Linux Practicals

Index

Linux Practicals

Switching Terminals

Linux have 6 ttys by default, which we call as vcs & driver assigned to it is tty (/dev/tty*)
To - switch from gui use CTRL ALT F1 & to switch between the terminals use ALT -F2, F3...
To check current terminal use $ps command.

Basic Commands

```
$df -h      -> same as my computer in windows
$fdsik -l   -> list partition
$man <cmd>  -> manual
$clear      -> clear the screen
$^l         -> clear the screen
$ls         -> list content
$ls -l      -> list content in long listing format
$ls -al     -> list all subcontent in long listing format
$ll         -> an alias for the above
$ls -R      -> list content recursively
$l.         -> list hidden files
$ls -F      -> list content and classify them
$alias      -> display all aliases for current user
$alias <statement>  -> make alias eg alias c='clear'
$unalias <alias>    -> remove alias eg unalias c
$exit       -> log out from the system
$logout     -> log out from the system
$^d         -> log out from the system
$tree       -> list content in a tree (hierarchial) diagram
$tree -d    -> list subdirectories only - no files
$tree -p    -> list content with their permissions
$cd <directory>     -> change directory to...
$cd ..      -> change to parent directory
$cd -       -> change to previous directory
$cd         -> change to home directory
$cd ~       -> change to home directory
$pushd      -> change dir with pwd
$cat        -> display a content of a file
$pwd        -> print work (current) directory
$pwd -P     -> print parent working dir of this symlink dir
$mkdir <directory>  -> make directory
$mkdir -p <directory>   -> make parent directories also if it does
not exist
$touch      -> make a 0 byte file if it does not exist
$cp         -> copy (for files)
$cp -a      -> copy (for directories)
$cp -p      -> copy and preserve date and time
$mv         -> move OR rename
$rmdir      -> remove empty directory
$rm         -> remove         (for files)
$rm -f      -> remove forcefully  ( " " )
$rm -r      -> remove recursively       (for directories)
$rm -rf     -> remove recursively and forcefully ( " " )
$cat        -> display content of the file
$cat -n     -> display content of the file and number the lines
$cal        -> display calendar for current month
$date       -> display system date and time
$date -s '<value>' -> change system date and time in mm/dd/yy
$hwclock    -> display the hardware clock
$hwclock -hctosys  -> set the system time from the hardware clock
$ln -s      -> make a soft/sym/symbolic link
$ln         -> make a hard link
$history    -> display the list of the last 1000 commands
$! 100      -> Run command 100 in history
$vi         -> text editor
$vimtutor   -> vi manual withexercise
$pico       -> pico manual withexercise
$mcedit     -> mcedit manual withexercise
```

```
$joe         -> joe manual withexercise
$aspell -c <filename>   -> check the spelling in the file
$elinks      -> check the web links
$file        -> display the type of file
$which       -> display the path of the binary
$whereis     -> display all paths
$hostname    -> display system name with domain
$id          -> display id info of current user
$id -u       -> display user id of current user
$id -un      -> display username of current user
$id -g       -> display group id of current user
$id -gn      -> display groupname of current user
$uptime      -> display for how long the system has been running
$tty         -> display current terminal number
$users       -> display no. of users currently logged in
$whoami      -> display username of current user
$who         -> display users logged in the system with their
                respective terminals and time since logged in
$who am I    -> display current user, terminal and uptime
$w           -> display is details which files are open on which
                terminal
```

http://www.oraclehome.co.uk/linux-commands.htm

```
$mkdir -p /opt/funny/test
$cd /opt/funny/test -- absolute path
$cd /opt/funny
$pwd
/opt/funny
$cd test -- relative path
```

Fdisk

Partitioning with fdisk

This section shows you how to actually partition your hard drive with the **fdisk** utility. Linux allows only 4 primary partitions. You can have a much larger number of logical partitions by sub-dividing one of the primary partitions. Only one of the primary partitions can be sub-divided.

