NETWORK ADMINISTRATION USING RHEL

Submitted to:

INSTITUTE OF HIMALAYAN BIORESOURCE TECHNOLOGY
(COUNCIL OF SCIENTIFIC AND INDUSTRIAL RESEARCH), PALAMPUR

SUBMITTED BY:

NITISH SINGH GULERIA

B.Tech (IT) - III

Roll no.2652 (752)

UNIVERSITY INSTITUTE OF INFORMATION TECHNOLOGY, HPU, SHIMLA

1. INTRODUCTION

History

UNIX

In order to understand the popularity of Linux, we need to travel back in time, about 30 years ago...

Imagine computers as big as houses, even stadiums. While the sizes of those computers posed substantial problems, there was one thing that made this even worse: every computer had a different operating system. Software was always customized to serve a specific purpose, and software for one given system didn't run on another system. Being able to work with one system didn't automatically mean that you could work with another. It was difficult, both for the users and the system administrators.

Computers were extremely expensive then, and sacrifices had to be made even after the original purchase just to get the users to understand how they worked. The total cost per unit of computing power was enormous.

Technologically the world was not quite that advanced, so they had to live with the size for another decade. In 1969, a team of developers in the Bell Labs laboratories started working on a solution for the software problem, to address these compatibility issues. They developed a new operating system, which was

1. Simple and elegant.
2. Written in the C programming language instead of in assembly code.
3. Able to recycle code.

The Bell Labs developers named their project "UNIX."

The code recycling features were very important. Until then, all commercially available computer systems were written in a code specifically developed for one system. UNIX on the other hand needed only a small piece of that special code, which is now commonly named the kernel. This kernel is the only piece of code that needs to be adapted for every specific system and forms the base of the UNIX system. The operating system and all other functions were built around this kernel and written in a higher programming language, C. This language was especially developed for creating the UNIX system. Using this new technique, it was much easier to develop an operating system that could run on many different types of hardware.

The software vendors were quick to adapt, since they could sell ten times more software almost effortlessly. Weird new situations came in existence: imagine for instance

computers from different vendors communicating in the same network, or users working on different systems without the need for extra education to use another computer. UNIX did a great deal to help users become compatible with different systems.

Throughout the next couple of decades the development of UNIX continued. More things became possible to do and more hardware and software vendors added support for UNIX to their products.

UNIX was initially found only in very large environments with mainframes and minicomputers (note that a PC is a "micro" computer). You had to work at a university, for the government or for large financial corporations in order to get your hands on a UNIX system.

But smaller computers were being developed, and by the end of the 80's, many people had home computers. By that time, there were several versions of UNIX available for the PC architecture, but none of them were truly free and more important: they were all terribly slow, so most people ran MS DOS or Windows 3.1 on their home PCs.

Linus and Linux

By the beginning of the 90s home PCs were finally powerful enough to run a full blown UNIX. Linus Torvalds, a young man studying computer science at the university of Helsinki, thought it would be a good idea to have some sort of freely available academic version of UNIX, and promptly started to code.

He started to ask questions, looking for answers and solutions that would help him get UNIX on his PC.

In those days plug-and-play wasn't invented yet, but so many people were interested in having a UNIX system of their own, that this was only a small obstacle. New drivers became available for all kinds of new hardware, at a continuously rising speed. Almost as soon as a new piece of hardware became available, someone bought it and submitted it to the Linux test, as the system was gradually being called, releasing more free code for an ever wider range of hardware. These coders didn't stop at their PC's; every piece of hardware they could find was useful for Linux.

Back then, those people were called "nerds" or "freaks", but it didn't matter to them, as long as the supported hardware list grew longer and longer. Thanks to these people, Linux is now not only ideal to run on new PC's, but is also the system of choice for old and exotic hardware that would be useless if Linux didn't exist.

Two years after Linus' post, there were 12000 Linux users. The project, popular with hobbyists, grew steadily, all the while staying within the bounds of the POSIX standard. All the features of UNIX were added over the next couple of years, resulting in the mature operating system Linux has become today. Linux is a full UNIX clone, fit for use on workstations as well as on middle-range and high-end servers. Today, a lot of the

important players on the hard- and software market each have their team of Linux developers; at your local dealer's you can even buy pre-installed Linux systems with official support - eventhough there is still a lot of hard- and software that is not supported, too.

Current application of Linux systems

Today Linux has joined the desktop market. Linux developers concentrated on networking and services in the beginning, and office applications have been the last barrier to be taken down. We don't like to admit that Microsoft is ruling this market, so plenty of alternatives have been started over the last couple of years to make Linux an acceptable choice as a workstation, providing an easy user interface and MS compatible office applications like word processors, spreadsheets, presentations and the like.

On the server side, Linux is well-known as a stable and reliable platform, providing database and trading services for companies like Amazon, the well-known online bookshop, US Post Office, the German army and many others. Especially Internet providers and Internet service providers have grown fond of Linux as firewall, proxy- and web server, and you will find a Linux box within reach of every UNIX system administrator who appreciates a comfortable management station. Clusters of Linux machines are used in the creation of movies such as "Titanic", "Shrek" and others. In post offices, they are the nerve centers that route mail and in large search engine, clusters are used to perform internet searches.These are only a few of the thousands of heavy-duty jobs that Linux is performing day-to-day across the world.

It is also worth to note that modern Linux not only runs on workstations, mid- and high-end servers, but also on "gadgets" like PDA's, mobiles, a shipload of embedded applications and even on experimental wristwatches. This makes Linux the only operating system in the world covering such a wide range of hardware.

1.1 Properties of Linux

Linux Pros

A lot of the advantages of Linux are a consequence of Linux' origins, deeply rooted in UNIX, except for the first advantage, of course:

- Linux is free:

  As in free beer, they say. If you want to spend absolutely nothing, you don't even have to pay the price of a CD. Linux can be downloaded in its entirety from the Internet completely for free. No registration fees, no costs per user, free updates, and freely available source code in case you want to change the behavior of your system.

  Most of all, Linux is free as in free speech:

  The license commonly used is the GNU Public License (GPL). The license says that anybody who may want to do so, has the right to change Linux and eventually to redistribute a changed version, on the one condition that the code is still available after redistribution. In practice, you are free to grab a kernel image, for instance to add support for teletransportation machines or time travel and sell your new code, as long as your customers can still have a copy of that code.

- Linux is portable to any hardware platform:

  A vendor who wants to sell a new type of computer and who doesn't know what kind of OS his new machine will run (say the CPU in your car or washing machine), can take a Linux kernel and make it work on his hardware, because documentation related to this activity is freely available.

- Linux was made to keep on running:

  As with UNIX, a Linux system expects to run without rebooting all the time. That is why a lot of tasks are being executed at night or scheduled automatically for other calm moments, resulting in higher availability during busier periods and a more balanced use of the hardware. This property allows for Linux to be applicable also in environments where people don't have the time or the possibility to control their systems night and day.

- Linux is secure and versatile:

  The security model used in Linux is based on the UNIX idea of security, which is known to be robust and of proven quality. But Linux is not only fit for use as a fort against enemy attacks from the Internet: it will adapt equally to other

situations, utilizing the same high standards for security. Your development machine or control station will be as secure as your firewall.

- Linux is scalable:

  From a Palmtop with 2 MB of memory to a petabyte storage cluster with hundreds of nodes: add or remove the appropriate packages and Linux fits all. You don't need a supercomputer anymore, because you can use Linux to do big things using the building blocks provided with the system. If you want to do little things, such as making an operating system for an embedded processor or just recycling your old 486, Linux will do that as well.

- The Linux OS and most Linux applications have very short debug-times:

  Because Linux has been developed and tested by thousands of people, both errors and people to fix them are usually found rather quickly. It sometimes happens that there are only a couple of hours between discovery and fixing of a bug.

. Linux Cons

- There are far too many different distributions:

  "Quot capites, tot rationes", as the Romans already said: the more people, the more opinions. At first glance, the amount of Linux distributions can be frightening, or ridiculous, depending on your point of view. But it also means that everyone will find what he or she needs. You don't need to be an expert to find a suitable release.

  When asked, generally every Linux user will say that the best distribution is the specific version he is using. So which one should you choose? Don't worry too much about that: all releases contain more or less the same set of basic packages. On top of the basics, special third party software is added making, for example, TurboLinux more suitable for the small and medium enterprise, RedHat for servers and SuSE for workstations. However, the differences are likely to be very superficial. The best strategy is to test a couple of distributions; unfortunately not everybody has the time for this. Luckily, there is plenty of advice on the subject of choosing your Linux. A quick search on Google, using the keywords "choosing your distribution" brings up tens of links to good advise.

  Linux is not very user friendly and confusing for beginners:

  It must be said that Linux, at least the core system, is less userfriendly to use than MS Windows and certainly more difficult than MacOS, but... In light of its popularity, considerable effort has been made to make Linux even easier to use, especially for new users. More information is being released daily, such as this guide, to help fill the gap for documentation available to users at all levels.

- Is an Open Source product trustworthy?

  How can something that is free also be reliable? Linux users have the choice whether to use Linux or not, which gives them an enormous advantage compared to users of proprietary software, who don't have that kind of freedom. After long periods of testing, most Linux users come to the conclusion that Linux is not only as good, but in many cases better and faster that the traditional solutions. If Linux were not trustworthy, it would have been long gone, never knowing the popularity it has now, with millions of users. Now users can influence their systems and share their remarks with the community, so the system gets better and better every day. It is a project that is never finished, that is true, but in an ever changing environment, Linux is also a project that continues to strive for perfection.

1.2 DIFFERENCE

Linux and Unix

Unix is popular operating system, developed by AT&T in 1969 and it has been very important to the development of the Internet. It is a multi-processing, multi-user, family of operating systems that run on a variety of architechtures. UNIX allows more than one user to access a computer system at the same time.

A widely used Open Source Unix-like operating system kernel. Linux was first released by its inventor Linus Torvalds in 1991. Combining the Linux kernel with the GNU software forms the basis of the operating system family generally known as 'Linux'. There are distributions of GNU/Linux for almost every available type of computer hardware from desktop machines to IBM mainframes. The inner workings of GNU/Linux are open and available for anyone to examine and change as long as they make their changes available to the public, as set out in the terms of the GNU General Public License. Because of its robustness and availability, Linux has won popularity in the Open Source community as well as among commercial application developers.

Here is more input:

- Unix requires a more powerful hardware configuration. It will work in large mainframe computers but will not work in an x86 based personal computer. Linux however, (which is built on the concept of Unix) has small hardware requirements and it will work on both a large mainframe computer and an x86 based personal computer.

- Unix is an Operating System developed in olden days in which the kernel, the heart of the OS, interacts directly with the hardware. (note: this is the definition of what a kernel is). Because UNIX treats everything as a file, it provides greater security for users. An example of a UNIX distribution is posix. (note: actually POSIX is a set of standards for interoperability of applications between UNIX and UNIX-like systems). Linux uses a the UNIX architecture as its basis and provides

more facilities and applications. Linux could be considered to be a GUI to the UNIX core. (note: this is plain wrong. GNU/ Linux was rewritten from scratch using UNIX as a guide. GNOME and KDE are GUIs for GNU/Linux). Examples of Linux distributions are Redhat, Fedora, Susee, Mandriva, and Ubuntu. Solaris OS also uses the UNIX kernal almost all UNIX commands will work on solaris in addition to 500 Solaris specific commands. (note: Solaris is also a rewrite of UNIX for x86, and does not use any original UNIX code). Both UNIX and LINUX are Open source. (note: UNIX is proprietary, Linux is open source)

- Unix is the foundation for a number of operating systems, with Linux being the most popular one. Novell and Free BSD are 2 other commonly used Unix varients.(note: Again, the BSD family are based on another rewrite of UNIX for x86, UNIX is not their foundation in the sense implied here)

- UNIX is an operating system created in the early days of computers. More recently, Linux was created as an open-source, freeware operating system. (note: Linux is free software, not freeware. Free software is open source that insists any developer reusing code releases their own work as free software. Freeware is proprietary software distributed at no cost [gratis]) It is "UNIX-LIKE", meaning that it uses many UNIX constructs but also departs from traditional UNIX in many ways. Like UNIX, Linux is faster than many of the other commercially available operating systems. (note: This is a sweeping generalization and depends on the hardware used, and what servers and applications are running) It appears to also be far more robust than any of the Microsoft products. Linux is being used in many time critical applications because of it's speed. It is also used in many applications that need to maintain uptime because Linux, like UNIX, can run for months at a time without rebooting. While the typical method of solving Microsoft problems is to "reboot", that particular requirement does not seem to be appropriate in a Linux/Unix environment. While UNIX has created a windows-like work environment, Linux has improved greatly on that concept. Linux has become a real player in the consumer operating system market... and it's free. While you may want to pay for a Linux distribution, the actual code is free and you are allowed to load it on as many machines as you want. You can get Linux for free if you wish to load it across the internet.

Linux and Windows

Linux - This is a version of the operating system "UNIX" and has gained much popularity because it has proved to be very stable as an operating system in running web servers. This is an open-source software, which means that the source code is free and available on the internet and so you can make any changes to it according to your needs. Since the PC hardware is inexpensive and Linux is essentially free, the combination of the two is considered to be the most practical way of developing inexpensive and reliable HTTP service.

