory
406 # for additional configuration directives.  See also the AllowOverride
407 # directive.
```

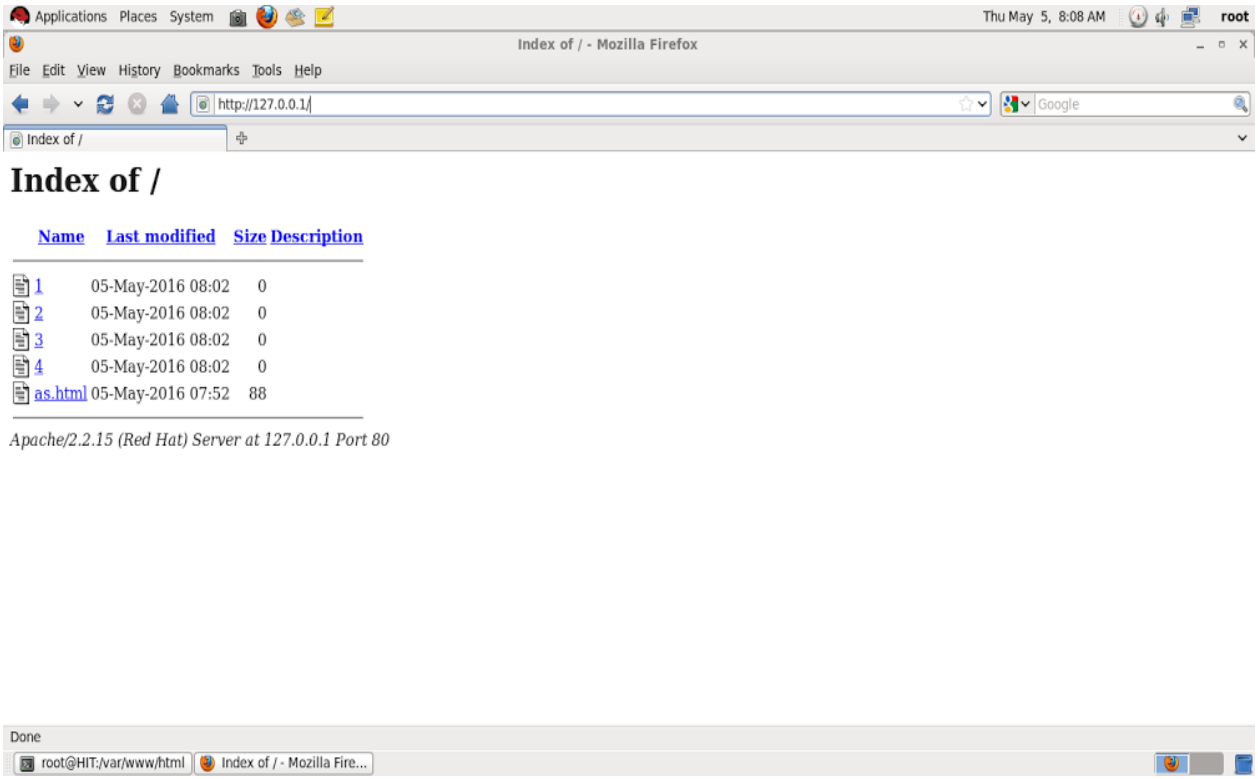
At this stage when we run our localhost, the default page of redhat is loaded. So, in order to avoid this default loading of redhat page and browse our directory /var/www/html, we need to modify the /etc/httpd/conf.d/welcome.conf file, we comment all the executable lines by adding #.

]# vi /etc/httpd/conf.d/welcome.conf



Then we restart the httpd service and run our localhost to notice that our directory browsing is successful.

]# service httpd restart



VIRTUAL HOSTING

Inter Website Hosting

Virtual Hosting is a method through which we can restrict our clients from browsing through specified sites and virtually hosting their request to the server specified diredtory index file.
The step-by-step process of configuring our Apache server so that it can virtual host particular urls like www.facebook.com, www.gmail.com,etc,

]# mkdir /var/www/html/abc
]# cat > /var/www/html/abc/abc.html

```
[root@HIT Desktop]# mkdir /var/www/html/abc
[root@HIT Desktop]# cat > /var/www/html/abc/abc.html
This is abc.html page
<b>you have been virtually hosted<b><br>
<i><h1>welcome to haldia institute of technology</h1></i>

^Z
[1]+  Stopped                 cat > /var/www/html/abc/abc.html
[root@HIT Desktop]#
```

]# mkdir /var/www/html/xyz
]# cat > /var/www/html/xyz/xyz.html
This is xyz.html page

We need to specify the html pages that we created in abc and xyz directory in the httpd configuration file.

]# vi /etc/httpd/conf/httpd.conf

```
400 # same purpose, but it is much slower.
401 #
402 DirectoryIndex as.html abc.html index.html
403
404 #
405 # AccessFileName: The name of the file to look for in each directory

988 # Use name-based virtual hosting.
989 #
990 NameVirtualHost *192.168.213.134:80
991 #
992 # NOTE: NameVirtualHost cannot be used without a port specifier

1009 #</VirtualHost>
1010 <VirtualHost 192.168.213.134:80>
1011 DocumentRoot /var/www/html
1012 ServerName www.yahoo.com
1013 </VirtualHost>
1014 <VirtualHost 192.168.213.134:80>
1015 DocumentRoot /var/www/html/abc
1016 ServerName www.facebook.com
1017 </VirtualHost>
```

Ip of the server.

We need to add these line at eof.
When we open yahoo it redirects to as.html.
facebook → abc.html

In order to resolve DNS name we update the info in /etc/hosts file as well.
 # vi /etc/hosts

```
1 127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
2 ::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
3 192.168.213.134 www.yahoo.com
4 192.168.213.134 www.facebook.com
```

We add these lines at the eof

Once we update the hosts file we then restart our services of hhtpd.
Now our domain are virtually hosted and if a user tries to navigate to www.facebook.com he or she is hosted to the specified abc directory.

WEBSITE AUTHENTICATION

Once we have virtually hosted some domains, we can further add security to those urls by authenticating those websites using Directory tag.
What we do here is we restrict our clients from browsing particular websites by generating a username and a password for that url. For example if a software engineer in a company tries to navigate through www..facebook.com then before loading that page he or she is asked for a username and a password. Om the other hand, the manager can access www.facebook.com as he has that username and password.

So, using website authentication we can restrict particular clients from navigating to a website as well as allow some clients through username and password to access the same website.

Now, we will configure our Apache server to implement website authentication using the following steps.
]# vi /etc/httpd/conf/htppd.conf

```
1010 <VirtualHost 192.168.213.134:80>
1011 DocumentRoot /var/www/html
1012 ServerName www.yahoo.com
1013 </VirtualHost>
1014
1015 <VirtualHost 192.168.213.134:80>
1016 DocumentRoot /var/www/html/abc
1017         <Directory /var/www/html/abc>
1018                 AuthName        "website authentication"
1019                 AuthType        basic
1020                 AuthUserFile    /var/www/html/abc/kk
1021                 Require         valid-user
1022         </Directory>
1023 ServerName www.facebook.com
1024 </VirtualHost>
```

We add these line in <VirtualHost>

kk is the password authentication file, where username and password is stored

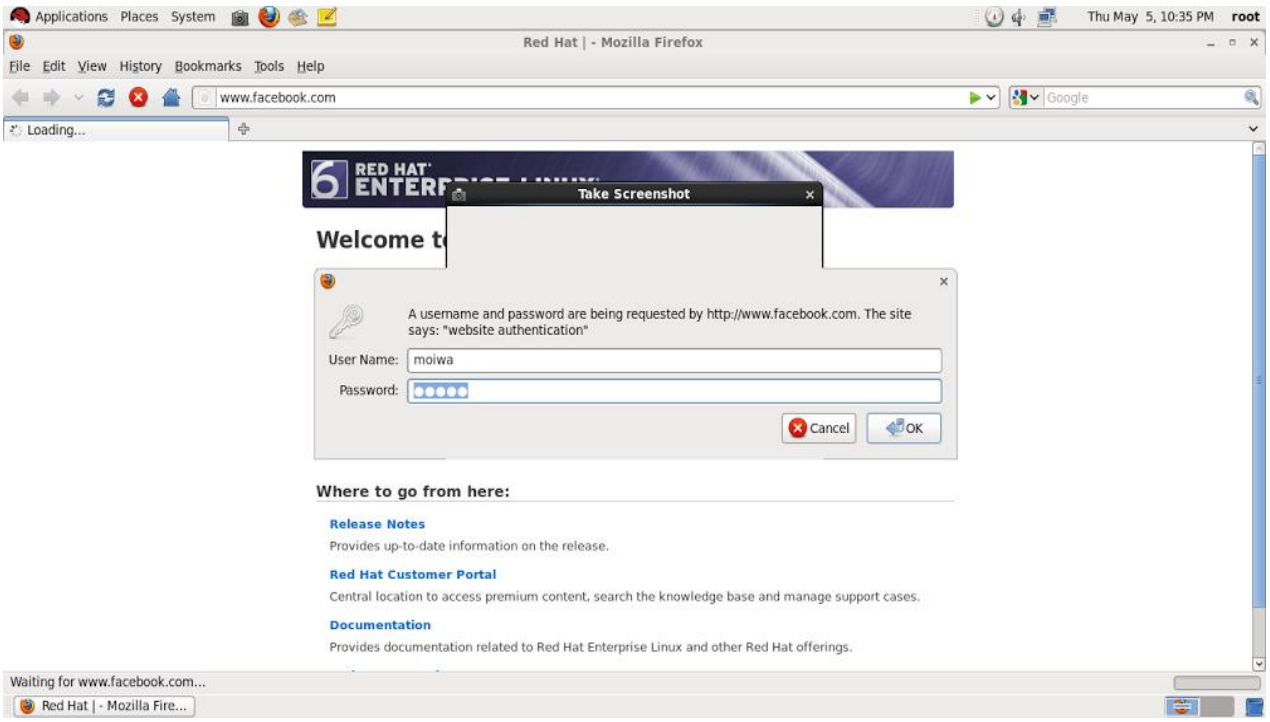
Since our httpd configuration has been modified and updated to handle website authentication, we need to set the username and password fot the authentication process using htpasswd with create and modify (-cm) parameters.