Examples:

1. Four primary partitions
2. Mixed primary and logical partitions

fdisk usage

fdisk is started by typing (as root) **fdisk *device*** at the command prompt. *device* might be something like /dev/hda or /dev/sda (see Section 2.1.1). The basic **fdisk** commands you need are:

p print the partition table

n create a new partition

d delete a partition

q quit without saving changes

w write the new partition table and exit

Changes you make to the partition table do not take effect until you issue the write (w) command. Here is a sample partition table:

```
Disk /dev/hdb: 64 heads, 63 sectors, 621 cylinders
Units = cylinders of 4032 * 512 bytes

 Device Boot Start End Blocks Id System
/dev/hdb1 * 1 184 370912+ 83 Linux
/dev/hdb2 185 368 370944 83 Linux
/dev/hdb3 369 552 370944 83 Linux
/dev/hdb4 553 621 139104 82 Linux swap
```

The first line shows the geometry of your hard drive. It may not be physically accurate, but you can accept it as though it were. The hard drive in this example is made of 32 double-sided platters with one head on each side (probably not true). Each platter has 621 concentric tracks. A 3-dimensional track (the same track on all disks) is called a cylinder. Each track is divided into 63 sectors. Each sector contains 512 bytes of data. Therefore the block size in the partition table is 64 heads * 63 sectors * 512 bytes er...divided by 1024. (See 4 for discussion on problems with this calculation.) The start and end values are cylinders.

$fdisk /dev/hdxx

n create a new partition

press <-| at first cylinder

define size +100M at Last cylinder

w write and quit

$sync

$partprobe -s /dev/hdxx

- rereads the partition table and updates the kernel table.
 -s - to show the output

Format

for ext2 format use $mke2fs for ext3 use $mke2fs -j

$ mke2fs - j /dev/hdxx

-j stands for journaling as ext3 is a journaling filesystem.

$mkdir /newdir

$mount /dev/hdxx /newdir

for permanent mount use fstab

$vi /etc/fstab

Append - /dev/hdxx /mnt ext3 defaults 0 0

fstab is 9th out of the 10 most critical and important configuration files which is stored in /etc directory, where all the configuration files are stored.
fstab stands for "File System TABle" and this file contains information of hard disk partitions and removeable devices in the system. It contains infor-mation of where the partitions and removeable devices are mounted and which device drivers are used for mounting them, which filesystem they are using and what permissions are assigned to them.

1st field device
2nd field mountpoint
3rd field filesystem
4th filed permisson
5th field backup for sixth field
6th field fsck sequence (same as chkdsk in windows)

Task

create 100mb partition for Linux.
Follow steps same as above
ext3 is a journaling which maintains record in its journal.
Fast recovery & recovery successful
Ext2 doesnt maintains journal.Slow recovery & no guarantee.

Task

create 100000kb partition for ext2.
Follow steps same as above.

Task

create 96mb partition for windows.
Follow steps same as above.

Mount all the created partitions under fstab.

Ext2 vs Ext3
At some point in your install, you'll probably want to switch filesystem types. In the base install, you're only given a choice of ext2 (short for ext2fs, or ``second extended filesystem," which is the ``standard" UNIX filesystem[7]. Ext3fs[8]is the same as ext2, but provides journaling.For those as sketchy on filesystem types as I am, it seems to be pretty basic. In the README on the original ext3 download page, the author answers the journaling question:

Q: What is journaling?
A: It means you don't have to fsck after a crash. Basically.

This is useful, because it means that every time your screen whites out and crashes while choosing the right video card (Section 1.2.1), you don't have to sit through an entire filesystem check of every inode. The filesystem still fscks itself every X mounts or Y days, but doesn't put you through the entire wait every time you crash it.To convert partition,s to the ext3 filesystem, you need to cleanly unmount them, boot something else (like the Debian CD you installed from -- see Section 6.2 on how to do this), and then, on a console, do:

```
tune2fs -j /dev/hdaX
```

wherein /dev/hdaX is the partition you want to add journaling to (hence the `-j' flag).Don't forget to modify the lines in your /etc/fstab to reflect that the partitions in question are to be mounted as ext3, not ext2. When cleanly unmounted, they can still be mounted as ext2, but the whole point of changing them was so they wouldn't be.

That's it. When you reboot, your partitions should come up as ext3.