Windows - This is the operating system developed by Microsoft Corporation. There are many versions of this but Windows XP is currently the world's most prevalent system. Windows NT is a more robust system intended for more demanding users. Windows comes with a graphical user interface, which is proven to be very user-friendly. The Windows source code is proprietary, that is, only Microsoft programmers can see it and work on it. This makes Windows stable and permanent.

Both the systems can be compared on the basis of the following attributes:

1) Ease of Use - Linux is considered to be harder to use than the Windows. This is mainly because of its inherent properties from UNIX, which is widely seen as a complicated system. Windows looks easy to use with all its user-friendly graphical interfaces. To compete with this, LINUX has come up with GUI versions KDE and GNOME. Yet, as days go by, more and more people are surprised by the power of LINUX to accomplish many great things by writing a few shell commands to it. For example, you can send, read emails, listen to MP3s and also burn them to CDs, all at the same time. But it takes more than a graphical user interface to make a computer easy to use - it takes tight integration between software and hardware. It requires an operating system that's graphical "from the ground up," so that users don't have to deal with character-based code.

2) Reliability and security - Both are considered to be equally reliable systems but many tests and researches are now indicating that LINUX components are more stable and reliable and can provide a robust, enterprise level environment for the customer over long periods of time.

3) Speed - There is no real difference in terms of speed between the two platforms. However LINUX is slightly faster in processing basic web page data.

4) Functionality - This is where two platforms differ the most. There isn't much that can be achieved in one platform that can't be on the other. The main difference is how the end is achieved. For example, if you need a database-driven website, you can choose either PHP/MySQL combination under Linux or ASP/MS SQL combination under Windows2000. The trend is for solutions to be supported in both platforms. For example, Microsoft FrontPage, one of the most popular website editors out there, is supported on both Linux and Windows2000.

5) Price - here Linux holds a slight edge due to the fact that most of the software licenses are free and its also easier to maintain Linux servers than Windows2000 servers for the same level of performance.

In an increasingly connected world that is dependent on Internet communications and transactions to conduct business, people are rightly concerned about the privacy of the information they share over the Internet and the 'ruggedness' of operating systems and software like firewalls and anti-virus programs to keep their machines secure against outside attack. Customers will be looking into security features more than anything else.

Each of these types of software has its particular features or quality-of-performance preferable to the other. However, everything finally comes down to what the customer wants, so the best software is something that is customised after understanding the importance of hearing and understanding the customers voice.

Linux and Mac Os-X

MAC OS is a commercially available Operating System that typically runs on Apple Computer products. The file system, commands, data structures and even the icons are different from most other operating systems.

Linux is a UNIX-like operating system, that is open and free. Open meaning that every detail about the operating system is freely available for review and evaluation and that you can not only look at it but you can change it for your own purposes. Free means that it doesn't cost anyone to use it. Some companies have put together "distributions" of Linux that will install it on your computer. The distributions can become elaborate and as such these companies charge a few dollars for the distribution. But the Linux inside is free to own, operate and upgrade.

A Comparison of Common DOS and Linux Commands

Many Linux commands typed at a shell prompt are similar to the commands you would type in MS-DOS. In fact, some commands are identical.

This appendix provides common commands used at the MS-DOS prompt in Windows 9*x* and their counterparts in Linux. Basic examples of how the command is used at the Linux shell prompt are also provided. Note that these commands usually have a number of options. To learn more about each command, read its associated man page (for example, type **`man ls`** at the shell prompt to read about the `ls` command).

DIFFERENCE:

Windows no more have MS-DOS Prompt since ME. The black box you see in Windows NT (2k, XP, Vista) is called Command Prompt, and is significantly different compared to real DOS prompt. Windows' Command Prompt is a shell, Linux Terminal is a shell. Linux's Terminal usually doesn't run things directly, they have a shell program like Bash (Bourne Again Shell), sh (Bourne Shell), csh (C Shell), ksh (Korn Shell), dash (Debian Almquist Shell). These shells provides a convenient environment for program to work in and each of them provides their own scripting languages for user to automate the shell.

On contrary, there is only one official shell for Windows, the Command Prompt (well, I lied, there is PowerShell but practically nobody apart from programmer uses it... as of yet). Command Prompt has a scripting language too, usually called batch file (with file extension .bat).

Despite that there is many shell in Linux, many share the same basic commands. For

example, file deletion is "rm", change directory "cd", move file "mv", open help "man" or "info", etc. Windows' Command Prompt equivalent is "del", "cd", "move", and "help" respectively.

In general, Linux's shell scripting language is much, much, more powerful than batch scripting language. This is because of difference in philosophy, in Linux, program's main interface should be text stream and GUI is made on top of that text stream, in Windows, they ignore text stream and thought that modern OS should be GUI oriented. This ignorance of text stream is both Window's power and weaknesses. Many system administrator prefers a command line interface to work with since command line is simple, thus more powerful than GUI. Many users, on the other hand, prefers GUI, since they think clicking is easier than typing. Problem is, it is easy to create a GUI from text-based program, but not the other way.

2. LINUX BASICS

The Linux Shell:

To use Linux or any Unix-like system, for that matter we need to know a few things about shells. A shell is a program that acts as an intermediary between user and the operating system
In a DOS environment, **command.com** acts as shell. Linux shells have more interesting names (like bash, pdksh, and tcsh), but they do pretty much the same thing. In addition to translating our commands into something the kernel can understand and act upon, the shell adds some important functions that the base operating system doesn't supply.
Using a Linux shell means working with a command line, which is much like working from a DOS prompt

Root Vs Other users:

The Root User:

In any Linux system, the root user is called a *superuser* because it has powers far beyond those of normal users.
As **root**, we can access files and change the system in ways other users cannot. But we can also wipe out our entire hard drive in just ten keystrokes.

Unless we plan to install new software or make configuration of our system, we should log in to Linux as a user other than **root**.

Adding New Users:

In the Linux universe, new users do not evolve--they are created by a benevolent superuser. To add a new user account, log in to the root account and enter a command like the one shown here. There's no limit to the number of new users we can add.

adduser nitish

After using the **adduser** command, we must assign a password to the new account before it can be used. Use this command to set the password:

passwd nitish123

. . . and enter the initial password for nitish when prompted.

Linux Virtual Console:-

Even if we not have more than one physical console (a monitor plus a keyboard) connected to our PC, we can use *virtual consoles* to log in simultaneously to more than one account on our system.

We can use virtual consoles to perform two activities in parallel. For example, I used one virtual console to write this section and another to test the commands as they were introduced. We can even use our mouse to cut and paste text from one virtual console to another. When we start our Linux system and get the log-in prompt, we're looking at console number 1. Log in as root here; then press ctl**+**alt-F2**.** We should then see another log-in prompt. We can log in as user say nitish on this console and then press ctl+alt-F3 to access a third console or press ctl+alt-F1 to return to the first console.

Virtual consoles come in particularly handy if we have a long-running task to perform, like installing a big software package from a CD-ROM--we can pop over to another console and log in again to stay productive while our CD-ROM churns away.

Note: We don't have to use a different user account for each console. Linux lets we log in to an account multiple times simultaneously.

Shutting down Linux server:

If we're ready to cash in our console and call it a day, use the **logout** command. Entering

#logout:

At the command prompt exits our current user account and returns we to the log-in prompt. (The exit command does the same thing as logout.) To log out from multiple consoles, use alt-F*n* to switch between consoles and then log out from each one. But note that even if we log out from all of our active consoles, Linux is still running.

#shutdown -h now

We'll see some messages indicating that various subsystems are being shut down, and then the computer will be reset. When we see a message indicating that shutdown is complete, it's safe to turn off our PC.

Note: Pressing ctrl-alt-delete will also safely shut down our Red Hat Linux system.

2.1 Linux Commands:-

When we enter a command in Linux, we type a command at a prompt and then press **enter**. Commands can be more than one word--some require switches (which modify the command's behavior) and/or file names (which tell the command what data to act on). Let's dissect the command shown here:

$ **ls -l sample.doc**

SOME LINUX COMMANDS

1. mkdir : Creates a directory
Eg. mkdir dir1

2. ls : lists all the files and directories in the current directory.
Eg.
• ls → Simple listing.
• ls -l → long listing(includes permissions,size etc.)
• ls -t → lists in the modification time order
• ls -a → lists all files(including hidden files)

3. cd : Changes the directory.
Eg.
• cd dir1 → Change to dir1
• cd .. → You will move one level up in the directory hierarchy i.e, going into the parent directory.
• cd → takes to your home directory. Default directory is your home directory.

4. pwd : prints the path of the current working directory

5. gedit : Graphical text editor.
Eg.
• gedit & → opens the gedit editor(as a background job).Type something in to that and save the file.
• gedit file1.txt & → opens gedit showing the contents of file1.txt

6. cp : copy the contents from source to destination
Eg.
• cp file1.txt file2.txt →Make a copy of file1.txt and names it as file2.txt
• cp file1.txt dir1 → copies the file1.txt into directory dir1. The file will have same name “file1.txt” in dir1.

7. mv:
Eg.
1
• mv file1.txt file2.txt → renames the file1.txt with file2.txt. Use ”ls” to see the result. Now you don’t find file1.txt.
• mv file1.txt dir1 → moves file file1.txt in to directory dir1. The file is deleted in the current directory and in dir1 it will have the same name file1.txt

8. rm: Removes files or directories
Eg.
• rm file1.txt → Removes file1.txt
• rm -f file1.txt → Removes the file with our asking for confirmation(forceful removing)
• rm -rf dir1 → Removes the directory dir1 recursively,i.e, it will delete all the directory hierarchy below the dir1 also.
Note:
• Use rm command with extreme caution. Data will be lost forever.
• If the you don’t have the write permission for that file and you are trying to remove that file, then it asks your confirmation for the deletion of a file.

9. find: Search for a file or directory. Eg. find -name file1.txt → searches for file1.txt in the current directory hierarchy.

10. history : displays all the recently used commands

11. cat:
Eg.
• cat file1.txt → displays the contents of the fiel1.txt on to the terminal.
• cat file1.txt file2.txt → concatenates the contents of both the files and displays on to the terminal.

12. echo: echoes the content on the screen
Eg.
• echo I Love India → Displays “I Love India” on the terminal.

13. grep :
Eg.
• grep dog file1.txt → displays the lines containing ”dog” in file1.txt
• grep -i dog file1.txt → Case Insensitive search. Displays all line matching with dog not considering the case(eg DOG,Dog,doG, etc..)

14. wc: Word count
Eg.
• wc file1.txt → Displays number of lines, words, characters present in

the file1.txt
• wc -l file1.txt → Prints only the number of lines of file1.txt

15. sort:
Eg.
• sort file1.txt → sorts the lines of file1.txt
• sort -f file1.txt → sort the lines of file1.txt with ignoring case
• sort -n file1.txt → sort the line of file1.txt in numeric order
• sort -r file1.txt → prints the reverse order of sorted lines of file1.txt

16. chmod: change mode/permissions
Eg.
• chmod 777 file1.txt → gives read,write and execute permission to ownern,group and others. Can be checked by using ls -l command.
• chmod 764 file1.txt → gives read,write,execute permissions to owner, read and write permissions to group and only read permission to the others.

17. chown : change ownership of a file/directory
Eg. chown remo file1.txt → makes remo as the owner of file1.txt. Can be observed by using the ”stat file1.txt” command before and after using this command.

18. su :
Eg.
• su remo → will ask the password for remo account and changes the user.
• su → will ask the root password and changes the user as root(super user).

19. passwd: Change password
Eg. passwd → ask your old password first and then a new password to choose.

20. who: prints the logged on users
Eg.
• who
• who -b → prints the last system boot time.

21. ps: prints the process’s information(pid,process name) which are created by this terminal only.
Eg. ps

22. top: Shows system processes in real time(ps only gives a snapshop).

23. bg: makes the process as backgroud job.

Eg. Type “gedit” in terminal and this will start the process in foreground. Now type “ctrl+z” this will stop the gedit process. Now type “bg” and this makes your gedit process to run in background.

24. jobs:
Eg. jobs → gives the jobs and their ids that are running in background.

25. fg:
Eg.
• fg → makes the last background process as foreground process.
• fg hidi → (“id1” can be taken from jobs command)background job with id “id1” will be made as forground process.

26. tar: compresses and decompresses the files.
Eg.
• tar -zcvf dir1.tar.gz dir1 → To compress dir1
• tar -zxvf archive.tar.gz → To decompress archive.tar

27. zip & unzip:
Eg.
• zip file1.txt.zip file1.txt → zip file file1.txt
• unzip file1.txt.zip → unzip file1.txt.zip
• zip -r dir1.zip dir1 → zip a directory dir1
• unzip dir1.zip → unzip dir1.zip

28. mount & umount: Need to be superuser to execute these commands. Check the partition to be mounted in /dev dir.Create a directory dir1 in your home directory. Now use the following command “mount /dev/sda5 dir1” this makes you to access the files on sda5 device through dir1 directory(i.e, you can see all the files in device sda5 in dir1). To unmount this device sda5 use the follwoing command “umount dir1”

29. du:
Eg.
• du → Prints the each file’s size in the current directory.
• du file1.txt → Prints only for file1.txt
• du –time → Gives last modification time of each file.