```
[root@HIT Desktop]# htpasswd -cm /var/www/html/abc/kk moiwa
New password:
Re-type new password:
Adding password for user moiwa
[root@HIT Desktop]# service httpd restart
```

NOTE: The password for user "moiwa" is moiwa.

We then restart the service of httpd.
]# service httpd restart

Now, when we navigate to a password protected url, we are asked for the required credentials.



UNDERSTANDING NIS

The Network Information Service distributes information that needs to be shared throughout a Linux network to all machines on the network. NIS was originally developed by ***Sun Microsystems*** and known as *Yellow Pages (YP*), so many commands begin with the letters yp, such as ypserv, ypbind, and yppasswd. Unfortunately for Sun, the phrase "Yellow Pages" was a registered trademark of British Telecom in the United Kingdom, so Sun changed the name of their Yellow Pages services to Network Information Service.

The information most commonly distributed across a network using NIS consists of *user database and authentication information*, such as /etc/passwd and /etc/group.

If, for example, a user's password entry is shared by all login hosts via the NIS password database, that user is able to log in on all login hosts on the network, all hosts, that is, that are running the NIS client programs.

However, user authentication databases are not the only use for NIS — any information that needs to be distributed across a network and that can or should be centrally administered is a viable candidate for sharing via NIS. For example, we can use NIS to distribute a company telephone directory or a listing of accounting codes.

NIS vs DNS

An NIS domain is not the same as an Internet or DNS domain. A *DNS domain name* (more specifically, a fully qualified domain name, or FQDN), as we know, is the official name that uniquely identifies a system to the Internet domain name system. An *NIS domain name* refers to a group of systems, typically on a LAN or on only a subnet of a LAN, that use the same NIS maps. NIS domains are typically used as system management tools, a convenient method for organizing groups of machines that need to access the information shared across a network using a set of common NIS maps.

NIS databases are stored in *DBM format*, a binary file format based on simple ASCII text files. For example, the files /etc/passwd and /etc/group can be converted directly to DBM format using an ASCII-to-DBM conversion program named ***makedbm***.

NIS SERVER & NIS CLIENT

Each NIS domain must have at least one system that functions as an NIS server for that domain. An *NIS server* is a centrally-administered repository for information that is shared across the network using NIS. *NIS clients* are programs that use NIS to query designated servers for information that is stored in the servers' databases, which are known as *maps*.

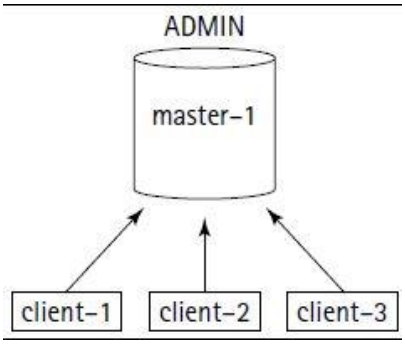
NIS servers can be further subdivided into master and slave servers. A *master* server maintains the authoritative copies of the NIS maps. A *slave* server maintains copies of the NIS databases, which it receives from the master NIS server whenever changes are made to the databases stored on the master. In NIS configurations that use slave servers, the slaves receive copies of the DBM databases, not the ASCII source files. The yppush program notifies slave servers of changes to the NIS maps, and then the slaves automatically retrieve the updated maps in order to synchronize their databases with the master. NIS clients do not need to do this because they communicate with their designated server(s) to obtain current information. The rationale for slave servers is to provide redundancy — if the master server is unavailable for some reason, slave servers function as backup servers until the master is again available.

NIS TOPOLOGIES

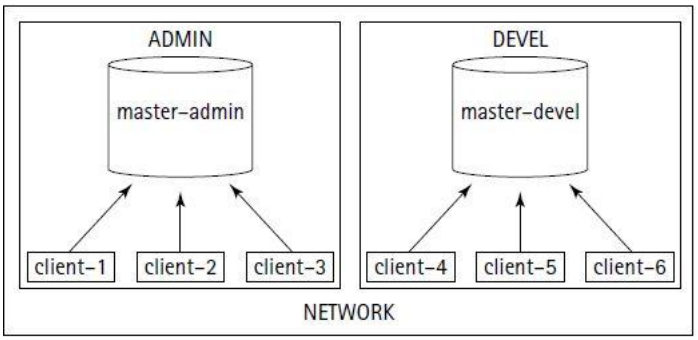
Four NIS topologies are commonly used:

1. A single domain with a master server, no slave servers, and one or more clients.
2. A single domain with a master server, one or more slave servers, and one or more clients.
3. Multiple domains, each with its own master server, no slave servers, and one or more clients.
4. Multiple domains, each with its own master server, one or more slave servers, and one or more clients.

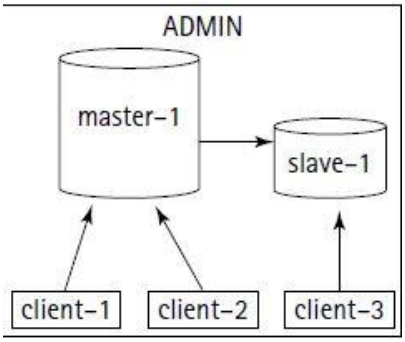
The single domain configurations are the most widely used in most situations.



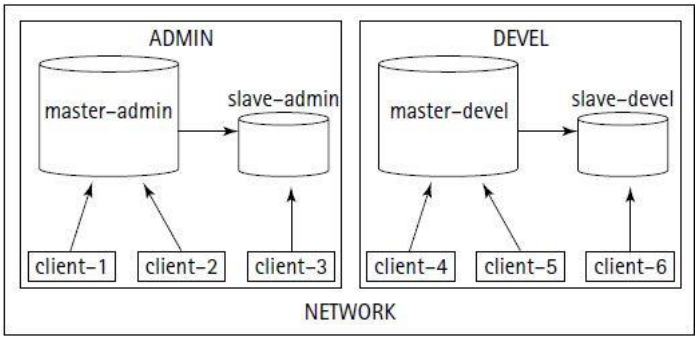
the single server, master-1, responds to all queries from NIS clients (client-1, client-2, and client-3) and is the sole source of information for the domain, named admin.



two domains, admin and devel, each with its own master server, master admin and master-devel. Clients in the admin domain communicate only with the master-admin server, and clients in the devel domain communicate only with devel.



In this case, client-1 and client-2 continue to query the master server, but client-3 communicates with the slave server when performing NIS queries.



each domain has a slave server, slave-admin and slave-devel, and some of the clients in each domain communicate with the slave servers rather than with the master.

Configuring the NIS Server

NIS server configuration involves the following steps:
1. Setting the NIS domain name.
2. Configuring and starting the server daemon, ypbind.
3. Initializing the NIS maps.
4. Starting the NIS password daemon.
5. Starting the NIS transfer daemon if you use slave servers.
6. Modifying the startup process to start the NIS daemons when the system reboots.

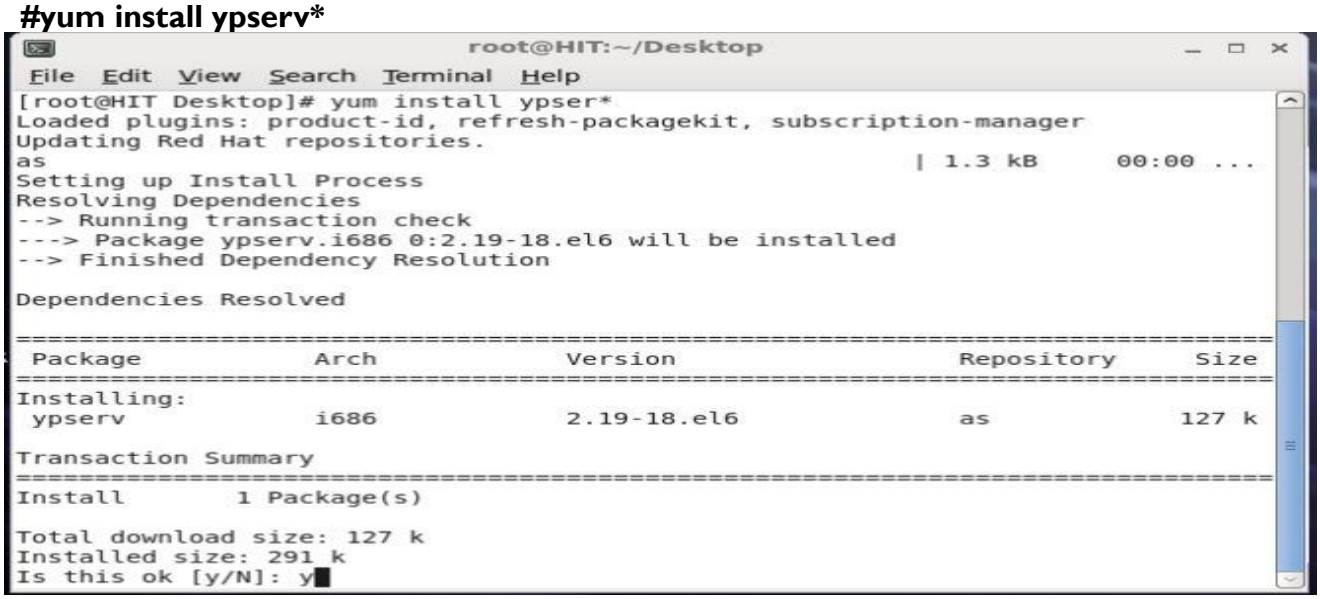
Setting the NIS domain name

The initial step in configuring an NIS client is to set the NIS domain name. When first configuring the NIS server, the quickest way to set an NIS domain name is to use the nisdomainname command:
nisdomainname *nisdomain*
Replace *nisdomain* with the name of your NIS domain. Next, reissue the nisdomainname command with no arguments to confirm that the NIS domain name was successfully set.
#nisdomainname haldia.com
#domainname haldia.com

We also need to know the hostname of our system, which we can get using hostname command with no arguments. In order to change our hostname temporiarly we can paas the name as an argument and it sets the hostname.
#hostname