Runlevels

Red Hat Linux/Fedora runlevels ID Description

0 Halt

1 Single-User mode

2 Multi-User mode with network enabled, but most network services disabled

3 Multi-User mode, console logins only

4 Not used/User-definable

5 Multi-User mode, with display manager as well as console logins

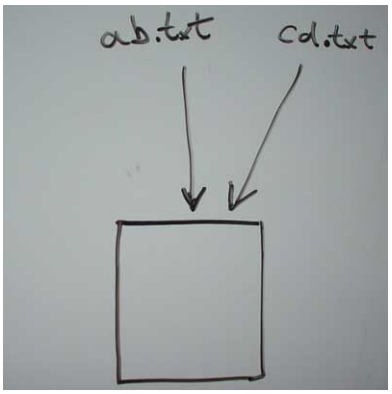
6 Reboot

Symlinks & Hardlinks

Files are arranged in directories (or folders if you prefer that term), and each file can be reached through a series of directories and sub-directories from the root - correct? Yes ... BUT ... there are some times that the same file can be reached through several names, and on Unix and Linux systems this is known as a "link".

There are two ways a link can be set up.

Hard Link

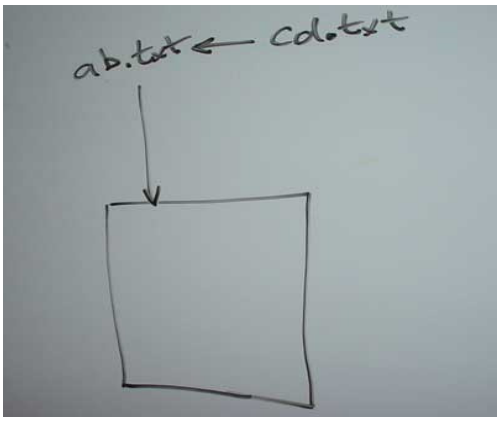


A **Hard Link** is where a file has two names which are both on an equal weighting, and both of the file names in the "inode table" point directly to the blocks on the disc that contain the data. See diagram to the left.
You set up a hard link with an ln command without options - if the file ab.txt already exists and you want to give an additional name (hard link) to it, you'll write ln ab.txt cd.txt and then both names will have equal ranking. The only way you'll know that there's a link there is by doing a long listing and you'll see a link count of 2 rather than 1, and if you need to find out what's linked to what, use the -i option to ls.

Symbolic Link

A **Symbolic Link** is where a file has one main name, but there's an extra entry in the file name table that refers any accesses back to the main name. This is slighly slower at runtime that a hard link, but it's more flexible and much more often used in day to day admin work.

Symbolic links are set up using the ln command with the -s option - so for example
ln -s ab.txt cd.txt
will set up a new name cd.txt that points to the (existing) file ab.txt. If you do a log listing (ls -l) of a directory that contains a symbolic link, you'll be told that it's a symbolic link with an "l" in the first column, and you'll be told where the file links to in the file name column. Very easy to spot!



Soft Links(Symbolic Links) :

1.Links have different inode numbers.

2. ls -l command shows all links with second column value 1 and the link points to original file.

3. Link has the path for original file and not the contents.

4.Removing soft link doesn't affect anything but removing original file the link becomes dangling link which points to nonexistant file.

In Softlink Inode is diff and the linked file will b a shortcut of first file

Hard Links :

1. All Links have same inode number.

2.ls -l command shows all the links with the link column(Second) shows No. of links.

3. Links have actual file contents

4.Removing any link just reduces the link count but doesn't affect other links.

In Hardlink Inode is same and both are independent

Soft link can create directories but hard link can't. Hard links created within that particular file system but soft link cross that file system

Hard links canot cross partition

A single inode number use to represent file in each file system. All hard links
based upon inode number.

So linking across file system will lead into confusing references for UNIX or
Linux. For example, consider following scenario

* File system: /home
* Directory: /home/sadhiq
* Hard link: /home/sadhiq/file2
* Original file: /home/sadhiq/file1

Now you create a hard link as follows:

```
$ touch file1
$ ln file1 file2
$ ls -l
```

Output:

```
-rw-r--r-- 2 sadhiq sadhiq 0 2006-01-30 13:28 file1
-rw-r--r-- 2 sadhiq sadhiq 0 2006-01-30 13:28 file2
```

Now just see inode of both file1 and file2:

```
$ ls -i file1
782263

$ ls -i file2
782263
```