30. df :
Eg. df → Prints all the disks and the memory available and used.

31. quota:
Eg. quota -v → Gives the memory used and memory allocated to you.

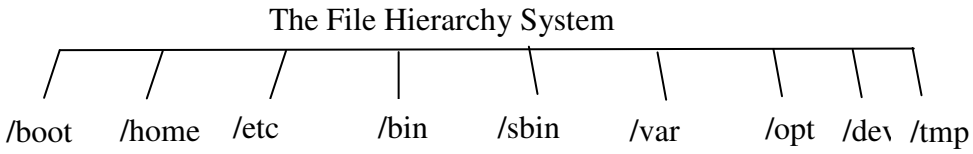
32. reboot & shutdown: needs to be superuser to execute these commands
Eg. reboot → will restart the system

Linux Commands Are Case Sensitive:

One of the most important things to remember about entering commands in any Unix environment is that case matters. In DOS, we can type **DIR** or **dir** or even **DiR** and get the same result, but not in Linux. The best rule to follow is this: Almost everything in Linux is lowercase, so avoid capital letters when naming our files.

2.2 Linux File Systems:

File system refers to the files and directories stored on a computer. A file system can have different formats called *file system types*. These formats determine how the information is stored as files and directories. Some file system types store redundant copies of the data, while some file system types make hard drive access faster.



Where,

/boot contains everything required for the boot process except for configuration files not needed at boot time (the most notable of those being those that belong to the GRUB boot-loader) and the map installer.

/home contains user's home folders and profiles

/etc contains all system related configuration files in here or in its sub-directories

/bin contains Essential command executable binaries eg. Cat,chmod,date, mkdir etc.

/sbin contains Essential system binaries i.e system administration binaries like fdisk

/var contains variable data like system logging files, mail and printer spool directories, and transient and temporary files /opt contains all the software and add-on packages that are not part of the default installation

/dev contains device files (containing information about devices)

/tmp contains mostly files that are required temporarily

In addition to this there is a Proc file system:

/proc is very special in that it is also a virtual file system. It's sometimes referred to as a process information pseudo-file system. It doesn't contain 'real' files but runtime system information (e.g. system memory, devices mounted, hardware configuration, etc). For this reason it can be regarded as a control and information centre for the kernel. In fact, quite a lot of system utilities are simply calls to files in this directory.

For example, # /proc/meminfo (it will provide information about the memory)

Note: */root is the home directory of the System Administrator ("root")*

The ext3 File System:

The default file system is the journaling *ext3* file system. The ext3 file system is essentially an enhanced version of the ext2 file system. These improvements provide the following advantages:

Availability

> After an unexpected power failure or system crash (also called an *unclean system shutdown*), each mounted ext2 file system on the machine must be checked for consistency by the e2fsck program. This is a time-consuming process that can delay system boot time significantly, especially with large volumes containing a large number of files. During this time, any data on the volumes is unreachable.
>
> The journaling provided by the ext3 file system means that this sort of file system check is no longer necessary after an unclean system shutdown. The only time a consistency check occurs using ext3 is in certain rare hardware failure cases, such as hard drive failures. The time to recover an ext3 file system after an unclean system shutdown does not depend on the size of the file system or the number of files; rather, it depends on the size of the *journal* used to maintain consistency. The default journal size takes about a second to recover, depending on the speed of the hardware.

Data Integrity

> The ext3 file system provides stronger data integrity in the event that an unclean system shutdown occurs. The ext3 file system allows we to choose the type and level of protection that our data receives. By default, the ext3 volumes are configured to keep a high level of data consistency with regard to the state of the file system.

Speed

Despite writing some data more than once, ext3 has a higher throughput in most cases than ext2 because ext3's journaling optimizes hard drive head motion. We can choose from three journaling modes to optimize speed, but doing so means trade offs in regards to data integrity.

Easy Transition

It is easy to change from ext2 to ext3 and gain the benefits of a robust journaling file system without reformatting.

If we performed a fresh installation, the default file system assigned to the system's Linux partitions is ext3

Linux Swap Space:

Swap space in Linux is used when the amount of physical memory (RAM) is full. If the system needs more memory resources and the physical memory is full, inactive pages in memory are moved to the swap space. While swap space can help machines with a small amount of RAM, it should not be considered a replacement for more RAM. Swap space is located on hard drives, which have a slower access time than physical memory.

Swap space can be a dedicated swap partition (recommended), a swap file, or a combination of swap partitions and swap files.

The size of our swap space should be equal to twice our computer's RAM, or 32 MB, whichever amount is larger, but no more than 2048 MB (or 2 GB).

2.3 RAID:-

The basic idea behind RAID is to combine multiple small, inexpensive disk drives into an array to accomplish performance or redundancy goals not attainable with one large and expensive drive. This array of drives appears to the computer as a single logical storage unit or drive.

RAID is a method in which information is spread across several disks, using techniques such as *disk striping* (RAID Level 0), *disk mirroring* (RAID level 1), and *disk striping with parity* (RAID Level 5) to achieve redundancy, slower latency and/or increase bandwidth for reading or writing to disks, and maximize the ability to recover from hard disk crashes.

The underlying concept of RAID is that data may be distributed across each drive in the array in a consistent manner. To do this, the data must first be broken into consistently-sized *chunks* (often 32K or 64K in size, although different sizes can be used). Each chunk is then written to a hard drive in RAID according to the RAID level used. When the data

is to be read, the process is reversed, giving the illusion that multiple drives are actually one large drive.

Use of RAID:

Anyone who needs to keep large quantities of data on hand (such as a system administrator) would benefit by using RAID technology. Primary reasons to use RAID include:

- Enhanced speed
- Increased storage capacity using a single virtual disk
- Lessened impact of a disk failure

RAID Levels and Linear Support:

RAID supports various configurations, including levels 0, 1, 4, 5, and linear. These RAID types are defined as follows:

- *Level 0* — RAID level 0, often called "striping," is a performance-oriented striped data mapping technique. This means the data being written to the array is broken down into strips and written across the member disks of the array, allowing high I/O performance at low inherent cost but provides no redundancy. The storage capacity of a level 0 array is equal to the total capacity of the member disks in a Hardware RAID or the total capacity of member partitions in a Software RAID.
- *Level 1* — RAID level 1, or "mirroring," has been used longer than any other form of RAID. Level 1 provides redundancy by writing identical data to each member disk of the array, leaving a "mirrored" copy on each disk. Mirroring remains popular due to its simplicity and high level of data availability. Level 1 operates with two or more disks that may use parallel access for high data-transfer rates when reading but more commonly operate independently to provide high I/O transaction rates. Level 1 provides very good data reliability and improves performance for read-intensive applications but at a relatively high cost. [1] The storage capacity of the level 1 array is equal to the capacity of one of the mirrored hard disks in a Hardware RAID or one of the mirrored partitions in a Software RAID.
- *Level 4* — Level 4 uses parity [2] concentrated on a single disk drive to protect data. It is better suited to transaction I/O rather than large file transfers. Because the dedicated parity disk represents an inherent bottleneck, level 4 is seldom used without accompanying technologies such as write-back caching. Although RAID level 4 is an option in some RAID partitioning schemes, it is not an option allowed in Red Hat Enterprise Linux RAID installations. [3] The storage capacity of Hardware RAID level 4 is equal to the capacity of member disks, minus the capacity of one member disk. The storage capacity of Software RAID level 4 is equal to the capacity of the member partitions, minus the size of one of the partitions if they are of equal size.

- *Level 5* — This is the most common type of RAID. By distributing parity across some or all of an array's member disk drives, RAID level 5 eliminates the write bottleneck inherent in level 4. The only performance bottleneck is the parity calculation process. With modern CPUs and Software RAID, that usually is not a very big problem. As with level 4, the result is asymmetrical performance, with reads substantially outperforming writes. Level 5 is often used with write-back caching to reduce the asymmetry. The storage capacity of Hardware RAID level 5 is equal to the capacity of member disks, minus the capacity of one member disk. The storage capacity of Software RAID level 5 is equal to the capacity of the member partitions, minus the size of one of the partitions if they are of equal size.

3. NETWORK BASICS

A network is a collection of computers and devices connected to each other. The network allows computers to communicate with each other and share resources and information. The Advance Research Projects Agency (ARPA) designed "Advanced Research Projects Agency Network" (ARPANET) for the United States Department of Defense. It was the first computer network in the world in late 1960's and early 1970's.

Connection method

Computer networks can also be classified according to the hardware and software technology that is used to interconnect the individual devices in the network, such as Optical fiber, Ethernet, Wireless LAN, HomePNA, or Power line communication.

Ethernet uses physical wiring to connect devices. Frequently deployed devices include hubs, switches, bridges and/or routers.

Wireless LAN technology is designed to connect devices without wiring. These devices use radio waves or infrared signals as a transmission medium.

3.1 Network topology

Network topology is the study of the *arrangement* or *mapping* of the elements(links, nodes, etc.) of a network, especially the physical (real) and logical (virtual) interconnections between nodes. A local area network (LAN) is one example of a network that exhibits both a physical topology and a logical topology. Any given node in the LAN will have one or more links to one or more other nodes in the network and the mapping of these links and nodes onto a graph results in a geometrical shape that determines the physical topology of the network. Likewise, the mapping of the *flow of data* between the nodes in the network determines the logical topology of the network. The physical and logical topologies *might* be identical in any particular network but they also may be *different*.

Basic types of topologies

There are six basic types of topology in networks:

1. Bus topology
2. Star topology
3. Ring topology
4. Mesh topology
5. Tree topology
6. Hybrid topology

Bus

Linear bus

The type of network topology in which all of the nodes of the network are connected to a common transmission medium which has exactly two endpoints (this is the 'bus', which is also commonly referred to as the backbone, or trunk) – all data that is transmitted between nodes in the network is transmitted over this common transmission medium and is able to be received by all nodes in the network virtually simultaneously

Distributed bus

The type of network topology in which all of the nodes of the network are connected to a common transmission medium which has more than two endpoints that are created by adding branches to the main section of the transmission medium – the physical distributed bus topology functions in exactly the same fashion as the physical linear bus topology (i.e., all nodes share a common transmission medium).

Star

The type of network topology in which each of the nodes of the network is connected to a central node with a point-to-point link in a 'hub' and 'spoke' fashion, the central node being the 'hub' and the nodes that are attached to the central node being the 'spokes' (e.g., a collection of point-to-point links from the peripheral nodes that converge at a central node) – all data that is transmitted between nodes in the network is transmitted to this central node, which is usually some type of device that then retransmits the data to some or all of the other nodes in the network, although the central node may also be a simple common connection point (such as a 'punch-down' block) without any active device to repeat the signals.

Ring

The type of network topology in which each of the nodes of the network is connected to two other nodes in the network and with the first and last nodes being connected to each other, forming a ring – all data that is transmitted between nodes in the network travels from one node to the next node in a circular manner and the data generally flows in a single direction only.

Tree

Also known as a **hierarchical network**.

The type of network topology in which a central 'root' node (the top level of the hierarchy) is connected to one or more other nodes that are one level lower in the hierarchy (i.e., the second level) with a point-to-point link between each of the second

level nodes and the top level central 'root' node, while each of the second level nodes that are connected to the top level central 'root' node will also have one or more other nodes that are one level lower in the hierarchy (i.e., the third level) connected to it, also with a point-to-point link, the top level central 'root' node being the only node that has no other node above it in the hierarchy (The hierarchy of the tree is symmetrical.) Each node in the network having a specific fixed number, of nodes connected to it at the next lower level in the hierarchy, the number, being referred to as the 'branching factor' of the hierarchical tree.

Hybrid network topologies

The hybrid topology is a type of network topology that is composed of one or more interconnections of two or more networks that are based upon the same physical topology, but where the physical topology of the network resulting from such an interconnection does not meet the definition of the original physical topology of the interconnected networks (e.g., the physical topology of a network that would result from an interconnection of two or more networks that are based upon the physical star topology might create a hybrid topology which resembles a mixture of the physical star and physical bus topologies or a mixture of the physical star and the physical tree topologies, depending upon how the individual networks are interconnected, while the physical topology of a network that would result from an interconnection of two or more networks that are based upon the physical distributed bus network retains the topology of a physical distributed bus network).

3.2 Types of networks

Personal area network

A personal area network (PAN) is a computer network used for communication among computer devices close to one person. Some examples of devices that are used in a PAN are printers, fax machines, telephones, PDAs and scanners. The reach of a PAN is typically about 20-30 feet (approximately 6-9 meters), but this is expected to increase with technology improvements**.**

Local area network

local area network (LAN) is a computer network covering a small physical area, like a home, office, or small group of buildings, such as a school, or an airport. Current LANs are most likely to be based on Ethernet technology. For example, a library may have a wired or wireless LAN for users to interconnect local devices (e.g., printers and servers) and to connect to the internet. On a wired LAN, PCs in the library are typically connected by category 5 (Cat5) cable, running the IEEE 802.3 protocol through a system of interconnected devices and eventually connect to the Internet. The cables to the servers are typically on Cat 5e enhanced cable, which will support IEEE 802.3 at 1 Gbit/s. A wireless LAN may exist using a different IEEE protocol, 802.11b, 802.11g or possibly 802.11n. The staff computers (bright green in the figure) can get to the color printer, checkout records, and the academic network *and* the Internet. All user computers can get

to the Internet and the card catalog. Each workgroup can get to its local printer. Note that the printers are not accessible from outside their workgroup.