```
[root@HIT Desktop]# nisdomainname haldia.com
[root@HIT Desktop]# domainname haldia.com
[root@HIT Desktop]# cd /usr/lib/yp
[root@HIT yp]# hostname hitit
```

We need to install the required packages for NIS which is ypserv.
#yum install ypserv*



PORTMAPPER

Before starting the server, make sure the portmapper daemon, portmap, is running. NIS requires the portmapper because NIS uses remote procedure calls (RPC). To see if the portmapper is running, you can use the portmapper's initialization script, /etc/rc.d/init.d/portmap, or the rpcinfo command. If the portmapper is not running, you can easily start it. Using the initialization script, the command to execute and its output, are:

```
# /etc/rc.d/init.d/portmap status
portmap (pid 559) is running...
```

The output indicates the process ID (PID) of the portmapper. On the other hand, if the portmapper is *not* running, the output of the command looks like the following:

```
# /etc/rc.d/init.d/portmap status
portmap is stopped
```

To start the portmapper, execute the following command:

```
# /etc/rc.d/init.d/portmap start
Starting portmapper:                    [ OK ]
```

You can also use the rpcinfo command shown next to see if the portmapper is running:

```
# rpcinfo -p localhost
program    vers    proto    port
100000     2       tcp      111 portmapper
100000     2       udp      111 portmapper
```

Again, if you do not see output indicating that the portmapper is running, use the initialization script shown previously to start the portmapper. Once you have the portmapper started, start the NIS server using the command:

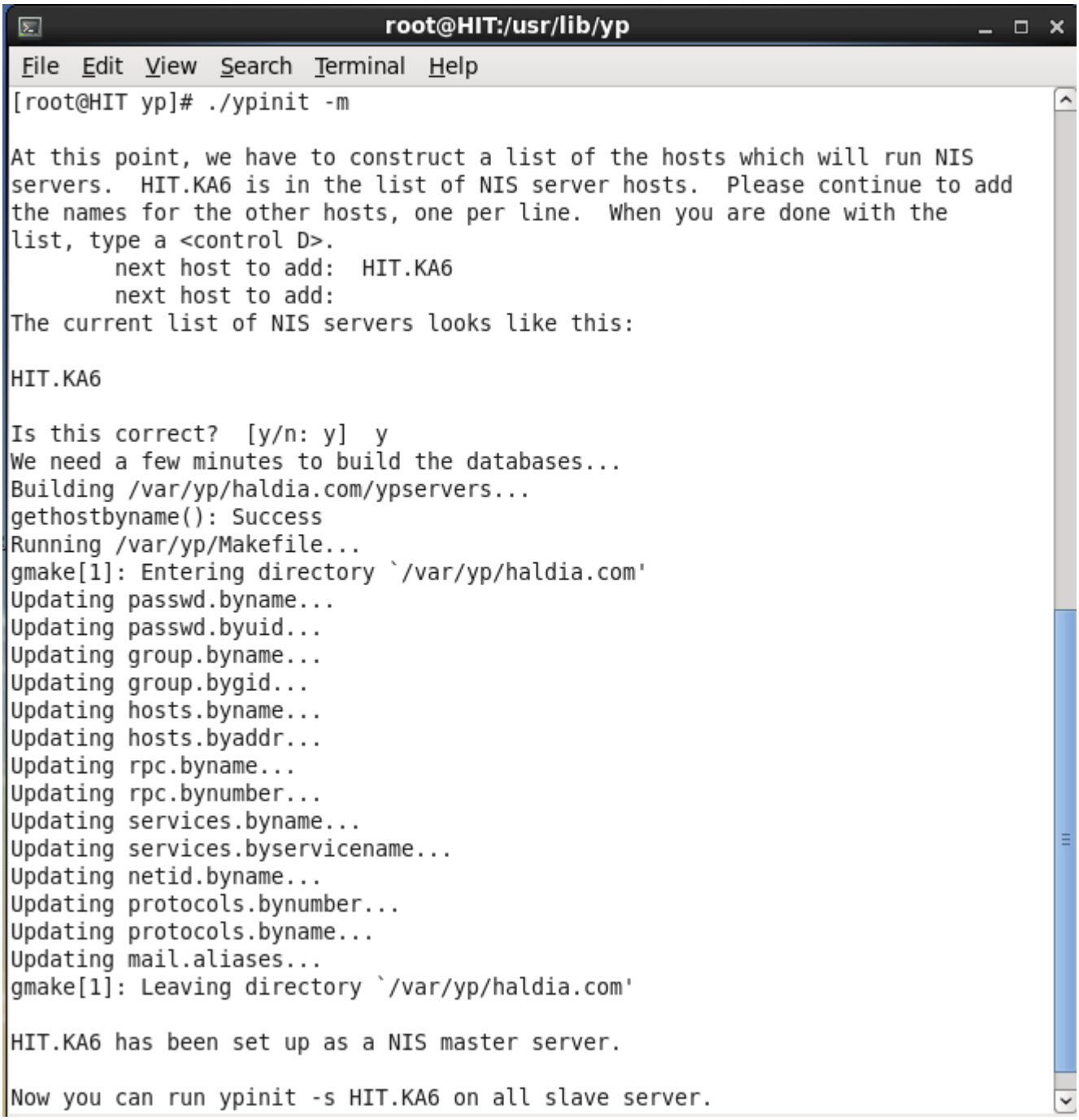
```
# /etc/rc.d/init.d/ypserv start
Starting YP server services:            [ OK ]
```

Next, use the following rpcinfo invocation to make sure that the server is running:

```
# rpcinfo -u localhost ypserv
program 100004 version 1 ready and waiting
program 100004 version 2 ready and waiting
```

Initializing the NIS maps

Once the NIS server is running, you have to create something for it to serve, which means you need to create the NIS databases on the machine acting as the NIS server. The command for doing so is ypinit.
ypinit builds a complete set of NIS maps for your system and places them in a subdirectory of /var/yp named after the NIS domain.
To create the NIS databases, execute the command
#/usr/lib/yp/ypinit –m
This command uses the -m option to indicate that it is creating maps for the master server.
Your NIS server is now up and running.



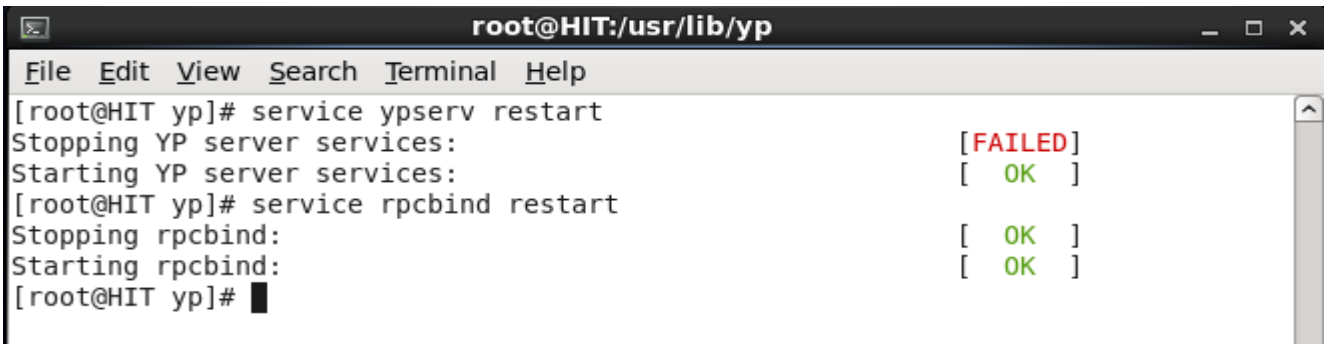
If you need to update a map, run make in the /var/yp directory on the NIS master. This command updates a map if the source file is newer, and propagates the new map out to the slave servers, if present.

#make –C /var/yp

Then we restart our ypserv and rpcbind services.

#service ypserv restart

#service rpcbind restart



One last step on the server is to share the home directory with the clients. The home directory is shared through /etc/exports file. It is a blank file and we add the directories with their permissions on each separate lines.

#vi /etc/exports

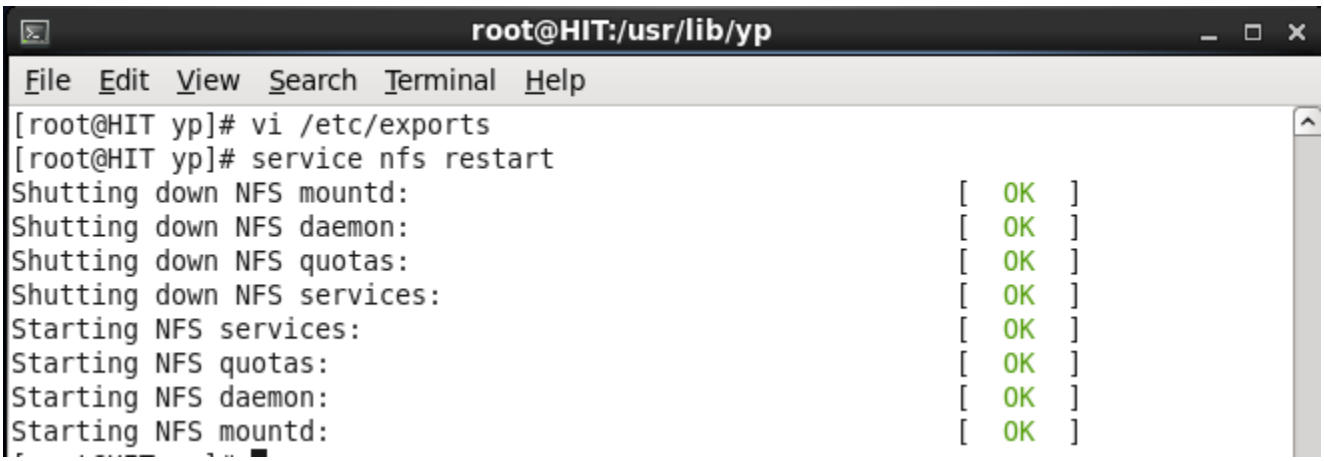
1| /home *(rw)

Sharing the /home directory

* Represents that any ip client may access and rw declaers that the client has read and write permissions.

We then save this file and restart the services of nfs.

#service nfs restart



Configuring the NIS Client

After you have successfully configured at least one master NIS server, you are ready to configure one or more NIS clients. The general procedure for setting up an NIS client involves the following steps:

1. Set the NIS domain name.
2. Configure and start the NIS client daemon.
3. Test the client daemon.
4. Configure the client's startup files to use NIS.
5. Reboot the client.

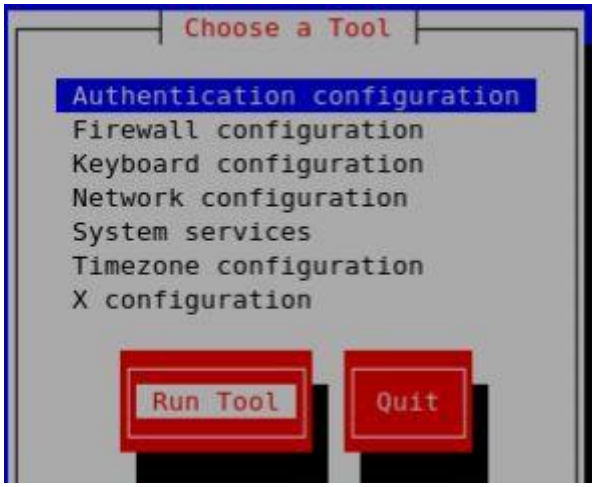
Setting the NIS domain name

#nisdomainame haldia.com
#domainname haldia.com

We need to install the packages ypbind on the client side though ypbind is preinstalled in the operating systems.
#yum install ypbind*

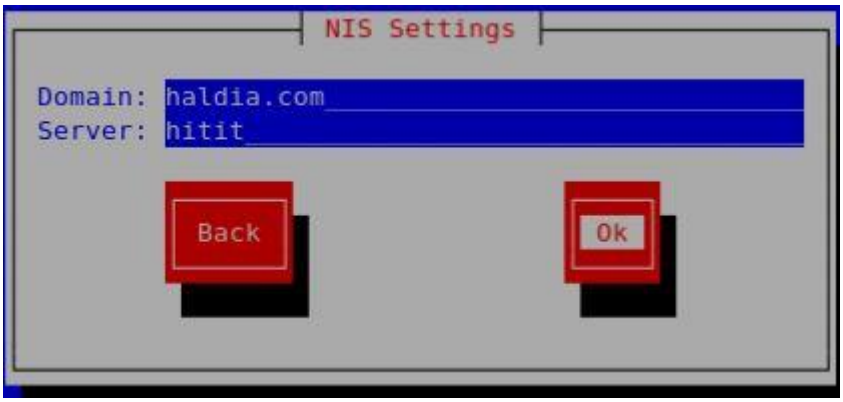
The NIS client daemon, ypbind uses a configuration file named /etc/yp.conf that specifies which NIS servers clients should use and how to locate them, a process known as *binding* the client to the server.
To set the client's NIS servers, you can edit /etc/yp.conf directly or use the authconfig tool, a text mode program for configuring how your system performs user authentication.

#setup



Highlight Authentication configuration and select Run Tool and press enter.

Star mark Use NIS and select next and press enter.



In the NIS settings, you need to specify the domainanme and Server's hostname.
In our project we have used HIT.KA6 as our server.

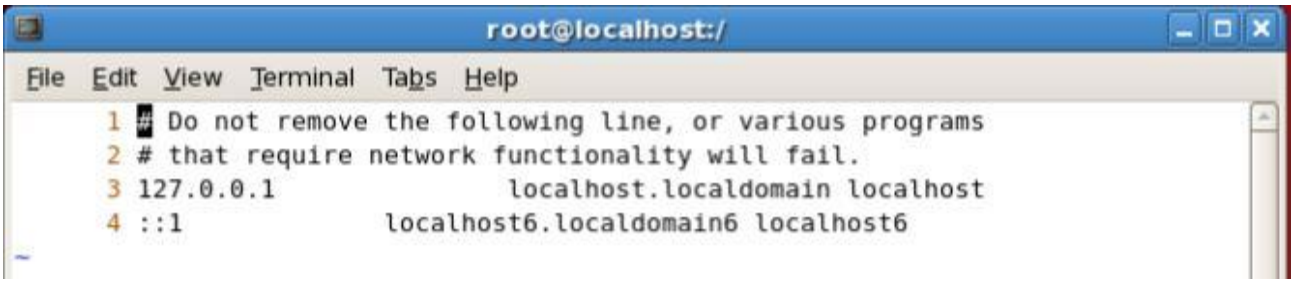
After Setting up the NIS settings we need to restart the ypbind service.
#service ypbind restart

NIS client programs, like the NIS servers, require RPC to function properly, so make sure the portmapper is running before starting the client daemon, ypbind. To start the client daemon, execute the following command, which invokes the ypbind initialization script:
 # /etc/rc.d/init.d/ypbind start
Binding to the NIS domain: [OK]
Listening for an NIS domain server.

After starting the NIS client daemon, use the command rpcinfo -u localhost ypbind to confirm that ypbind was able to register its service with the portmapper. The output should resemble the following:
 # rpcinfo –u localhost ypbind
program 100007 version 1 ready and waiting
program 100007 version 2 ready and waiting

Now we run the yptest command to check our configuration is properly set up or not.
#yptest

Now, edit /etc/hosts so that it uses NIS for hostname lookups.
#vi /etc/hosts



At line number 3 we mention the ip of the server and the servers hostname.
So the /etc/hosts files now looks like this at line number 3.

3| 192.168.213.134 HIT.KA6

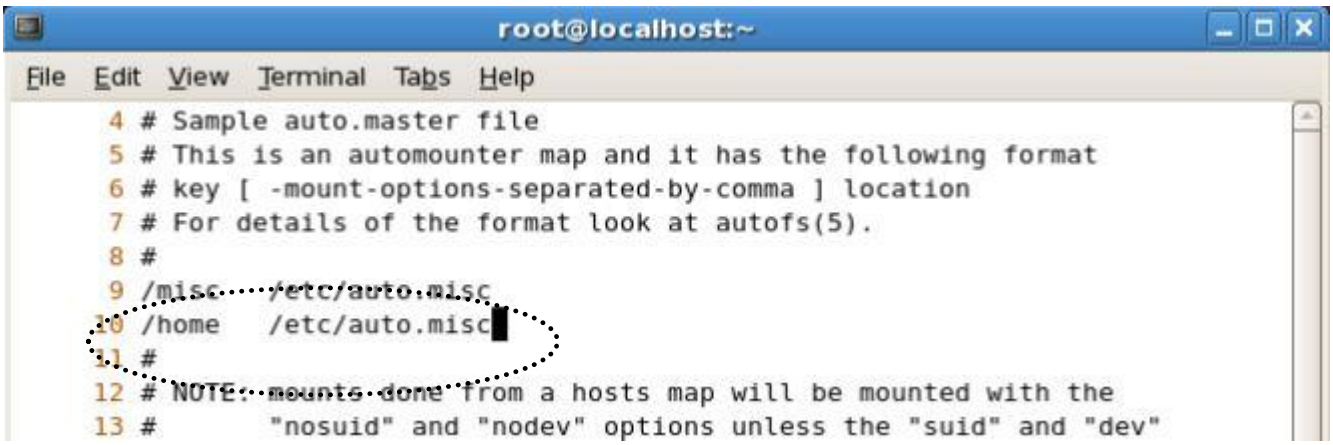
We then restart the network services.
#service network restart

ENABLING AUTOMOUNTING