As you can see inode number is same for hard link file called file2 in inode

table under /home file system. Now if you try to create a hard link for /tmp
file system it will lead to confusing references for UNIX or Linux file system.
Is that a link no. 782263 in the /home or /tmp file system? To avoid this
problem UNIX or Linux does not allow creating hard links across file system
boundaries. Continue reading rest of the Understanding Linux file system series

Practical

$mkdir /opt/new-file
$mkdir /usr/local/link-file

$vi /opt/newfile/abc

Append some content & save the above file

Now create a softlink for abc as xyz under /usr/local/link-file

$pushd /usr/local/link-file

$pwd

$ln -s /opt/new-file/abc xyz

Or

If u want to create symlink as from /home then

$pushd /home

$ln -s /opt/new-file/abc /usr/loal/link-file/xyz

Now check with the following & also note symlink files always have 777 perm

$ll | grep ^l

Also chk the size of both file and its self expalinatory

Now

$Append some data in xyz file u will get the same under abc

Now try removing the parent file in our case “abc”

$rm -rf /opt/new-file/abc

Now verify the symblink

$ll /usr/local/link-file/

Your file has broken symlink so its called orphaned

So when ever u delete a parent file it will effect & if softlink is deleted there is no effect in softlinks

Softlink files have different inodes of parent

Softlink can also cross partitions.

Now what if u want run a binary from different path and with different name

$which mount

$ln -s /sbin/mount /opt/mapping

$pushd /opt/

$./mapping

$ln -s /bin/pwd /usr/bin/prnt-work-dir

Now u can run the follow for “pwd”

$prnt-work-dir

$mkdir /opt/hard-link

$pushd /opt/link-file

Create a new file name file1 and aapend data

$echo “This is an new file” > file1

$cat file1

Now create a hard link from current path file1 to file2

$ln file1 /opt/hard-link/file2

Now try deleting and appending and try u done as above for soft link

Hardlinks are type of backup if parent & child is deleted no effect

Hardlinks have same inode numbers

Harslinks cannot cross parttitons, Also try crossing partitions

Also try creating 2 to 3 links for a single parent file in softlink and hardlink.

More

17. Hard Links and Symbolic Links

Today we're going to test your virtual imagination ability! You're probably familiar with shortcuts in Microsoft Windows or aliases on the Mac. Linux has something, or actually some things similar, called hard links and symbolic links.

Symbolic links (also called symlinks or softlinks) most resemble Windows shortcuts. They contain a pathname to a target file. Hard links are a bit different. They are listings that contain information about the file. Linux files don't actually live in directories. They are assigned an *inode* number, which Linux uses to locate files. So a file can have multiple hardlinks, appearing in multiple directories, but isn't deleted until there are no remaining hardlinks to it. Here are some other differences between hardlinks and symlinks:

1. You cannot create a hardlink for a directory.

2. If you remove the original file of a hardlink, the link will

still show you the content of the file.

3. A symlink *can* link to a directory.

4. A symlink, like a Windows shortcut, becomes useless when you remove the original file.

Hardlinks

Let's do a little experiment to demonstrate the case. Make a new directory called Test and then move into it. to do that, type:

```
$ mkdir Test
$ cd Test
```

Then make a file called FileA:

```
$ vi FileA
```

Press the I key to enter Insert mode:

```
i
```

Then type in some funny lines of text (like "Why did the chicken cross the road?") and save the file by typing:

```
Esc
ZZ
```

So, you made a file called FileA in a new directory called "Test" in your /home. It contains an old and maybe not so funny joke. Now, let's make a hardlink to FileA. We'll call the hardlink FileB.

```
$ ln FileA FileB
```

Then use the "i" argument to list the inodes for both FileA and its hardlink. Type:

```
$ ls -il FileA FileB
```

This is what you get:

```
1482256 -rw-r--r-- 2 sadhiq sadhiq 21 May 5 15:55 FileA
1482256 -rw-r--r-- 2 sadhiq sadhiq 21 May 5 15:55 FileB
```

You can see that both FileA and FileB have the same inode number (1482256). Also both files have the same file permissions and the same size. Because that size is reported for the same inode, it does not consume any extra space on your HD!

Next, remove the original FileA:

```
$ rm FileA
```

And have a look at the content of the "link" FileB:

```
$ cat FileB
```

You will still be able to read the funny line of text you typed. Hardlinks are cool.