All interconnected devices must understand the network layer (layer 3), because they are handling multiple subnets (the different colors). Those inside the library, which have only 10/100 Mbit/s Ethernet connections to the user device and a Gigabit Ethernet connection to the central router, could be called "layer 3 switches" because they only have Ethernet interfaces and must understand IP. It would be more correct to call them access routers, where the router at the top is a distribution router that connects to the Internet and academic networks' customer access routers.

The defining characteristics of LANs, in contrast to WANs (wide area networks), include their higher data transfer rates, smaller geographic range, and lack of a need for leased telecommunication lines. Current Ethernet or other IEEE 802.3 LAN technologies operate at speeds up to 10 Gbit/s. This is the data transfer rate. IEEE has projects investigating the standardization of 100 Gbit/s, and possibly 40 Gbit/s.

Campus area network

A **campus area network (CAN)** is a computer network made up of an interconnection of local area networks (LANs) within a limited geographical area. It can be considered one form of a metropolitan area network, specific to an academic setting.

In the case of a university campus-based campus area network, the network is likely to link a variety of campus buildings including; academic departments, the university library and student residence halls. A campus area network is larger than a local area network but smaller than a wide area network (WAN) (in some cases).

The main aim of a campus area network is to facilitate students accessing internet and university resources. This is a network that connects two or more LANs but that is limited to a specific and contiguous geographical area such as a college campus, industrial complex, office building, or a military base. A CAN may be considered a type of MAN (metropolitan area network), but is generally limited to a smaller area than a typical MAN. This term is most often used to discuss the implementation of networks for a contiguous area. This should not be confused with a Controller Area Network. A LAN connects network devices over a relatively short distance. A networked office building, school, or home usually contains a single LAN, though sometimes one building will contain a few small LANs (perhaps one per room), and occasionally a LAN will span a group of nearby buildings. In TCP/IP networking, a LAN is often but not always implemented as a single IP subnet.

Metropolitan area network

A **metropolitan area network (MAN)** is a network that connects two or more local area networks or campus area networks together but does not extend beyond the boundaries of

the immediate town/city. Routers, switches and hubs are connected to create a metropolitan area network.

Wide area network

A **wide area network (WAN)** is a computer network that covers a broad area (i.e., any network whose communications links cross metropolitan, regional, or national boundaries [1]). Less formally, a WAN is a network that uses routers and public communications links [1]. Contrast with personal area networks (PANs), local area networks (LANs), campus area networks (CANs), or metropolitan area networks (MANs) are usually limited to a room, building, campus or specific metropolitan area (e.g., a city) respectively. The largest and most well-known example of a WAN is the Internet. A WAN is a data communications network that covers a relatively broad geographic area (i.e. one city to another and one country to another country) and that often uses transmission facilities provided by common carriers, such as telephone companies. WAN technologies generally function at the lower three layers of the OSI reference model: the physical layer, the data link layer, and the network layer.

Global area network

A **global area networks (GAN)** specification is in development by several groups, and there is no common definition. In general, however, a GAN is a model for supporting mobile communications across an arbitrary number of wireless LANs, satellite coverage areas, etc.

3.3 GATEWAYS

GATEWAY:A node on a network that serves as an entrance to another network. In enterprises, the gateway is the computer that routes the traffic from a workstation to the outside network that is serving the Web pages. In homes, the gateway is the ISP that connects the user to the internet.

In enterprises, the gateway node often acts as a proxy server and a firewall. The gateway is also associated with both a router, which use headers and forwarding tables to determine where packets are sent, and a switch, which provides the actual path for the packet in and out of the gateway.

3.4 Networking ROUTERS:-

Much of the work to get a message from one computer to another is done by routers, because they're the crucial devices that let messages flow **between networks**, rather than within networks.
The router is the only device that sees every message sent by any computer on either of the company's networks. When the animator in our example sends a huge file to another animator, the router looks at the recipient's address and keeps the traffic on the animator's network. When an animator, on the other hand, sends a message to the bookkeeper asking

about an expense-account check, then the router sees the recipient's address and forwards the message between the two networks.

One of the tools a router uses to decide where a packet should go is a configuration table. A configuration table is a collection of information, including:

- Information on which connections lead to particular groups of addresses
- Priorities for connections to be used
- Rules for handling both routine and special cases of traffic

A configuration table can be as simple as a half-dozen lines in the smallest routers, but can grow to massive size and complexity in the very large routers that handle the bulk of Internet messages.

A router, then, has two separate but related jobs:

- The router ensures that information doesn't go where it's not needed. This is crucial for keeping large volumes of data from clogging the connections of "innocent bystanders."
- The router makes sure that information does make it to the intended destination.

In performing these two jobs, a router is extremely useful in dealing with two separate computer networks. It joins the two networks, passing information from one to the other and, in some cases, performing translations of various **protocols** between the two networks. It also protects the networks from one another, preventing the traffic on one from unnecessarily spilling over to the other. As the number of networks attached to one another grows, the configuration table for handling traffic among them grows, and the processing power of the router is increased. Regardless of how many networks are attached, though, the basic operation and function of the router remains the same. Since the Internet is one huge network made up of tens of thousands of smaller networks, its use of routers is an absolute necessity.

4. SERVER

Server Definition:

Server is a software program, or the computer on which that program runs, that provides a specific kind of service to *client* software running on the same computer or other computers on a network.

The *client-server* model is an *architecture* (i.e., a system design) that divides processing between clients and servers that can run on the same machine or on different machines on the same network. It is a major element of modern operating system and network design.

The client provides the user interface, such as a GUI (graphical user interface), and performs some or all of the processing on requests it makes from the server, which maintains the data and processes the requests.

An example is a *web server*, which stores files related to web sites and *server* (i.e., sends) them across the Internet to clients (i.e., web browsers) when requested by a user. By far the most popular web server program is *Apache*, which is claimed to host more than 68 percent of all web sites on the Internet.

A single computer can have multiple server software applications running on it. Also, it is possible for a computer to be both a client and a server simultaneously; this is accomplished by connecting to itself in the same way that a separate computer would.

Many large enterprises employ numerous dedicated server machines. A collection of servers in one location is commonly referred to as a *server farm*. If very heavy traffic is expected, *load balancing* is usually employed to distribute the requests among the various servers so that no single machine is overwhelmed.

Due to the continual demand for ever more powerful servers in ever decreasing spaces, higher density configurations have been developed. In particular, *blade server* incorporate a number of sets of server hardware, sometimes as many as nine, each housed inside a high-density module known as a *blade*, within the space typically occupied by a single computer.

4.1 Linux RPM:

RPM is a powerful Package Manager, which can be used to build, install, query, verify, update, and erase individual software packages. A package consists of an archive of files and meta-data used to install and erase the archive files. The meta-data includes helper scripts, file attributes, and descriptive information about the package. Packages come in two varieties: binary packages, used to encapsulate software to be installed, and source packages, containing the source code and recipe necessary to produce binary packages.

RPM Terminology

When working with RPM, understanding the package concept is key. RPM packages are provided as compressed archive files that contain one or more files, as well as instructions specifying installation information about those files, including the ownerships and permissions that should be applied to each file during installation. The instructions can also contain scripts to be run after installation or before uninstallation. These package files are extremely convenient; they provide a single file that can be easily transferred between machines for installation rather than having to transfer each file to be installed.

To help in installation and management, all package files are labeled with highly identifiable names. Package files have four-part names, which typically look something like:

- ftp-0.17-22.athlon.rpm
- ftp-0.17-22.i586.rpm
- ftp-0.17-22.i686.rpm
- ftp-0.17-22.i386.rpm
- rootfiles-7.2-1.noarch.rpm

Here, the four parts of each name are separated from each other by dashes or periods. The structure of the package file name is

name-version-release.architecture.rpm

The name identifies what software is contained within the archive file. Typically, this is a name of an application or package that the archive installs on the system. The rootfiles package, for example, is not an application but is a collection of basic environmental configuration files for the root user's account (such as /root/.bashrc, the root user's Bash configuration file) that provides a usable, preconfigured working environment for the root user.

The second field in every package file's name is the version field. This field identifies the version number of the software that is contained in the package file. For example, ftp-0.17-22 indicates the RPM holds the 0.17 release of the FTP version, and rootfiles-7.2 is the 7.2 release of the rootfiles configuration files.

Every package file name also has a third component: the release field. This field identifies which release of that version of the software the package file contains. Package files contain both software and instructions about how to install that software. As packages of a particular version of software are being prepared, mistakes are sometimes made in these instruction files, or bugs are sometimes fixed within a software version; more recent package files of that software version need to be prepared that correct the problem. The –1 in the rootfiles-7.2-1 package shows this is the first release of the 7.2 version of the rootfiles software. The packager of rootfiles version 7.2 got everything right on the first try and had no need to prepare more than one release. The –3 in the ftp-0.17-22 package, on the other hand, is the twenty second release of the 0.17 version of the FTP. This release incorporates new patches to fix bugs present in older releases of the 0.17 version of the FTP package. The software packager increased the release number so that end users could distinguish the more recent, bug-fixed package file from the older, less bug-free package file.

The final field in package file names is the architecture, which identifies the system types for which the package file is appropriate. For example, the ftp-0.17-22.i586 is intended for use on machines with an i586 (Pentium-class) CPU or better. An architecture name of noarch indicates this is a special architecture such that the files in the package work on any architecture. Typically, this is because the files are all interpreted scripts, not binary executables, or are documentation.

Using RPM

i) Installing an linux package:

 # rpm -ivh ftp-0.17-22.i386.rpm

Where: v-Print verbose information - normally routine progress messages will be displayed,

-h, --hash Print 50 hash marks as the package archive is unpacked. Use with **-v|--verbose** for a nicer display.

 # rpm -i ftp://ftp.redhat.com/pub/redhat/RPMS/ ftp-0.17-22.i386.rpm

Used to install a RPM package. Note that RPM packages have file naming conventions like ftp-0.17-22.i386.rpm, which include the package name (ftp), version (0.17), release (22), and architecture (i386). Also notice that RPM understands FTP and HTTP protocols for installing and querying remote RPM files.

ii) Erasing an linux package:

 # rpm -e ftp-0.17-22

To uninstall a RPM package. Note that we used the package name foo, not the name of the original package file ftp-0.17-22.i386.rpm above.

iii)Upgrading an linux package:

 # rpm -Uvh ftp**-0.17-22.i386.rpm**

 # rpm -Uvh ftp://ftp.redhat.com/pub/redhat/RPMS/ ftp-0.17-22.i386.rpm

To upgrade a RPM package. Using this command, RPM automatically uninstall the old version of the ftp package and install the new package. It is safe to always use rpm -Uvh to install and upgrade packages, since it works fine even when there are no previous versions of the package installed! Also notice that RPM understands FTP and HTTP protocols for upgrading from remote RPM files.

iv) Querying an linux package:

 # rpm -qa
To query all installed packages. This command will print the names of all installed packages installed on our Linux system.

 # rpm -q ftp
To query a RPM package. This command will print the package name, version, and release number of the package ftp only if it is installed. Use this command to verify that a package is or is not installed on our Linux system.

 # rpm -qi ftp
To display package information. This command display package information including the package name, version, and description of the installed program. Use this command to get detailed information about the installed package.

 # rpm -ql ftp
To list files in installed package. This command will list all of files in an installed RPM package. It works only when the package is already installed on our Linux system.

 # rpm -qf /usr/bin/mysql
mysql-3.23.52-3
Which package owns a file? This command checks to determine which installed package a particular file belongs to.

v) Verifying an installed linux package:
 # rpm --verify mysql

To verify an installed package. This command will list all files that do NOT pass the

verify tests (done on size, MD5 signature, etc).

5. NETWORK SERVICES

5.1 Network File System (NFS)

NFS allows a system to share directories and files with others over a network. By using NFS, users and programs can access files on remote systems almost as if they were local files.

Some of the most notable benefits that NFS can provide are:

- Local workstations use less disk space because commonly used data can be stored on a single machine and still remain accessible to others over the network.
- There is no need for users to have separate home directories on every network machine. Home directories could be set up on the NFS server and made available throughout the network.
- Storage devices such as floppy disks, CDROM drives, and Zip® drives can be used by other machines on the network. This may reduce the number of removable media drives throughout the network.

How NFS Works

NFS consists of at least two main parts: a server and one or more clients. The client remotely accesses the data that is stored on the server machine. In order for this to function properly a few processes have to be configured and running.

Daemon	**Description**
nfsd	The NFS daemon which services requests from the NFS clients.
mountd	The NFS mount daemon which carries out the requests that nfsd(d) passes on to it.
rpcbind	This daemon allows NFS clients to discover which port the NFS server is using.