When a linux administrator has configured NIS server and client then if the client log in to a user created on the server, the home directory of the user is not automounted and the client is provided with a bash shell with no user home directory.
So in order to avoid this situation we need to use autofs services on client.

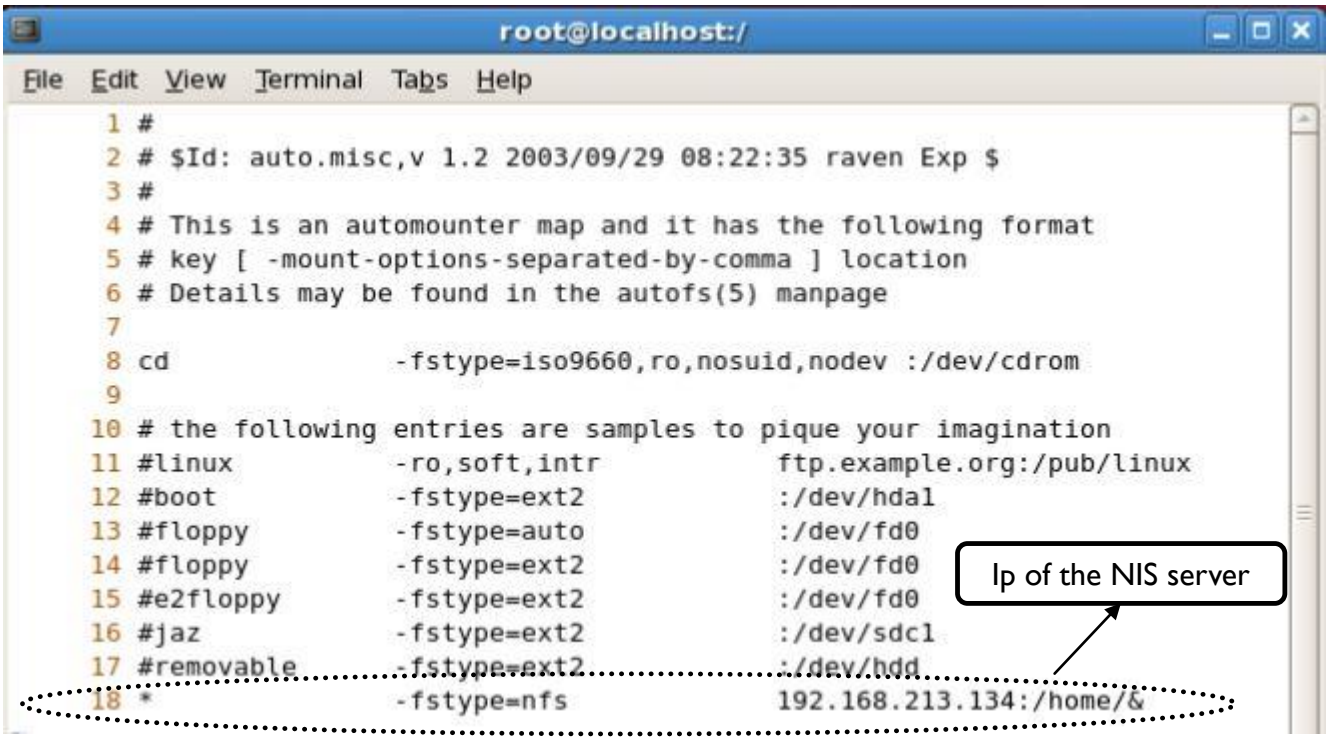
To install autofs we use yum. Though autofs comes preinstalled in linux operating systems.
#yum install autofs*

We then configure /etc/auto.master file where we indicate the directory to be automounted.
#vi /etc/auto.master



We then open the /etc/auto.misc file to add the server's ip address and the directories that needed to be automounted.

#vi /etc/auto.misc

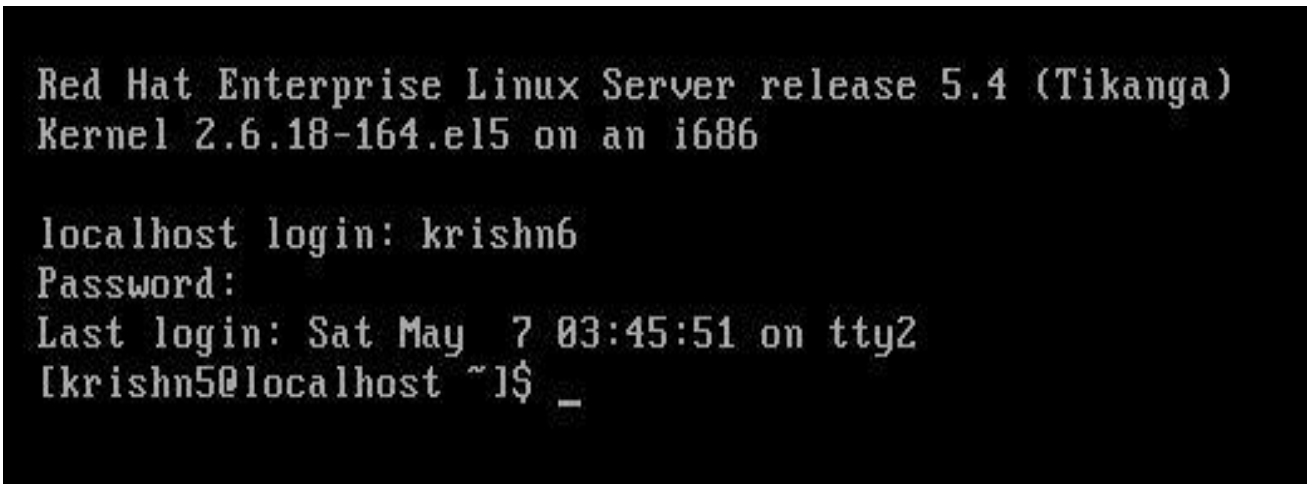


#service autofs stop
#service autofs restart

#chvt 2

Now the client can login to server's user and create and manage files on the server through the client.

In our example a user name krishn6 is logging into her account created on the server through the client.



Understanding OpenLDAP

If we are running a single Red Hat Enterprise Linux Server or even just a few servers, there is no problem in managing configuration files on each individual server. However, if we are responsible for operating dozens of servers, we will need a solution that helps us maintain essential system information from a single location in our network. An LDAP server is such a solution.

- LDAP is an acronym for ***Lightweight Directory Access Protocol (LDAP).***
- Cross Platform support (server and client may be using different OS windows/Linux).
- Port number is 389.
- Client must run lower version of Linux when client and server both are on Linux operating system.

Types of Information in OpenLDAP

OpenLDAP normally includes the following piece of information:

1. Users and groups.
2. Information about systems in the network.

The LDAP Name Schema

One of the typical properties in OpenLDAP is that it is organized in a hierarchical manner. The advantage of this approach is that information can be grouped into containers where it is really needed, and clients can refer to the actual location where this information is stored.
We can easily build an OpenLDAP hierarchy where objects in other locations are easily referred to without storing them on local servers. This makes OpenLDAP a *lightweight Directory*

If, for instance, there is a user *eguser* in the hierarchy *hit.example.com*, the fully distinguished name of this user is referred to as :
cn=eguser, dc=hit, dc=example, dc=com.

Here cn and dc are two different object types.
cn(*common name*) : It is a generic way to refer to *leaf entries*, which are the end objects, such as users and groups, that by themselves cannot contain anything else.
dc(*domain component*) : It is used to refer to one of the container entries in the LDAP hierarchy. If in a setup the LDAP hierarchy is mapped to a DNS hierarchy, typically all the DNS domains are referred to as dc objects.

Configuring an OpenLDAP Server

We will just set up a single instance of an LDAP server that can be used by multiple clients in our network for authentication. To configure an OpenLDAP server, we need to follow these steps:

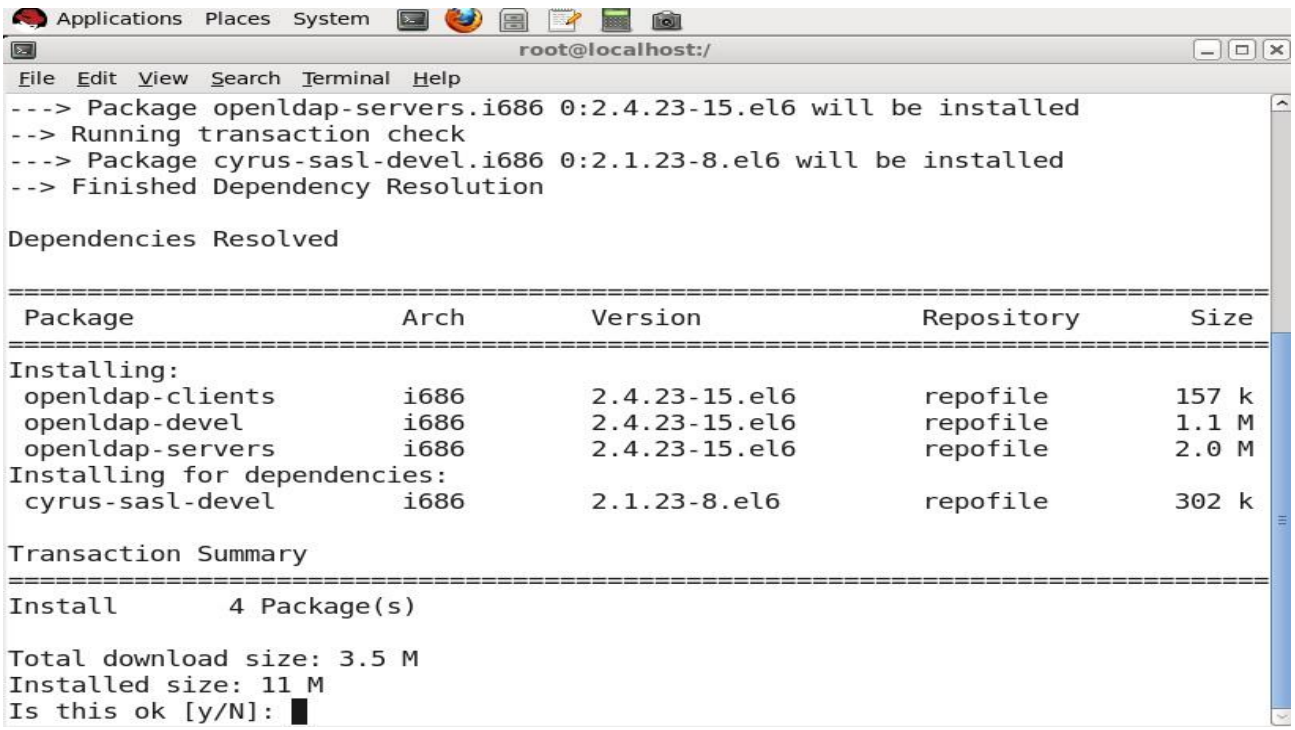
1. Install the OpenLDAP software.
2. Configure the LDAP process to service your needs.
3. Run the OpenLDAP server.

Installing OpenLDAP

To install OpenLDAP, we need to run the following command.

]# yum install openldap*
it will install three packages :

1. openldap-clients
2. openldap-devel
3. openldap-servers



OpenLDAP does not contain a single configuration file. Instead, it uses a configuration directory, which by itself is organized in a way that is typical for LDAP.

The default configuration file is **/etc/openldap/slapd.d/cn=config.ldif**

Configuring 'slapd.conf' file

After installing openldap packages we tried to get the location of slapd.conf file using the following command.

]# locate slapd.conf

Running this command just after the installation might not work as we need to update the database where locate command can search. We do so using the following command and then we tried locating the file.

]# updatedb