Symlinks

Staying in the same test directory as above, let's make a symlink to FileB. Call the symlink FileC:

```
$ ln -s FileB FileC
```

Then use the i argument again to list the inodes.

```
$ ls -il FileB FileC
```

This is what you'll get:

```
1482256 -rw-r--r-- 1 sadhiq sadhiq 21 May 5 15:55 FileB
1482226 lrwxrwxrwx 1 sadhiq sadhiq 5 May 5 16:22 FileC -> FileB
```

You'll notice the inodes are different and the symlink got a "l" before the rwxrwxrwx. The link has different permissions than the original file because it is just a symbolic link. Its real content is just a string pointing to the original file. The size of the symlink (5) is the size of its string. (The "-> FileB" at the end shows you where the link points to.

Now list the contents:

```
$ cat FileB
$ cat FileC
```

They will show the same funny text.
Now if we remove the original file:

```
$ rm FileB
```

and check the Test directory:

```
$ ls
```

You'll see the symlink FileC is still there, but if you try to list the contents:

```
$ cat FileC
```

It will tell you that there is no such file or directory. You can still list the inode. Typing:

```
$ ls -il FileC
```

will still give you:

```
1482226 lrwxrwxrwx 1 sadhiq sadhiq 5 May 5 16:22 FileC -> FileB
```

But the symlink is obsolete because the original file was removed, as were all the hard links. So the file was deleted even though the symlink remains. (Hope you're still following.)

OK. The test is over, so you can delete the Test directory:

```
$ cd ..
```

$ rm -rf Test (r stands for recursive and f is for force)

Note: Be cautious using "rm -rf"; it's very powerful. If someone tells you to do "rm -rf /" as root, you might loose all your files and directories on your / partition! Not good advice.

Now you know how to create (and remove) hardlinks and symlinks to make it easier to access files and run programs. See you on the links!

Archiving & Compression

Archiving means that you take 10 files and combine them into one file, with no difference in size. If you start with 10 100KB files and archive them, the resulting single file is 1000KB. On the other hand, if you compress those 10 files, you might find that the resulting files range from only a few kilobytes to close to the original size of 100KB, depending upon the original file type.

ll of the archive and compression formats in this chapter — zip, gzip, bzip2, and tar — are popular, but

Zip

zip is probably the world's most widely used format. That's because of its almost universal use on Windows, but zip and unzip are well supported among all major (and most minor) operating systems,

Gzip

gzip was designed as an open-source replacement for an older Unix program, compress. It's found on virtually every Unix-based system in the world, including Linux and Mac OS X, but it is much less common on Windows. If you're sending files back and forth to users of Unix-based machines, gzip is a safe choice.

Bzip2

The bzip2 command is the new kid on the block. Designed to supersede gzip, bzip2 creates smaller files, but at the cost of speed. That said, computers are so fast nowadays that most users won't notice much of a difference between the times it takes gzip or bzip2 to compress a group of files.

Practical

zip both archives and compresses files, thus making it great for sending multiple files as email attachments, backing up items, or for saving disk space.

Create

```
$mkdir -p /opt/test/zip_dir;cd /opt/test/zip_dir
```

Append man pages to a file

```
$man ls > file-ls;cat /etc/fstab > file-fstab;cat
/root/anaconda.cfg > file-anaconda
```

```
$ls -lh
$ls -al
```

Zip the files to man-file.zip

```
$zip man-file.zip *
```

```
$ls -lF
```

```
$ man ls > file-ls.txt;cat /etc/fstab > file.txt;cat
/root/anaconda.cfg > file-anaconda.txt;man fdisk > file1.cfg;man
fstab > fstab.cfg;man man > man.cfg
```

Try compressing the files created using zip and verify the size of moby.zip files

```
$ zip -0 moby.zip1 *.txt
$ ls -l
$ zip -1 moby.zip2 *.cfg
$ ls -l
$ zip -9 moby.zip3 *.cfg
$ ls -l
```

You can also try

```
$alias zip='zip -9'
```

Create backup dir under mnt $mkdir/mnt/backup

Copy /opt/test contents with rsync
$rsync -parv/opt/test/* /mnt/backup/

Exclude moby.zip under /mnt/backup and create backup.zip under /usr/local/
$zip -r /usrlocal/backup.zip /mnt/backup -x "/mnt/backup/zip_dir/moby.zip1"