NFS serving is taken care of by five daemons: **rpc.nfsd**, which does most of the work; **rpc.lockd** and **rpc.statd**, which handle file locking; **rpc.mountd**, which handles the initial mount requests, and **rpc.rquotad**, which handles user file quotas on exported volumes. **lockd** is called by **nfsd** upon demand, so we do not need to worry about starting it our self. **statd** will need to be started separately. Most recent Linux distributions will have startup scripts for these daemons.

Server Side Configuration:

There are three main configuration files which we will need to edit to set up an NFS server:

1) /etc/exports
2) /etc/hosts.allow
3) /etc/hosts.deny

The /etc/exports Configuration File

The /etc/exports file controls which file systems are exported to remote hosts and specifies options. Blank lines are ignored, comments can be made by starting a line with the hash mark (#), and long lines can be wrapped with a backslash (\). Each exported file system should be on its own individual line, and any lists of authorized hosts placed after an exported file system must be separated by space characters. Options for each of the hosts must be placed in parentheses directly after the host identifier, without any spaces separating the host and the first parenthesis.

A line for an exported file system has the following structure:

<Export> <Host1>(Options)

In this structure, replace <*export*> with the directory being exported, replace <*host1*> with the host or network to which the export is being shared, and replace <*options*> with the options for that host or network. Additional hosts can be specified in a space separated list.

The following methods can be used to specify host names:

- *single host* — Where one particular host is specified with a fully qualified domain name, hostname, or IP address.
- *wildcards* — Where a * or ? character is used to take into account a grouping of fully qualified domain names that match a particular string of letters. Wildcards should not be used with IP addresses; however, it is possible for them to work accidentally if reverse DNS lookups fail.

  Be careful when using wildcards with fully qualified domain names, as they tend to be more exact than expected. For example, the use of *.example.com as a wildcard allows sales.example.com to access an exported file system, but not bob.sales.example.com. To match both possibilities both *.example.com and *.*.example.com must be specified.

- *IP networks* — Allows the matching of hosts based on their IP addresses within a larger network. For example, 192.168.135.0/28 allows the first 16 IP addresses,

from 192.168.0.0 to 192.168.0.15, to access the exported file system, but not 192.168.0.16 and higher.

Eg:

/shared 192.168.135.162(ro, sync, wdelay)

In the example, 192.168.135.162 can mount /shared.

- ro — Mounts of the exported file system are read-only. Remote hosts are not able to make changes to the data shared on the file system. To allow hosts to make changes to the file system, the read/write (rw) option must be specified.
- wdelay — Causes the NFS server to delay writing to the disk if it suspects another write request is imminent. This can improve performance by reducing the number of times the disk must be accessed by separate write commands, reducing write overhead. The no_wdelay option turns off this feature, but is only available when using the sync option.
- root_squash — Prevents root users connected remotely from having root privileges and assigns them the user ID for the user nfsnobody. This effectively "squashes" the power of the remote root user to the lowest local user, preventing unauthorized alteration of files on the remote server. Alternatively, the no_root_squash option turns off root squashing. To squash every remote user, including root, use the all_squash option
- Sync- By default, all but the most recent version (version 1.11) of the exportfs command will use async behavior, telling a client machine that a file write is complete - that is, has been written to stable storage - when NFS has finished handing the write over to the filesystem. This behavior may cause data corruption if the server reboots, and the **sync** option prevents this.

The /etc/hosts.allow and /etc/hosts.deny

These two files specify which computers on the network can use services on our machine. Each line of the file contains a single entry listing a service and a set of machines. When the server gets a request from a machine, it does the following:

1. It first checks hosts.allow to see if the machine matches a rule listed here. If it does, then the machine is allowed access.
2. If the machine does not match an entry in hosts.allow the server then checks hosts.deny to see if the client matches a rule listed there. If it does then the machine is denied access.
3. If the client matches no listings in either file, then it is allowed access.

We could add the following entry to */etc/hosts.allow*:

portmap: 192.168.135.162

For recent nfs-utils versions, we would also add the following (again, these entries are harmless even if they are not supported):

lockd: 192.168.135.162
rquotad: 192.168.135.162
mountd: 192.168.135.162
statd: 192.168.135.162

Starting Services:

service portmap start

service nfs start

Verifying that NFS is running:

exportfs –v It dispalys a list of shared directories and options on the server
exportfs –r It refreshes the servers share list after modifying /etc/exports

NFS Client Configuration Files:

Step1: Make directory where the remote file system is to be mounted:

mkdir /mnt/localshare

Step2: There are two ways of mounting NFS:

1) NFS shares are mounted on the client side using the mount command. The format of the command is as follows:

mount -t <*nfs-type*> <*host*>:<*/shared*> <*/mnt/ localshare*>

Replace <*nfs-type*> with nfs. Replace <*host*> with the remote host, <*/shared*> with the remote directory being mounted, and <*/mnt/localshare*> with the local directory where the remote file system is to be mounted.

Note: A localshare folder must be created where the file system has to be mounted on a client server.

2) NFS File Systems to be Mounted at Boot Time :

The /etc/fstab file is referenced by the netfs service at boot time, so lines referencing NFS shares have the same effect as manually typing the mount command during the boot process.

A sample /etc/fstab line to mount an NFS export looks like the following example:

<*Server*>:<*/shared*> <*/mnt/localshare*> <*nfs-type*> *defaults* 0 0

Eg:

192.168.135.7:/shared /mnt/localshare nfs defaults 0 0

Replace <*server*> with the hostname, IP address, or fully qualified domain name of the server exporting the file system.

Replace <*/shared*> with the path to the exported directory.

Replace <*/mnt/localshare*> with the local file system on which the exported directory is mounted. This mount point must exist before /etc/fstab is read or the mount fails.

Replace <*nfs-type*> with either nfs for NFSv2 or NFSv3 servers, or nfs4 for NFSv4 servers.

Practical Uses of NFS:

Some of the more common ones are listed below:

- Set several machines to share a CDROM or other media among them. This is cheaper and often a more convenient method to install software on multiple machines.
- On large networks, it might be more convenient to configure a central NFS server in which to store all the user home directories. These home directories can then be exported to the network so that users would always have the same home directory, regardless of which workstation they log in to.
- Several machines could have a common /usr/ports/distfiles directory. That way, when we need to install a port on several machines, we can quickly access the source without downloading it on each machine.

NOTE:

- *Service portmap should be started, i.e < # service portmap start > because NFS server is an RPC service and thus requires portmap*

- *Stop the iptables services, < # service iptables stop > or configure a rule in iptables*

5.2 SAMBA SERVER:

Samba is a suite of utilities that allows the Linux box to share files and other resources, such as printers, with Windows boxes. This chapter describes how we can make the Linux box into a Windows Primary Domain Controller (PDC) or a server for a Windows Workgroup. Either configuration will allow everyone at home to have:

- their own logins on all the home windows boxes while having their files on the Linux box appear to be located on a new Windows drive
- shared access to printers on the Linux box
- shared files accessible only to members of their Linux user group.

Difference between a PDC and Windows Workgroup member:

- A PDC stores the login information in a central database on its hard drive. This allows each user to have a universal username and password when logging in from all PCs on the network.
- In a Windows Workgroup, each PC stores the usernames and passwords locally so that they are unique for each PC.

How to Get Samba Started:

We can configure Samba to start at boot time using the chkconfig command:

[root@bigboy tmp]# *chkconfig smb on*

We can start/stop/restart Samba after boot time using the smb initialization script as in the examples below:

[root@bigboy tmp]# *service smb start*

[root@bigboy tmp]# *service smb stop*

[root@bigboy tmp]# *service smb restart*

Note:

Unlike many Linux packages, Samba does not need to be restarted after changes have been made to its configuration file, as it is read after the receipt of every client request.

Configuring File and Directory Sharing:

i) #vi /etc/samba/smb.conf

```
        # Share User1 Home directory

        comment  = This is user1 Home directory

        path = /home/nitish

        writable = yes

        Browseable = yes

# Common share among two users

        comment  = This is user1 and user2 shared directory

        valid users = user1 user2

        path = /shared

        writable = yes

        Browseable = yes
```

The [printers] Share Section:

Samba has special shares just for printers, and these are configured in the [printers] section of SWAT. There is also a share under [printers] called printers which governs common printer settings. Print shares always have the printable parameter set to yes. The default smb.conf [printers] share section looks like this:

```
[printers]

   comment = All Printers

   path = /var/spool/samba

   printable = Yes

   browseable = No

   create mask = 0640

  directory mask = 0750
```

ii) To create the user, use the command:

useradd nitish

Give them a Linux Password

passwd nitish

Changing password for user nitish.

New password:

Retype new password:

passwd: all authentication tokens updated successfully.

iii) Mapping The Linux Users To An smbpassword:

Next, we need to create Samba domain login passwords for the user

[root@bigboy tmp]# smbpasswd -a nitish password

The -a switch adds the user nitish to **the** /etc/smbpasswd file. Use a generic password then have users change it immediately from their workstations in the usual way.

Remember the smbpasswd sets the Windows Domain login password for a user, which is different from the Linux login password to log into the Samba box.

Samba Passwords

We should be aware that Linux password and Samba passwords are stored in two different locations. This provides the Samba administer the flexibility of allowing only some of the Linux users to have Samba accounts.

Use the passwd command to change Linux passwords, which are stored in the /etc/shadow file. Samba passwords are stored in the /etc/samba/smbpasswd file and can be changed smbpasswd command.

Note:

1) By default, the home directories are exported read only. Change next parameter to "no" if we want to be able to write to them.
read only = no

2)File creation mask is set to 0700 for security reasons. If we want to create files with group=rw permissions, set next parameter to 0775. create mask = 0755

3)Directory creation mask is set to 0700 for security reasons. If we want to create dirs. with group=rw permissions, set next parameter to 0775. directory mask = 0755

4)Sticky bit: A sticky bit is added to the shared directory so as to avoid deletion of files created by other users.

5.3 DHCP Server

DHCP(Dynamic Host Configuration Protocol), provides a method for hosts on the network to request, and be granted, configuration information including the addresses of routers and name servers. Usually there is single DHCP server per network segment, but in some cases there may be more than one. DHCP forwarding agents allow clients to receive addresses from a server which is not located on the same network segment.

IP address are either dynamically assigned from a range or pool of addresses, or statically assigned by MAC address. The assignments are made for a configurable amount of time (termed as 'lease' period) and may be renewed by the client. The server may be configured to accept request from only a specific set of MAC addresses, if desired.

Typically, the server will supply information about the network's subnet address and netmask, its default gateway, domain name and DNS server, and location of kickstart configuration files.

Server Side Configuration:

To configure a DHCP server, the /etc/dhcpd.conf configuration file must be created. A sample file can be found at /usr/share/doc/dhcp-<*version*>/dhcpd.conf.sample.

DHCP also uses the file /var/lib/dhcp/dhcpd.leases to store the client lease database.

The /etc/dhcpd.conf File

The first step in configuring a DHCP server is to create the configuration file that stores the network information for the clients. Global options can be declared for all clients, while other options can be declared for individual client systems.

The configuration file can contain extra tabs or blank lines for easier formatting. Keywords are case-insensitive and lines beginning with a hash mark (#) are considered comments.

There are two types of statements in the configuration file:

- Parameters — State how to perform a task, whether to perform a task, or what network configuration options to send to the client.
- Declarations — Describe the topology of the network, describe the clients, provide addresses for the clients, or apply a group of parameters to a group of declarations.

Some parameters must start with the option keyword and are referred to as options. Options configure DHCP options; whereas, parameters configure values that are not optional or control how the DHCP server behaves.

Parameters (including options) declared before a section enclosed in curly brackets ({ }) are considered global parameters. Global parameters apply to all the sections below it.

Note: If the configuration file is changed, the changes do not take effect until the DHCP daemon is restarted with the command service dhcpd restart.

In this example, there are global options for every DHCP client in the subnet and a range declared. Clients are assigned an IP address within the range.

```
subnet 192.168.135.0 netmask 255.255.255.0 {
      option routers                  192.168.135.1;
      option subnet-mask              255.255.255.0;

      option domain-name              "nitishkr.com";
      option domain-name-servers      192.168.135.1;

      option time-offset              -18000;

           range 192.168.135.10 192.168.135.20;

           host compcell {
                        hardware ethernet 00:A0:78:8E:9E:AA;
                        fixed-address 192.168.135.20;
                       }
     }
```

In this configuration, the routers, subnet-mask, domain-name, domain-name-servers, and time-offset options are used for any host statements declared below it.

Additionally, a subnet can be declared, a subnet declaration must be included for every subnet in the network. If it is not, the DHCP server fails to start.

In this example, there are global options for every DHCP client in the subnet and a range declared. Clients are assigned an IP address within the range.

Range Parameter: To assign an IP address to a client based on the MAC address of the network interface card, use the hardware ethernet parameter within a host declaration. As stated, the host compcell declaration specifies that the network interface card with the MAC address 00:A0:78:8E:9E:AA always receives the IP address 192.168.135.20.