```
[root@localhost /]# locate slapd.conf
[root@localhost /]# updatedb
[root@localhost /]# locate slapd.conf
/etc/openldap/slapd.conf
/usr/share/man/man5/slapd.conf.5.gz
/usr/share/openldap-servers/slapd.conf.obsolete
```

Now we need to copy some files in slapd.conf directory.
]# cp /usr/share/openldap-servers/slapd.conf/obsolete /etc/openldap/slapd.conf

The slapd.conf file consists of a series of global configuration options that apply to slapd as a whole (including all backends), followed by zero or more database backend definitions that contain information specific to a backend instance. Global options can be overridden in a backend (for options that appear more than once, the last appearance in the slapd.conf file is used). Blank lines and comment lines beginning with a `#' character are ignored. If a line begins with white space, it is considered a continuation of the previous line.

- The ***suffix*** line names the domain for which the LDAP server provides information and should be changed from.
- The ***rootdn*** entry is the Distinguished Name (DN) for a user who is unrestricted by access controls or administrative limit parameters set for operations on the LDAP directory. The rootdn user can be thought of as the root user for the LDAP directory.
When populating an LDAP directory over a network, change the ***rootpw*** line — replacing the default value with an encrypted password string. To create an encrypted password string, type the following command:

/]# slappasswd

When prompted, type and then re-type a password. The program prints the resulting encrypted password to the shell prompt.
we can set rootpw password as encryption or in plain text as it is.

Now we are going to illustrate the changes in the configuration file:, to do this we need to open the configuration file in an editor.

]# vi /etc/openldap/slapd.conf

Fig :Actual configuration file

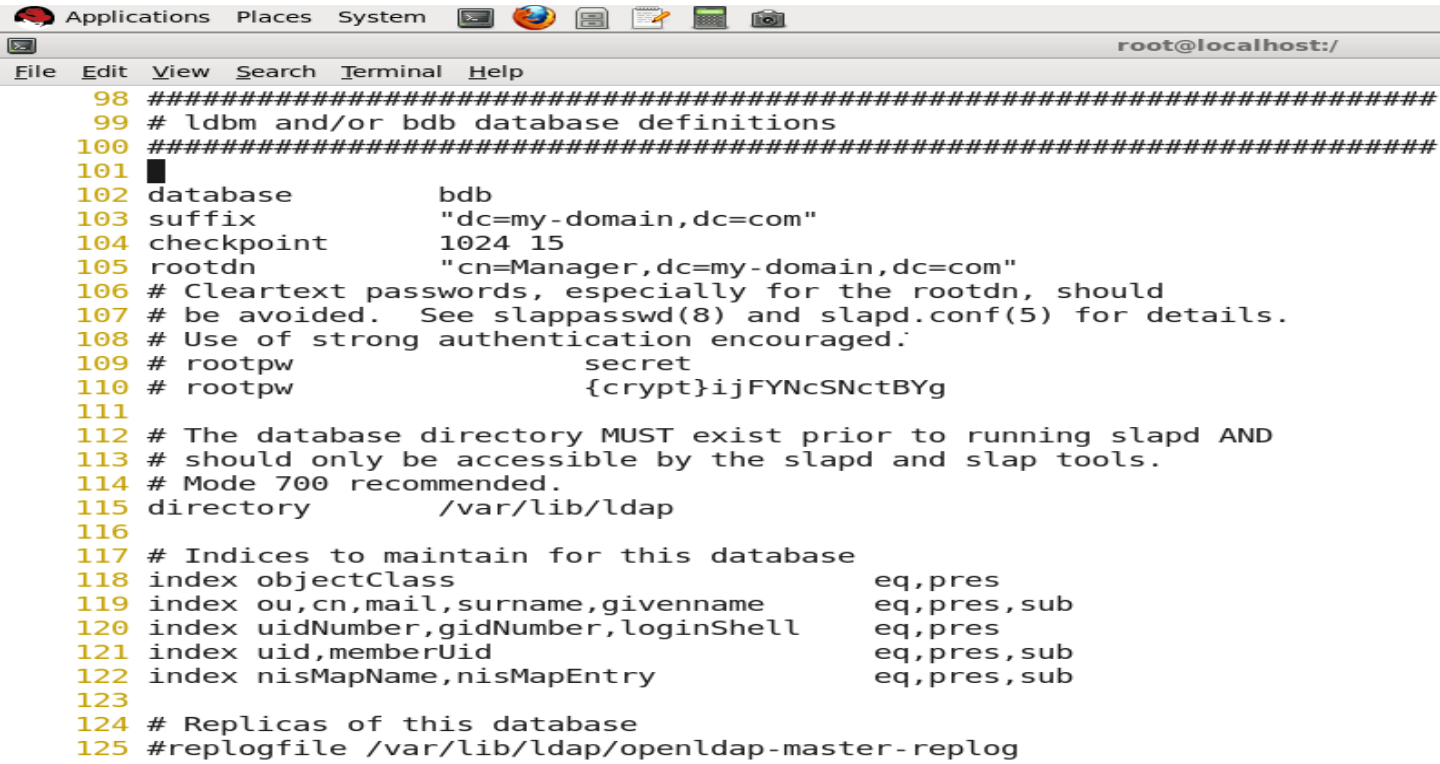
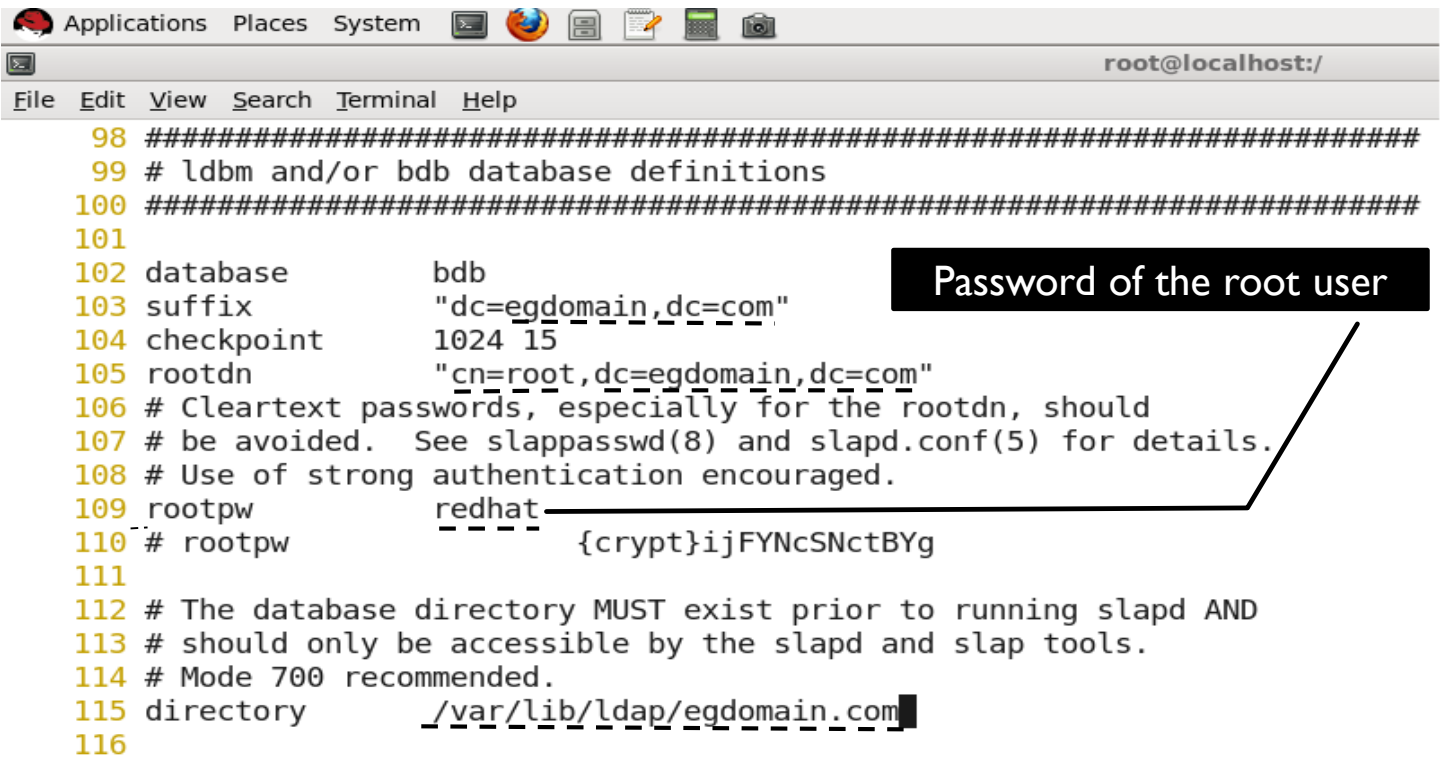


Fig: Modified configuration file.



Configuring olcDatabase ldif file

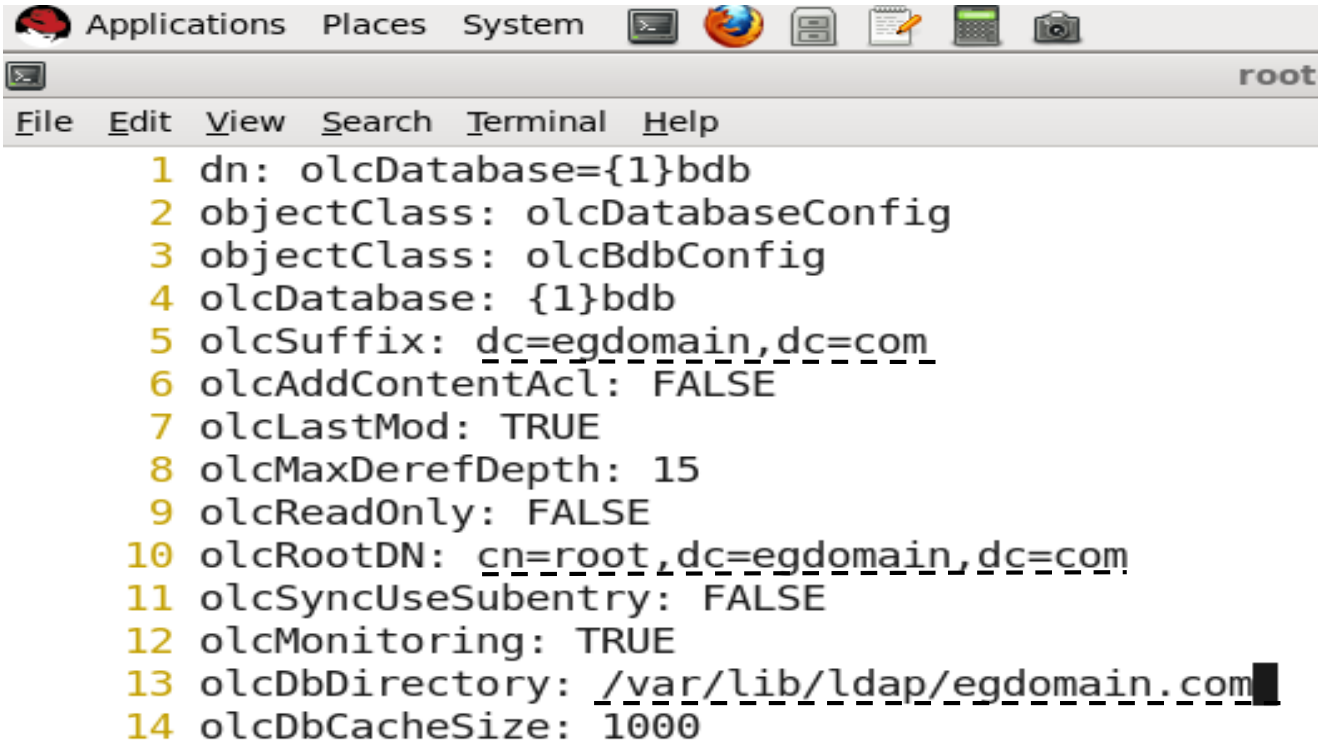
LDIF: These are the standard LDAP input files in which the entries and all the properties that we want to define for these entries are specified. Before adding users and groups that we can use to authenticate on a Linux machine, we first need to create the base structure. In this base structure, we will create the LDAP domain (which often matches the name our organization uses in DNS).

At this point we will create a directory using the following command which will be used while configuring olcDatabase ldif file.
/]# mkdir /var/lib/ldap/egdomain.com

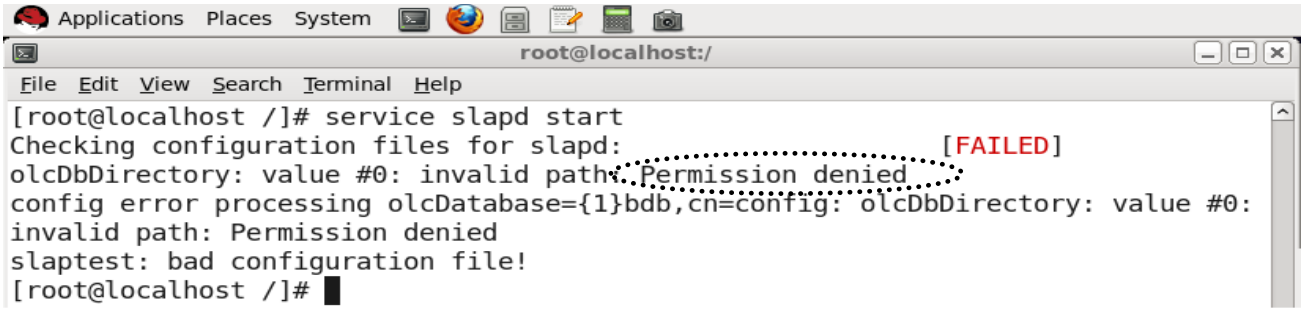
We need to change the olcDatabase 1 file of ldif extension. To create the OLC (cn=config) DIT you can EITHER create a series of LDIF files that describe the configuration and apply them using slapadd OR convert your existing slapd.conf. If the slapd.d directory is not found then slapd looks for slapd.conf.

/]# vi /etc/openldap/slapd.d/cn\=config/olcDatabase\=\{1\}bdb.ldif

Fig : Reflected changes in olcDatabase file.