Change dir to /usr/local by pushd cmd (man pushd)
$pushd /usr/local/

Try Password protected zip

$ zip -P 12345678 backup.zip *.txt
$ zip -e backup.zip *.txt

$unzip -l

$unzip --ql backup.zip
verbose
unzip -v moby2.zip

list zipped files

$ unzip -l moby3.zip

List type
$ unzip -t moby2.zip

Now try any the following same as zip under any dir

gzip paradise_lost.txt
$ ls -l

```
$ gzip -c paradise lost.txt
w`
  I
◆l◆,(3 ◆◆i`+◆◆M◆S3◆t1*f%eY◆◆'[q◆◆
D◆◆]d]C%g◆      R◆◆,r◆e◆trB3+3/◆◆|*◆◆0D◆◆s
➥BAqn◆◆,Y8*#"]]RU
*b◆U\◆◆◆◆G◆◆◆'t(-◆◆x◆Yz3-◆o'~cnS◆◆K
 ◆c◆
```

Not good. Instead, output to a file.

$ ls -l
$ gzip -c paradise_lost.txt > paradise_lost.txt.gz
$ gzip -c -1 moby-dick.txt > moby-dick.txt.gz
$ ls -l
$ gzip -c -9 moby-dick.txt > moby-dick.txt.gz
$ ls -l
$ gzip -t paradise_lost.txt.gz
$ gunzip -c paradise_lost.txt.gz > paradise_lost.txt
$ bzip2 moby-dick.txt
$ ls -l

```
$ bzip2 -c moby-dick.txt > moby-dick.txt.bz2
$ ls -l
$ bzip2 -c -1 moby-dick.txt > moby-dick.txt.bz2
$ bzip2 -c -9 moby-dick.txt > moby-dick.txt.bz2
$ ls -l
$ bunzip2 moby-dick.txt.bz2
$ bunzip2 -c moby-dick.txt.bz2 > moby-dick.txt
$ bunzip2 -t paradise_lost.txt.gz
```

Get the Best Compression Possible with zip

-[0-9]

It's possible to adjust the level of compression that zip uses when it does its job. The zip command uses a scale from 0 to 9, in which 0 means "no compression at all" (which is like tar, as you'll see later), 1 means "do the job quickly, but don't bother compressing very much," and 9 means "compress the heck out of the files, and I don't mind waiting a bit longer to get the job done." The default is 6, but modern computers are fast enough that it's probably just fine to use 9 all the time.

In tabular format, the results look like this:

Book	zip -0	zip -1	zip -9
Moby-Dick	0%	54%	61%
Paradise Lost	0%	50%	56%
Job	0%	58%	65%
Total (in bytes)	1848444	869946	747730

Password-Protect Compressed Zip Archives

-P

-e

The Zip program allows you to password-protect your Zip archives using the -P option. You shouldn't use this option. It's completely insecure, as you can see in the following example (the actual password is 12345678):

unzip

Expanding a Zip archive isn't hard at all. To create a zipped archive, use the zip command; to expand that archive, use the unzip command.

Archive with Tar

Archive and Compress Files with tar and gzip
 -zcvf
If you look back at "Archive and Compress Files Using gzip" and "Archive and Compress Files Using bzip2" and think about what was discussed there, you'll probably start to figure out a problem. What if you want to compress a directory that contains 100 files, contained in various subdirectories? If you use gzip or bzip2 with the -r (for recursive) option, you'll end up with 100 individually compressed files, each stored neatly in its original subdirectory. This is undoubtedly not what you want. How would you like to attach 100 .gz or .bz2 files to an email? Yikes!
That's where tar comes in. First you'd use tar to archive the directory and its contents (those 100 files inside various subdirectories) and then you'd use gzip or bzip2 to compress the resulting tarball. Because gzip is the most common compression program used in concert with tar, we'll focus on that.
You could do it this way:

```
$mkdir -p /mnt/common/moby-dick
$cd /mnt/common/moby-dick
$ man ls > file-ls.txt;cat /etc/fstab > file.txt;cat
/root/anaconda.cfg > file-anaconda.txt;man fdisk > file1.cfg;man
fstab > fstab.cfg;man man > man.cfg
```

```
$cd ..
$pwd
/mnt/common/
```

```
$ ls -l moby-dick/*
$ tar -cf moby1.tar moby-dick/ | gzip -c > moby1.tar.gz
$ ls -l
```