Lease Database

On the DHCP server, the file /var/lib/dhcp/dhcpd.leases stores the DHCP client lease database. This file should not be modified by hand. DHCP lease information for each recently assigned IP address is automatically stored in the lease database. The information includes the length of the lease, to whom the IP address has been assigned, the start and end dates for the lease, and the MAC address of the network interface card that was used to retrieve the lease.

The lease database is recreated from time to time so that it is not too large. First, all known leases are saved in a temporary lease database. The dhcpd.leases file is renamed dhcpd.leases~ and the temporary lease database is written to dhcpd.leases.

The DHCP daemon could be killed or the system could crash after the lease database has been renamed to the backup file but before the new file has been written. If this happens, the dhcpd.leases file does not exist, but it is required to start the service. Do not create a new lease file. If we do, all old leases are lost which causes many problems. The correct solution is to rename the dhcpd.leases~ backup file to dhcpd.leases and then start the daemon.

Note: When the DHCP server is started for the first time, it fails unless the dhcpd.leases file exists. Use the command touch /var/lib/dhcp/dhcpd.leases to create the file if it does not exist.

If the same server is also running BIND as a DNS server, this step is not necessary, as starting the named service automatically checks for a dhcpd.leases file.

Starting The DHCP server:
#service dhcpd start

Configuring a DHCP Linux Client

To configure a DHCP client manually, modify the /etc/sysconfig/network file to enable networking and the configuration file for each network device in the /etc/sysconfig/network-scripts directory. In this directory, each device should have a configuration file named ifcfg-eth0, where eth0 is the network device name.

The /etc/sysconfig/network file should contain the following line:

NETWORKING=yes

The NETWORKING variable must be set to yes if we want networking to start at boot time.

The /etc/sysconfig/network-scripts/ifcfg-eth0 file should contain the following lines:

DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes

Configuring Windows Clients to Use DHCP

Fortunately Windows defaults to using DHCP for all its NIC cards so we don't have to worry about doing any reconfiguration.

5.4 Kickstart Installations:

Many system administrators would prefer to use an automated installation method to install Red Hat Enterprise Linux on their machines. To answer this need, Red Hat created the kickstart installation method. Using kickstart, a system administrator can create a single file containing the answer to all the questions that would normally be asked during a typical installation.

Kickstart files can be kept on a single server system and read by individual computers during the installation. This installation method can support the use of a single kickstart file to install Red Hat Enterprise Linux on multiple machines, making it ideal for network and system administrators.

Kickstart provides a way for users to automate a Red Hat Enterprise Linux installation.

Performing a Kickstart Installation:

Kickstart installations can be performed using a local CD-ROM, a local hard drive, or via NFS, FTP, or HTTP.

To use kickstart, we must:

1. Create a kickstart file via GUI of linux (ks.cfg)
2. Create a boot media with the kickstart file or make the kickstart file available on the network.
3. Make the installation tree available.
4. Start the kickstart installation.

This chapter explains these steps in detail.

Making the Kickstart File Available

A kickstart file must be placed in one of the following locations:

- On a boot diskette
- On a boot CD-ROM
- On a network

Normally a kickstart file is copied to the boot diskette, or made available on the network. The network-based approach is most commonly used, as most kickstart installations tend to be performed on networked computers.

Making the Kickstart File Available on the Network

Network installations using kickstart are quite common, because system administrators can easily automate the installation on many networked computers quickly and painlessly. In general, the approach most commonly used is for the administrator to have both a BOOTP/DHCP server and an NFS server on the local network. The BOOTP/DHCP server is used to give the client system its networking information, while the actual files used during the installation are served by the NFS server. Often, these two servers run on the same physical machine, but they are not required to.

To perform a network-based kickstart installation, we must have a BOOTP/DHCP server on our network, and it must include configuration information for the machine on which we are attempting to install Red Hat Enterprise Linux. The BOOTP/DHCP server provides the client with its networking information as well as the location of the kickstart file.

If a kickstart file is specified by the BOOTP/DHCP server, the client system attempts an NFS mount of the file's path, and copies the specified file to the client, using it as the kickstart file. The exact settings required vary depending on the BOOTP/DHCP server we use.

Making the Installation Tree Available

The kickstart installation must access an installation tree. An installation tree is a copy of the binary Red Hat Enterprise Linux CD-ROMs with the same directory structure.

If we are performing a CD-based installation, insert the Red Hat Enterprise Linux CD-ROM #1 into the computer before starting the kickstart installation.

If we are performing a hard drive installation, make sure the ISO images of the binary Red Hat Enterprise Linux CD-ROMs are on a hard drive in the computer.

If we are performing a network-based (NFS, FTP, or HTTP) installation, we must make the installation tree available over the network.

Step1: Make a directory in server which we will share using NFS for kickstart installation of linux Via NFS:

#mkdir /share

Step2: Copy RedHat directory and .discinfo file to this /shared folder from RedHat CD#1

#cp /mnt/media/cdrom/RedHat/*.* /shared/

And, #cp/mnt/media/cdrom/.discinfo /shared/

Step3: Copy RPMS folders to the /shared/RedHat/RPMS folders from redHat CD2,3,4

#cp /mnt/media/cdrom/RedHat/RPMS/*.* /shared/RedHat/RPMS

Step4: Copt The Kickstart configuration file to this /shared folder

#cp /ks.cfg /shared/

Step5: now share this folder /shared in the server via NFS so as to make it available to the clients.

Starting a Kickstart Installation

To begin a kickstart installation, we must boot the system from boot media we have made or the Red Hat Enterprise Linux CD-ROM #1, and enter a special boot command at the boot prompt. The installation program looks for a kickstart file if the ks command line argument is passed to the kernel.

The command at boot prompt is:
#ks=nfs:192.168.135.7:/shared/ks.cfg ksdevice=eth0

ks=nfs:<*server*>:/<*path*>

> The installation program looks for the kickstart file on the NFS server <*server*>, as file <*path*>. The installation program uses DHCP to configure the Ethernet card. For example, if our NFS server is server.ihbt.res.in and the kickstart file is in the NFS share /shared/ks.cfg, the correct boot command would be ks=nfs:server.ihbt.res.in:/shared/ks.cfg.

ksdevice=<*device*>

> The installation program uses this network device to connect to the network. For example, to start a kickstart installation with the kickstart file on an NFS server that is connected to the system through the eth1 device, use the command ks=nfs:<*server*>:/<*path*> ksdevice=eth0 at the boot: prompt.

Note: NFS and DHCP server must be configured and started for proper kickstart installation.

If a DHCP server is not present then the command will like this:

#ks=nfs:192.168.135.7:/shared/ks.cfg ksdevice=eth0 ip:192.168.135.162
netmask=255.255.255.0 gateway=192.168.135.1

5.5 SQUID SERVER:

It is an Internet object cache that can act as a proxy server for HTTP, FTP and other requests. Clients request URL's(Uniform Resource Locators) from squid, which then either serves cached copies of the URL's if they have been previously requested. URL's associated with dynamic content get forwarded, rather than being served out of the cache.

Assignment. Configure a proxy server using Squid such that we have two subnets of the range 192.168.135.0 and 192.168.205.0 and configure squid server so that clients from both the network can surf internet through it, also restrict 192.168.205.0 subnet from accessing internet in office hours (9:00 AM - 5:30 PM) and also configure acl's for restricting users to visit undesired sites

Confiuration files:

The /etc/squid/squid.conf File

The main Squid configuration file is squid.conf, and, like most Linux applications, Squid needs to be restarted for changes to the configuration file can take effect.

i) The Visible Host Name

Squid will fail to start if we don't give the server a hostname. We can set this with the visible_hostname parameter. Here, the hostname is set to the real name of the server compcell**.**

visible_hostname compcell

ii) Restricting Web Access By IP Address

We can create an access control list that restricts Web access to users on certain networks. In this case, it's an ACL that defines a home network of 192.168.135.0.

Add this to the bottom of the ACL section of squid.conf

acl office_network src 192.168.135.0/255.255.255.0

acl home_network src 192.168.205.0/255.255.255.0

We also have to add a corresponding http_access statement that allows traffic that matches the ACL:

Add this at the top of the http_access section of squid.conf

http_access allow office_network

http_access allow home_network

http_access deny all

Access Control Lists:

We can limit users' ability to browse the Internet with **access control lists** (**ACLs**). Each ACL line defines a particular type of activity, such as an access time or source network, they are then linked to an http_access statement that tells Squid whether or not to deny or allow traffic that matches the ACL.

Squid matches each Web access request it receives by checking the http_access list from top to bottom. If it finds a match, it enforces the allow or deny statement and stops reading further. We have to be careful not to place a deny statement in the list that blocks a similar allow statement below it. The final http_access statement denies everything, so it is best to place new http_access statements above it

Note: The very last http_access statement in the squid.conf file denies all access. We therefore have to add the specific permit statements above this line. In the chapter's examples, I've suggested that we place the statements at the top of the http_access list for the sake of manageability, but we can put them anywhere in the section above that last line.

Squid has a minimum required set of ACL statements in the ACCESS_CONTROL section of the squid.conf file. It is best to put new customized entries right after this list to make the file easier to read.

iii) Restricting Web Access By Time

We can create access control lists with time parameters. For example, We can deny only business hour access from the home network.

Add this to the bottom of the ACL section of squid.conf

acl home_network src 192.168.205.0/24

acl business_hours time M T W H F 9:00-17:30

Add this at the top of the http_access section of squid.conf

http_access deny home_network business_hours

Or, we can allow morning access only:

Add this to the bottom of the ACL section of squid.conf

acl mornings time 08:00-12:00

Add this at the top of the http_access section of squid.conf

http_access allow mornings

iv) Preventing users from accessing “cooking” word:

acl COOKING url-regex cooking

http_access deny home_network COOKING

Manually Configuring Web Browsers To Use The Squid Server on client side:

If we don't have a firewall that supports redirection, then we need to configure the firewall to only accept HTTP Internet access from the Squid server, as well as configure the PC browser's proxy server settings manually to use the Squid server. The method we use depends on the browser.

For example, to make these changes using Internet Explorer

1. Click on the "Tools" item on the menu bar of the browser.
2. Click on "Internet Options"
3. Click on "Connections"
4. Click on "LAN Settings"
5. Configure with the address and TCP port (3128 default) used by the Squid server.

Here's how to make the same changes using Mozilla or Firefox.

1. Click on the "Edit" item on the browser's menu bar.
2. Click on "Preferences"
3. Click on "Advanced"
4. Click on "Proxies"
5. Configure with the address and TCP port (3128 default) used by the Squid server

v) Restart the services:

service squid restart

Squid logs:

Logs of squid can be seen in the directory /var/log/squid/access.log

The command is:

#tail –f /var/log/squid/access.log

5.6 DNS (Domain Name Server):

Working

- It resolves hostnames into IP addresses this is called forward lookup.
- It resolves IP addresses into hostname. This is called reverse lookup.
- It provides E-mail routing information

Name Resolution is a act of determining the ip address of a given hostname.

DNS Domains

Everyone in the world has a first name and a last, or family, name. The same thing is true in the DNS world: A family of Web sites can be loosely described a domain. For example, the domain ihbt.res.in has a number of children, such as www.ihbt.res.in and mail.ihbt.res.in for the Web and mail servers, respectively.

BIND

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.

DNS Clients

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. We shouldn't need to configure any other files.

Authoritative DNS Servers

Authoritative servers provide the definitive information for our DNS domain, such as the names of servers and Web sites in it. They are the last word in information related to our domain.

How DNS Servers Find Out Site Information

There are 13 root authoritative DNS servers (super duper authorities) that all DNS servers query first. These root servers know all the authoritative DNS servers for all the main domains - .com, .net, and the rest. This layer of servers keeps track of all the DNS servers that Web site systems administrators have assigned for their sub domains.

For example, when we register our domain ihbt.res.in, we are actually inserting a record on the .in DNS servers that point to the authoritative DNS servers we assigned for our domain.

How it works

A DNS server is just a computer that's running DNS software. Since most servers are Unix machines, the most popular program is BIND (Berkeley Internet Name Domain), but we can find software for the Mac and the PC as well.

DNS software is generally made up of two elements: the actual name server, and something called a resolver. The name server responds to browser requests by supplying name-to-address conversions. When it doesn't know the answer, the resolver will ask another name server for the information.

To see how it works, let's go back to the domain-name-space inverted tree.

When we type in a URL, our browser sends a request to the closest name server. If that server has ever fielded a request for the same host name (within a time period set by the administrator to prevent passing old information), it will locate the information in its cache and reply.

If the name server is unfamiliar with the domain name, the resolver will attempt to "solve" the problem by asking a server farther up the tree. If that doesn't work, the second server will ask yet another - until it finds one that knows. (When a server can supply an answer without asking another, it's known as an authoritative server.)

Once the information is located, it's passed back to our browser, and we're sent on our merry way. Usually this process occurs quickly, but occasionally it can take an excruciatingly long time (like 15 seconds). In the worst cases, we'll get a dialog box that says the domain name doesn't exist - even though we know damn well it does.