After creating *'egdomain.com'* directory and updating our olcDatabase file, If we try to start slapd service, it fails as we need to first change the permission of egdomain.com directory.



The owner and group of it must be ldap. We accomplish this using the following set of commands, then starting the slapd service will not produce any permission related error.

]# chown ldap:ldap /var/lib/ldap/egdomain.com

]# service slapd start

We now need to copy DB_CONFIG.example file in the directory we created 'egdomain.com'. DB_CONFIG.example file is a demo file used in egdomain.com.
This file should be placed in the same directory as specified by the directory configuration option in the slapd.conf file.

#]cp /usr/share/openldap-servers/DB_CONFIG.example /var/lib/ldap/egdomain.com/DB_CONFIG

We have to give DB_CONFIG some permissions specific to ldap using following command.
Since this file is stored in our created directory egdomain,.com so we will change the permission of all files contained in that directory.

]# chown ldap:ldap /var/lib/ldap/egdomain.com/*

Populating the OpenLDAP Database

Now that the slapd process has its basic configuration, we are ready to start populating it. To do this, we need to create LDIF files. These are the standard LDAP input files in which the entries and all the properties that we want to define for these entries are specified. Before adding users and groups that we can use to authenticate on a Linux machine, we first need to create the base structure. In this base structure, we will create the LDAP domain (which often matches the name your organization uses in DNS).

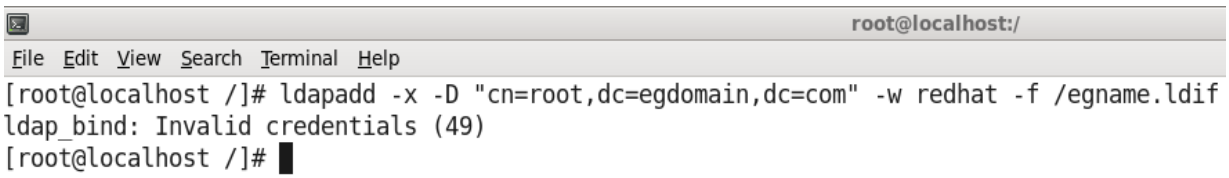
]# vi /egname.ldif

dn : dc=egdomain, dc=com
objectClass: dcObject
objectClass: organization
dc : egdomain
o : egdomain l

In the LDIF fi le, you can see that each section defines one entry. To begin, you see the top-level entry in this database, which is dc=example, dc=com. For each of these entries, you'll need to define which objectClass they are members of. At the end of the three sections that are added here, the actual object is created. This happens in the line dc: example (which creates the dc example)

To add this information to the LDAP Directory, you'll need to use the ldapadd command. With this command, you'll specify the name of the administrative user you've created while setting up the LDAP Directory and the password you've set for this user.

]# ldapadd -x -D "cn=root, dc=egdomain, dc=com" -w redhat -f egname.ldif



Executing ldapadd command at this stage pops up with error showing *Invalid Credentials(49)*
This error occurs due to mismatch between configuration and database file of LDAP.
In order to eliminate this error we need to execute following set of commands.

]# rm –rf /etc/openldap/slapd.d/*
]# slaptest –f /etc/openldap/slapd.conf -F /etc/openldap/slapd.d/
]# chmod –R 777 /etc/openldap/slapd.d/
]# service slapd restart

Now we can run ldapadd command successfully.

ldapadd

Now we need to install a package called migrationtools which are used to convert configuration files to LDIF format, making it compatible with the LDAP server.

]# yum install migrationtools*