This happens because the authoritative server is slow replying to the first, and our computer gets tired of waiting so it times-out (drops the connection). But if we try again, there's a good chance it will work, because the authoritative server has had enough time to reply, and our name server has stored the information in its cache.

BENEFITS :-- Domain names can be logically and easily remembered

Domain names are separated by dots with the top most element on right. Each element may be upto 63 characters long the entire name may 255 characters long. Letters numbers dashes may be used in an element. The right most element of the domain is called the Top Level Domain. Hostnames map to IP addresses in many to many relations.

DNS servers can be masters or slaves sometimes referred to as primary or secondary servers

BIND:-- It is the most widely used DNS server.

Assignment: Configure a DNS server for the domain ihbt.res.in against international ip address 210.212.55.215, also configure a email server for the domain ihbt.res.in and a web server for the same domain, configure the servers and test them.

HOW TO CONFIGURE DNS SERVER:

STEP1:

In /etc/sysconfig/named

check this line:

ROOTDIR=/var/named/chroot

This is applicable only in upper version of Red hat linux i.e in RHEL4

All configuration files resides in this directory

Forward Zone File References in named.conf:

In this assignment the zone file is named ihbt.res.in.zone, and, although not explicitly stated, the file ihbt.res.in.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.

#cd /var/named/chroot/etc

#vi /etc/named.conf

```
zone "ihbt.res.in" IN {
type master;
file " ihbt.res.in.zone";
};
```

Reverse Zone File References in named.conf

Here's how to format entries that refer to zone files used for reverse lookups for our IP addresses.

The forward domain lookup process for ihbt.res.in scans the FQDN from right to left to get to get increasingly more specific information about the authoritative servers to use. Reverse lookups operate similarly by scanning an IP address from left to right to get increasingly specific information about an address.

The similarity in both methods is that increasingly specific information is sought, but the noticeable difference is that for forward lookups the scan is from right to left, and for reverse lookups the scan is from left to right. This difference can be seen in the formatting of the zone statement for a reverse zone in /var/named/chroot/etc/named.conf file where the main in-addr.arpa domain, to which all IP addresses belong, is followed by the first 3 octets of the IP address in reverse order. This order is important to remember or else the configuration will fail. This reverse zone definition for named.conf uses a reverse zone file named 55.212.210.zone:

```
zone “55.212.210.in-addr.arpa” IN {
 type master;
file”55.212.210. in-addr.arpa.zone”;
};
```

STEP2: Change to directory /var/named/chroot/var and create files for forward and reverse lookups

#cd **/var/named/chroot/var**

#cp localhost.zone ihbt.res.in.zone

#cp named.local 55.212.210.in-addr.arpa.zone

STEP3: Configure file /etc/var/ihbt.res.in.zone for forward lookups

vi ihbt.res.in.zone

```
ID      IN        SOA         ns1.ihbt.res.in.      root.ihbt.res.in. (
                                    13 ; serial
                                     2880 ; refresh
                                     7200 ; retry
                                     604800 ; expire
                                    86400 ; ttl
                                      )
@       ID      IN      NS        ns1.ihbt.res.in.
@       ID      IN      MX        5     ns1.ihbt.res.in.
@       ID       IN       A         210.212.55.215
```

In this file some of the parameters are explained as below:

- Server ns1.ihbt.res.in is the name server for ihbt.res.in. In corporate environments there may be a separate name server for this purpose. Primary name servers are more commonly called ns1 and secondary name servers ns2.
- A record maps host to IP address
- The SOA designates the beginning of zone's data and sets default parameters for that zone. It is normally the forst resource record in the zone databse

- The minimum TTL value ($TTL) is three days, therefore remote DNS caching servers will store learned DNS information from our zone for three days before flushing it out of their caches.

- The MX record for ihbt.res.in points to the server named mail.ihbt.res.in and this server has the IP address 210.212.55.216.

- ns1 is actually a CNAME or alias for the Web server www. So here we have an example of the name server, and Web server being the same machine. If they were all different machines, then we'd have an A record entry for each.

*Note: Always remember to put **dot(.)** after fully qualified domain name*

STEP4: Configure file /etc/var/55.212.210.in-addr.arpa.zone for forward lookups

#vi 55.212.210. in-addr.arpa.zone

```
@     IN      SOA      ihbt.res.in. root. ihbt.res.in. (
                                  13 ; serial
                                  2880 ; refresh
                                  7200 ; retry
                                  604800 ; expire
                                 86400 ; ttl
                                  )
@     IN      NS       ihbt.res.in.
215   IN      PTR      ihbt.res.in.
```

*A PTR record maps IP addresses to host names.

Note: Always remember to put **dot(.)** after fully qualified domain name

Configuration on server network settings:

1) #vi /etc/host.conf (Contains the order of search)

order bind,host

2) #vi /etc/resolv.conf (Contains the name of the doamin to search and nameserver information)

domain ihbt.res.in

search ihbt.res.in

nameserver 218.248.240.79

4) #Vi /etc/hosts (Contains the entrry of Domain server)

127.0.0.1 localhost.localdomain localhost

210.212.55.215 ihbt.res.in www

Checking for the DNS Server:

The Host Command

The host command accepts arguments that are either the fully qualified domain name or the IP address of the server when providing results. To perform a forward lookup, use the syntax:

host www.ihbt.res.in
www.ihbt.res.in has address 210.212.55.215

To perform a reverse lookup

host 210.212.55.215

55.212.210.in-addr.arpa domain name pointer 65-115-71-34.myisp.net.

As we can see, the forward and reverse entries don't match. The reverse entry matches the entry of the ISP.

The nslookup Command

The nslookup command provides the same results on Windows PCs. To perform forward lookup, use.

#nslookup www.ihbt.res.in
Server: 55.212.210.in-addr.arpa
Address: 210.212.55.215

Non-authoritative ansour:
Name: www.ihbt.res.in
Address: 210.212.55.215

To perform a reverse lookup

nslookup 210.212.55.215
Server: 55.212.210.in-addr.arpa
Name: ns1.ihbt.res.in
Address: 210.212.55.215

The Dig command

#dig –x 210.212.55.215

To Confirm service at startup:

#ntsysv

#chkconfig –level 3 5 named on

#init6 (reboot the system)

5.7 Sendmail:

Email is an important part of any Web site we create. In a home environment, a free web based email service may be sufficient, but if we are running a business, then a dedicated mail server will probably be required.

We will use sendmail to create a mail server that will relay our mail to a remote user's mailbox or incoming mail to a local mail box. We'll also learn how to retrieve and send mail via the mail server using a with mail client such as Outlook Express or Evolution.

How Sendmail Works

As stated before, sendmail can handle both incoming and outgoing mail for the domain. Take a closer look.

Incoming Mail

Usually each user in our home has a regular Linux account on our mail server. Mail sent to each of these users (username@ihbt.res.in) eventually arrives at the mail server and sendmail then processes it and deposits it in the mailbox file of the user's Linux account.

Mail isn't actually sent directly to the user's PC. Users retrieve their mail from the mail server using client software, such as Microsoft's Outlook or Outlook Express, that supports either the POP or IMAP mail retrieval protocols.

Linux users logged into the mail server can read their mail directly using a text-based client, such as mail, or a GUI client, such as Evolution. Linux workstation users can use the same programs to access their mail remotely.

Outgoing Mail

The process is different when sending mail via the mail server. PC and Linux workstation users configure their e-mail software to make the mail server their outbound SMTP mail server.

If the mail is destined for a local user in the mysite.com domain, then sendmail places the message in that person's mailbox so that they can retrieve it using one of the methods above.

If the mail is being sent to another domain, sendmail first uses DNS to get the MX record for the other domain. It then attempts to relay the mail to the appropriate destination mail server using the Simple Mail Transport Protocol (SMTP). One of the main advantages of mail relaying is that when a PC user A sends mail to user B on the Internet, the PC of user A can delegate the SMTP processing to the mail server.

Note:

> *If mail relaying is not configured properly, then the mail server could be commandeered to relay spam.*

Sendmail Configuration:

Configuring DNS for sendmail

Remember that we will never receive mail unless we have configured DNS for the domain to make our new Linux box mail server the target of the DNS domain's MX record.

1) The # /etc/mail/sendmail.mc File:

We can define most of sendmail's configuration parameters in the /etc/mail/sendmail.mc file, which is then used by the m4 macros to create the /etc/mail/sendmail.cf file. (For example, # m4 /etc/mail/sendmail.mc > /etc/mail/sendmail.cf). Configuration of the sendmail.mc file is much simpler than configuration of sendmail.cf, but it is still often viewed as an intimidating task with its series of structured **directive** statements that get the job done. Fortunately, in most cases we won't have to edit this file very often.

How to Put Comments in sendmal.mc

In most Linux configuration files a # symbol is used at the beginning of a line convert it into a comment line or to deactivate any commands that may reside on that line.

The sendmail.mc file doesn't use this character for commenting, but instead uses the string "dnl". Here are some valid examples of comments used with the sendmail.mc configuration file:

These statements are disabled by dnl commenting.

dnl DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')

dnl # DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')

This statement is incorrectly disabled:

DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')

This statement is active:

DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')

#vi /etc/mail/sendmail.mc

dnl This changes sendmail to only listen on the loopback device 127.0.0.1
dnl and not on any other network devices. Comment this out if we want
dnl to accept email over the network.
dnl DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')

ADD another server which is email server:

DAEMON_OPTIONS(`Port=smtp,Addr=210.212.55.215, Name=MTA')

Then save the file and exit, now create sendmail.cf file with m4 macro:

#m4 /etc/mail/sendmail.mc > /etc/mail/sendmail.cf

or with another command:

#makemap hash /etc/mail/sendmail.cf < /etc/mail/sendmail.mc

Note: We have to configure /etc/dovecot.conf and start its service in RHEL4:

#vi /etc/dovecot.conf

protocols = imap pop3

#service dovecot restart

The /etc/mail/access File:

We can make sure that only trusted PCs on the network have the ability to relay mail via the mail server by using the /etc/mail/access file. That is to say, the mail server will relay mail only for those PCs on the network that have their e-mail clients configured to use the mail server as their outgoing SMTP mail server. (In Outlook Express, we set this using: Tools>Accounts>Properties>Servers)

If we don't take the precaution of using this feature, we may find the server being used to relay mail for spam e-mail sites. Configuring the /etc/mail/access file will not stop spam coming to we, only spam flowing through we.

The /etc/mail/access file has two columns. The first lists IP addresses and domains from which the mail is coming or going. The second lists the type of action to be taken when mail from these sources or destinations is received. Keywords include RELAY, REJECT, OK (not ACCEPT), and DISCARD. There is no third column to state whether the IP address or domain is the source or destination of the mail, sendmail assumes it could be either and tries to match both. All other attempted relayed mail that doesn't match any of the entries in the /etc/mail/access file, sendmail will reject. Despite this, my experience

has been that control on a per e-mail address basis is much more intuitive via the /etc/mail/virtusertable file.

The sample file that follows allows relaying for only the server itself (127.0.0.1, localhost), two client PCs on the home 192.168.1.X network, everyone on the 192.168.2.X network, and everyone passing e-mail through the mail server from servers belonging to ihbt.res.in. Remember that a server will be considered a part of ihbt.res.in only if its IP address can be found in a DNS reverse zone file:

localhost.localdomain	RELAY
localhost	RELAY
127.0.0.1	RELAY
210.212.55.215	RELAY

We'll then have to convert this text file into a sendmail readable database file named /etc/mail/access.db. Here are the commands we need:

[root@bigboy tmp]# ***m4 /etc/mail/access > /etc/mail/access.db***

or with another command:

#makemap hash /etc/mail/access.db < /etc/mail/access

The /etc/mail/local-host-names File

When sendmail receives mail, it needs a way of determining whether it is responsible for the mail it receives. It uses the /etc/mail/local-host-names file to do this. This file has a list of hostnames and domains for which sendmail accepts responsibility. For example, if this mail server was to accept mail for the domains my-site.com and another-site then the file would look like this:

vi local-host-names

Add this line:

ihbt.res.in

Sendmail Masquerading:

If we want the mail to appear to come from user@ihbt.res.in and not user@ns1.ihbt.res.in, then we have two choices:

Configure the email client, such as Outlook Express, to set the email address to user@ihbt.res.in

Set up masquerading to modify the domain name of all traffic originating from and passing trough the mail server.

In the DNS configuration, we made ns1.ihbt.res.in the mail server for the domain ihbt.res.in. We now have to tell ns1.ihbt.res.in in the sendmail configuration file sendmail.mc that all outgoing mail originating on ns1.ihbt.res.in should appear to be coming from ihbt.res.in;

We can solve this by editing the **sendmail.mc** configuration file and adding some masquerading commands and directives:

MASQUERADE_AS(`ihbt.res.in')dnl

Masquerading is an important part of any mail server configuration as it enables systems administrators to use multiple outbound mail servers, each providing only the global domain name for a company and not the fully qualified domain name of the server itself. All email correspondence then has a uniform email address format that complies with the company's brand marketing policies.

The /var/log/maillog File

Because sendmail writes all its status messages in the /var/log/maillog file, always monitor this file whenever we are doing changes.

tail -f /var/log/maillog

Fighting SPAM

Unsolicited Commercial Email (UCE or SPAM) can be annoying, time consuming to delete and in some cases dangerous when they contain viruses and worms. Fortunately there are ways we can use the mail server to combat SPAM.

Using Public SPAM Blacklists With Sendmail

There are many publicly available lists of known open mail relay servers and spam generating mail servers on the Internet. Some are maintained by volunteers, others are managed by public companies, but in all cases they rely heavily on complaints from spam victims. Some spam blacklists simply try to determine whether the e-mail is coming from a legitimate IP address.

The IP addresses of offenders usually remain on the list for six months to two years. In some cases, to provide additional pressure on the spammers, the blacklists include not only the offending IP address but also the entire subnet or network block to which it belongs. This prevents the spammers from easily switching their servers' IP addresses to the next available ones on their networks. Also, if the spammer uses a public data center, it is possible that their activities could also cause the IP addresses of legitimate e-mailers

to be black listed too. It is hoped that these legitimate users will pressure the data center's management to evict the spamming customer.

We can configure sendmail to use its dnsbl feature to both query these lists and reject the mail if a match is found. Here are some sample entries we can add to the /etc/sendmail.mc file; they should all be on one line.

RFC-Ignorant: A valid IP address checker.

An open proxy list:

FEATURE(`dnsbl', `ipwhois.rfc-ignorant.org',`"550 Mail from " $&{client_addr} " refused. Rejected for bad WHOIS info on IP of the SMTP server - see http://www.rfc-ignorant.org/"')

FEATURE(`dnsbl', `proxies.blackholes.easynet.nl', `"550 5.7.1 ACCESS DENIED to OPEN PROXY SERVER "$&{client_name}" by easynet.nl
DNSBLÂ Â (http://proxies.blackholes.easynet.nl/errors.html)"', `')dnl

FEATURE(`dnsbl', `relays.ordb.org', `"550 Email rejected due to sending server misconfiguration - see http://www.ordb.org/faq/\#why_rejected"')dnl

FEATURE(`dnsbl', `bl.spamcop.net', `"450 Mail from " $`'&{client_addr} " refused - see http://spamcop.net/bl.shtml"')

FEATURE(`dnsbl',`sbl.spamhaus.org',`Rejected - see http://spamhaus.org/')dnl

Starting Sendmail

We can use the chkconfig command to get sendmail configured to start at boot:

chkconfig sendmail on

To start, stop, and restart sendmail after booting, use

service sendmail start

Note:

Also start dovecot service

#service dovecot start

5.8 The Apache Web server:

Apache is a freely available Web server that is distributed under an "open source" license.

Configuring DNS For Apache:

Remember that we will never receive the correct traffic unless we configure DNS for our domain to make our new Linux box Web server the target of the DNS domain's www entry.

General Server Side Configuration Steps:

The configuration file used by Apache is /etc/httpd/conf/httpd.conf. As for most Linux applications, we must restart Apache before changes to this configuration file take effect.

Putting our Web Pages:

All the statements that define the features of each web site are grouped together inside their own <VirtualHost> section, or container, in the httpd.conf file. The most commonly used statements, or directives, inside a <VirtualHost> container are:

servername: Defines the name of the website managed by the **<VirtualHost>** container. This is needed in named virtual hosting only.

DocumentRoot: Defines the directory in which the web pages for the site can be found.

By default, Apache searches the DocumentRoot directory for an index, or home, page named index.html. So for example, if we have a servername of www.ihbt.res.in with a DocumentRoot directory of /var/www/html/, Apache displays the contents of the file /var/www/html/index.html when we enter http://www.ihbt.res.in in our browser.

The Default File Location:

By default, Apache expects to find all its web page files in the /var/www/html/ directory with a generic DocumentRoot statement at the beginning of httpd.conf. The examples in this chapter use the /home/www directory to illustrate how we can place them in other locations successfully.

Named Virtual Hosting:

We can make our Web server host more than one site per IP address by using Apache's named virtual hosting feature. We use the NameVirtualHost directive in the /etc/httpd/conf/httpd.conf file to tell Apache which IP addresses will participate in this feature.

The <VirtualHost> containers in the file then tell Apache where it should look for the Web pages used on each Web site. We must specify the IP address for which each <VirtualHost> container applies.

Named Virtual Hosting:

Consider an example in which the server is configured to provide content on 210.212.55.215. In the code that follows, notice that within each <VirtualHost> container we specify the primary Web site domain name for that IP address with the ServerName directive. The DocumentRoot directive defines the directory that contains the index page for that site.

We can also list secondary domain names that will serve the same content as the primary ServerName using the ServerAlias directive.

Apache searches for a perfect match of NameVirtualHost, <VirtualHost>, and ServerName when making a decision as to which content to send to the remote user's Web browser. If there is no match, then Apache uses the first <VirtualHost> in the list that matches the target IP address of the request.

This is why the first <VirtualHost> statement contains an asterisk: to indicate it should be used for all other Web queries.

NameVirtualHost 210.212.55.215

<VirtualHost 210.212.55.215>

servername ns1.ihbt.res.in

DocumentRoot /var/www/html

DirectoryIndex index.html

</VirtualHost>

Be careful with using the asterisk in other containers. A <VirtualHost> with a specific IP address always gets higher priority than a <VirtualHost> statement with an * intended to cover the same IP address, even if the ServerName directive doesn't match. To get consistent results, try to limit the use of our <VirtualHost *> statements to the beginning of the list to cover any other IP addresses our server may have.

Protecting Web Page Directories With Passwords:

We can password protect content in both the main and subdirectories of our DocumentRoot fairly easily. There is normal access to the regular Web pages, but sometimes we require passwords for directories or pages that show MRTG or Webalizer data. This configuration shows how to password protect the /home/www directory.

1) Use Apache's htpasswd password utility to create username/password combinations independent of our system login password for Web page access. We have to specify the location of the password file, and if it doesn't yet exist, we have to include a -c, or create, switch on the command line. I recommend placing the file in our /etc/httpd/conf

directory, away from the DocumentRoot tree where Web users could possibly view it. Here is an example for a first user named peter and a second named paul:

```
#htpasswd -c /etc/httpd/conf/.htpasswd nitish
New password:
Re-type new password:
Adding password for user peter

# htpasswd /etc/httpd/conf/.htpasswd paul
New password:
Re-type new password:
Adding password for user paul
```

2) Make the .htpasswd file readable by all users.

```
# chmod 644 /etc/httpd/conf/.htpasswd
```

3) Create a .htaccess file in the directory to which we want password control with these entries.

```
AuthUserFile /etc/httpd/conf/.htpasswd
AuthGroupFile /dev/null
AuthName EnterPassword
AuthType Basic
require user nitish
```

Remember this password protects the directory and all its subdirectories. The AuthUserFile tells Apache to use the .htpasswd file. The require user statement tells Apache that only user peter in the .htpasswd file should have access. If we want all .htpasswd users to have access, replace this line with require valid-user. AuthType Basic instructs Apache to accept basic unencrypted passwords from the remote users' Web browser.

4) Set the correct file protections on our new .htaccess file in the directory /home/www.

```
#chmod 644 /home/www/.htaccess
```

5) Make sure that our /etc/httpd/conf/http.conf file has an AllowOverride statement in a <Directory> directive for any directory in the tree above /home/www. In this configuration below, all directories below /var/www/ require password authorization.

```
<Directory /home/www/*>
  AllowOverride AuthConfig
</Directory>
```

To Get Apache Started:

Use the chkconfig command to configure Apache to start at boot:

chkconfig httpd on

service httpd start
service httpd stop
service httpd restart

We can test whether the Apache process is running with

pgrep httpd

We should get a response of plain old process ID numbers

The Apache Status Log Files:

The **/var/log/httpd/access_log** file is updated after every HTTP query and is a good source of general purpose information about our website. There is a fixed formatting style with each entry being separated by spaces or quotation marks.

tail –f /var/log/httpd/access_log

6. NETWORK SECURITY

Network security is a primary consideration in any decision to host a website as the threats are becoming more widespread and persistent every day. One means of providing additional protection is to invest in a firewall. Though prices are always falling, in some cases we may be able to create a comparable unit using the Linux iptables package on an existing server for little or no additional expenditure.

Here we will convert a Linux server into:

- A firewall while simultaneously being our home website's mail, web and DNS server.
- A router that will use **NAT** and **port forwarding** to both protect our home network and have another web server on our home network while sharing the public IP address of our firewall.

Creating an iptables firewall script requires many steps, but with the aid of the sample tutorials, we should be able to complete a configuration relatively quickly.

6.1 IP TABLES:-

IPTables is really a front end (user space) tool to manage Netfilter (integrated within the Linux kernel).
IPTables functions at OSI Layers 3 (Network(IP)) and 4(Transport(TCP,UDP)).
Layer 3 focuses on source address and destination address. IP addresses are based on 32-bit ranges (4 billions of addresses).
Layer 4 focuses on Protocols: Ports TCP:80, UDP:69. TCP/UDP ports use a 16-bit range (0-65535).

IPTables can manage ICMP
ICMP uses types: echo-request, echo-reply.

SOME BASIC COMMANDS:

Iptables –t table (action/direction) (Packet Pattern) –j (fate)
TABLES : filter(default), nat, mangle
ACTIONS : -A append, -D delete, -L list, -F flush
DIRECTION : INPUT, OUTPUT , FORWARD
PACKET PATTERN: -s Source IP address, -d Destination IP address.
FATE: DROP, ACCEPT, REJECT

EXAMPLES

The following command defines a rule that rejects all traffic from the 192.168.25.0 subnet, and it sends a `destination unreachable' error message back to any client that connect:

iptables –A INPUT –s 192.168.75.0/24 –j REJECT

This rule stops the users from the computer with n IP address 192.168.75.200 from pinging our system(remember that the ping command uses the ICMP protocol).

iptables –A INPUT –s 192.168.25.200 –p icmp –j DROP

The following command guards against TCP SYN attacks from outside network. Assume that your network IP address is 192.168.1.0. The exclamation point (!) inverts the meaning; in this case, the command applies to all IP addresses except to those with a 192.168.1.0 network addresses(and a 255.255.255.0 subnet mask).

iptables –A INPUT –s !192.168.1.0/24 –p tcp –j DROP

Then if we wanted to delete the rule related to the ping command in this list, use the following command

#iptables –D INPUT –s 192.168.25.200 –p icmp –j DROP

The default rule for INPUT , OUTPUT , and FORWARD is to accept all packets .One way to sot forwarding is to add the following rule:

iptables –A FORWARD –J DROP

Drop all outgoing network traffic. If possible, do not affect incoming traffic

#iptables –A OUTPUT –m state --state ESTABLISHED –j ACCEPT
#iptables –A OUTPUT –J REJECT

We want to block connections to a particular network service for example HTTP

#iptables –A INPUT –p tcp –dport www –j REJECT

We want to block incoming traffic from a particular host

#iptables –A INPUT –s remote_ip_address –j REJECT

To block requests from one particular service say SMTP mail service:

#iptables –A INPUT –p tcp –s remote_ip_address –dport smtp –j REJECT

We want to block the outoing traffic to a particular host
#iptables –A OUTPUT –d remote_ip_address –j REJECT

We want to prevent outgoing access to a network, e.g., all web servers at yahoo.com

#iptables –A OUTPUT –d remote_ip_address –j REJECT
#iptables –A output –d www.yahoo.com –j REJECT

We want to permit incoming SSH access but no other incoming access. Allow local connections to all services

#iptables –A INPUT p -- dport ssh –j ACCEPT
#iptables –A INPUT –I lo –j ACCEPT
#iptables –A INPUT –j REJECT
#iptables –A INPUT –p tcp –s 128.220.13.4 --dport ssh –j ACCEPT

We want to block outgoing telnet connections

#iptables –A OUTPUT –p tcp -- dport telnet –j REJECT

6.2 Using TCP Wrappers to secure Linux:

TCP Wrappers can be used to GRANT or DENY access to various services on our machine to the outside network or other machines on the same network. It does this by using simple Access List Rules which are included in the two files **/etc/hosts.allow** and **/etc/hosts.deny** .
Let us consider this scenario: A remote machine **remote_mc** trying to connect to our local machine **local_mc** using ssh.
When the request from the remote_mc is received by the tcp wrapped service (SSH in this case), it takes the following basic steps:

1. It checks the /etc/hosts.allow file and applies the first rule specified for that service. If it finds a matching rule , it allows the connection. If no rule is found, it moves on to step 2.
2. It checks the /etc/hosts.deny file and if a matching rule is found, it deny's the connection.

#FILE: /etc/hosts.deny
<Service name> : <IP address to be denied> or < Domain name to be denied>
vsftpd : 192.168.1. , .abc.co.

#FILE: /etc/hosts.allow
<Service name> : <IP address to be allowed> or < Domain name to be allowed>
in.telnetd : 192.168.1. , .abc.co.

BIBLIOGRAPHY

1. Computer Networks by S.Tanenbaum

2. Unix : The Complete Reference by Stephen Coffin

3. Learning Red Hat Enterprise Linux and Fedora by Bill McCarty

4. UNIX System Administrator's Bible by Yves Lepage and Paul Larrera