```
================================================================================
 Package                Arch           Version           Repository        Size
================================================================================
Installing:
 migrationtools         noarch         47-7.el6          repofile          24 k

Transaction Summary
================================================================================
Install       1 Package(s)

Total download size: 24 k
Installed size: 104 k
Is this ok [y/N]:
```

We need some users which will be used later as client to login to server, so in next steps we are going to add a user named 'eguser' on our server.

]# useradd eguser
]# passwd eguser

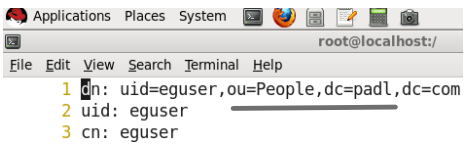
All users information are stored in /etc/passwd file, we will use this file and grep information for our newly created user.

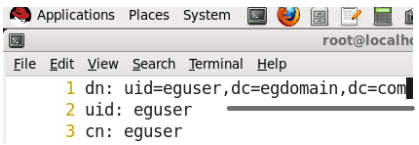
]# grep eguser /etc/passwd > /grepuser
]# cat /grepuser
eguser:x:501:501::/home/eguser:/bin/bash

Now we migrate user using migrationtool.
]# /usr/share/migrationtools/migrate_passwd.pl /grepuser > /grepuser1

we then open this grepuser1 file in an editor and configure the file to add our user.

]# vi /grepuser1

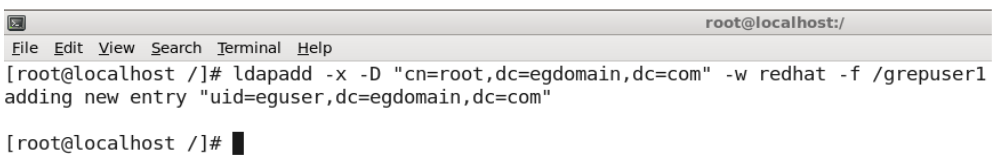




We only change the dc=egdomain field of line 1 and leave everything as it was.

Now we will add users using ldapadd as these users only can login from the client computer.

]# ldapadd –x –D "cn=root, dc=egdomain, dc=com" –w redhat –f /grepuser1



Sharing the home directory of the user

Since our user's home directory is present on the server machine, so in order to make it available on the client machine we need to share the user's home directory on the local network. We share any file or directory by adding the file name or path in */etc/exports* file.

]# vi /etc/exports
/home *(rw)

As a last step to make our LDAP server running is that we start nfs service on the server system using following command.

]# service nfs restart

Configuring a client

Once we our server was set up, we tried to login as a client using a different lower version of Linux. In following example we used RHEL5 as a client.

We need to enable LDAP support on our client using system administration in the GUI (Graphical User Interface) mode.

Click on System --> Administration → Authentication

Click on configure LDAP under User Information and fill in the following details.

LDAP Search Base DN : dc=egdomain, dc=com
LDAP Server : ldap://192.168.233.129/

192.168.233.129 is the ip of server computer. We can get the ip of a linux system by ***ipconfig*** command.

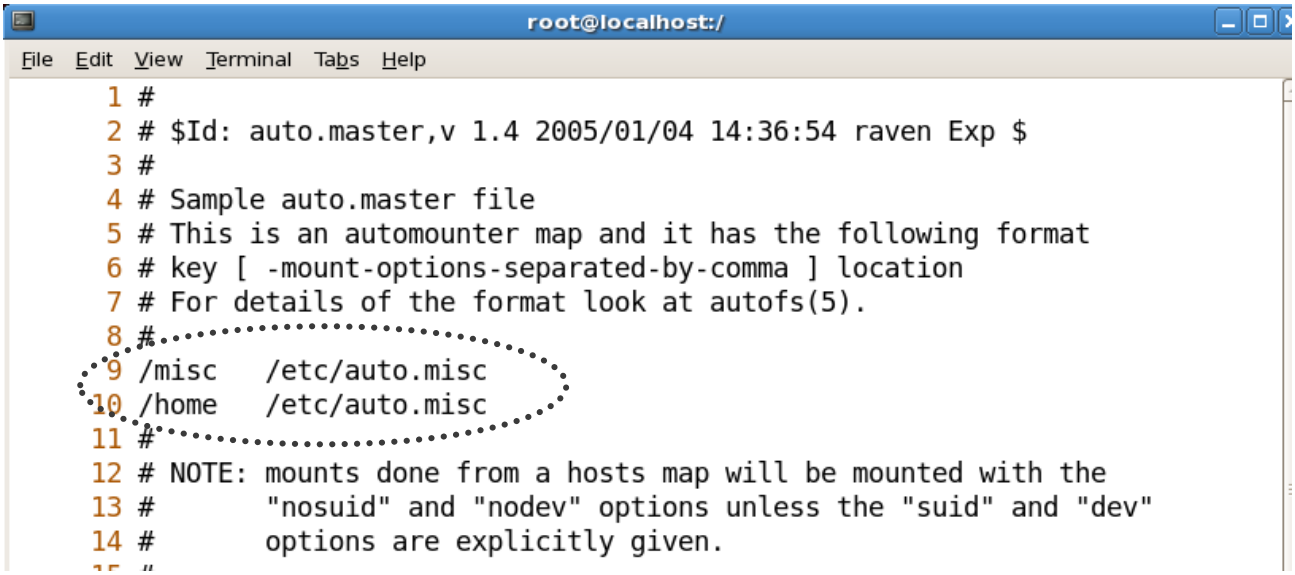
Click OK, and follow the same for *Authentication* tab as well.

Master map configuration file

Autofs consults the master map configuration file */etc/auto.master* to determine which mount points are defined. It then starts an automount process with the appropriate parameters for each mount point. Each line in the master map defines a mount point and a separate map file that defines the file systems to be mounted under this mount point. For example, the /etc/auto.misc file might define mount points in the /misc directory; this relationship would be defined in the /etc/auto.master file.
Each entry in auto.master has three fields. The first field is the mount point. The second field is the location of the map file, and the third field is optional. The third field can contain information such as a timeout value.

]# yum install autofs*

]# vi /etc/auto.master



/]# vi /etc/auto.misc

At the end of file add a line

```
*                -fstype=nfs              192.168.233.129:/home/&
16 #jaz          -fstype=ext2             :/dev/sdc1
17 #removable    -fstype=ext2             :/dev/hdd
18 *              -fstype=nfs              192.168.233.129:/home/&
```

]# service autofs restart

Switch to a different terminal using chvt command or pressing Alt+F2 key combination.
]# chvt2

Logging in as a client user on LDAP Server.

```
Red Hat Enterprise Linux Server release 5.4 (Tikanga)
Kernel 2.6.18-164.el5 on an i686

localhost login: eguser
Password:
FS-Cache: Loaded
id: cannot find name for group ID 501
[eguser@localhost ~]$ who am i
eguser   tty2         2015-07-06 10:59
```

[root@localhost /]# slapcat

On the server side, we can list all the user logged in or have logged previously on the server using this command.

CONCLUSION

Through this project we learned how to configure an OpenLDAP Server on our Red Hat Enterprise Linux Server. First we learned how to set up OpenLDAP and how to use the slapd process itself. After that, we learned how to configure the OpenLDAP server as a directory that can be used on our network for user authentication. We also used our server by logging in as a client on a different version of Red Hat Enterprise Linux 5.

Throughout, the main objective of our project was to see how configuring OpenLDAP Server and services can help in any organisation where an administrator needs to keep a check on what is the progress going on any client computer.

This is very useful when in an organisation, for example say a Call Centre company, where there are a number of employees working in shifts. The employees get the freedom of choosing whichever available computer they can work on and the project or task they have been assigned is hidden from other employees in completion as whatever progress one employee does is saved not on the client's system but on the server.

This also gives the employee an advantage of logging in from any system and whatever the work being done by them previously can be accessed in a very convenient manner, thus removing the burden of carrying their work in an external drive each time they logout.

REFERENCES

1. Book referred –
    - Red Hat Enterprise Linux 6, Administration by Sander van Vugt.
    - RedHat Linux Networking and System Administration by Terry Colllngs & Kurt Wall.

2. RedHat online support –
https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/ch-Directory_Servers.html