That method works, but it's just too much typing! There's a much easier way that should be your default. It involves two new options for tar: -z (or --gzip), which invokes gzip from within tar so you don't have to do so manually, and -v (or --verbose), which isn't required here but is always useful, as it keeps you notified as to what tar is doing as it runs.

```
$ ls -l moby-dick/*
$ ls -l
```

The usual extension for a file that has had the tar and then the gzip commands used on it is .tar.gz; however, you could use .tgz and .tar.gzip if you like.

Note - It's entirely possible to use bzip2 with tar instead of

gzip. Your command would look like this (note the -j option, which is where bzip2 comes in):

```
$tar -cvzf moby.tar.gz moby-dick
$ tar -jcvf moby.tar.bz2 moby-dick/
```

In that case, the extension should be .tar.bz2, although you may also use .tar.bzip2, .tbz2, or .tbz. Yes, it's very confusing that using gzip or bzip2 might both result in a file ending with .tbz. This is a strong argument for using anything but that particular extension to keep confusion to a minimum.

Test Files That Will Be Untarred and Uncompressed

```
$ tar -jvtf moby.tar.bz2
```

Before you take apart a tarball (whether or not it was also compressed using gzip), it's a really good idea to test it. First, you'll know if the tarball is corrupted, saving yourself hair pulling when files don't seem to work. Second, you'll know if the person who created the tarball thoughtfully tarred up a directory containing 100 files, or instead thoughtlessly tarred up 100 individual files, which you're just about to spew all over your desktop.

To test your tarball (once again assuming it was also zipped using gzip), use the -t (or --list) option.

```
$ tar -zvtf moby.tar.gz
```

This tells you the permissions, ownership, file size, and time for each file. In addition, because every line begins with moby-dick/, you can see that you're going to end up with a directory that contains within it all the files and subdirectories that accompany the tarball, which is a relief.

Be sure that the -f is the last option because after that you're going to specify the name of the .tar.gz file. If you don't, tar complains:

```
$ tar -zvft moby.tar.gz
 tar: You must specify one of the '-Acdtrux' options
 Try 'tar --help' or 'tar --usage' for more information.
```

Now that you've ensured that your .tar.gz file isn't corrupted, it's time to actually open it up, as you'll see in the following section.

Note - If you're testing a tarball that was compressed using bzip2, just use this command instead:

```
$ tar -jvtf moby.tar.bz2
```

Untar and Uncompress Files

-zxvf

To create a .tar.gz file, you used a set of options: -zcvf. To untar and uncompress the resulting file, you only make one substitution: -x (or --extract) for -c (or --create).

```
 $ ls -l
 $ tar -zxvf moby.tar.gz
```

```
$ ls -l
```

Make sure you always test the file before you open it, as covered in the previous section, "Test Files That Will Be Untarred and Uncompressed." That means the order of commands you should run will look like this:

```
$ tar -zvtf moby.tar.gz
$ tar -zxvf moby.tar.gz
```

Note - If you're opening a tarball that was compressed using bzip2, just use this command instead:

```
$ tar -jxvf moby.tar.bz2
```

Repeat with different path

```
$ tar -cvf /mnt/backup/sam.tar /opt/test/zip_dir/*
```

Archive & compress with gzip

```
$ tar -cvf /mnt/backup/ramu.tar.gz /opt/test/zip_dir/*
$ pushd /mnt/backup
```

List before extracting

```
$ tar -tvf ramu.tar.gz
```

Understand the following

```
$ mkdir ramu;tar -zxvf ramu.tar.gz ramu/
$ ls ramu/
$ rm ramu/*
```

Also try and understand

```
$ cat ramu.tar.gz | gunzip -d | tar -xvf - /mnt/backup/ramu
$ ls /mnt/backup/ramu/
$ rm -rf /mnt/backup/ramu/*
$ gzcat ramu.tar.gz | tar -xvf - /mnt/backup/ramu
```

Finding files and archiving them

You can make a tarball of only certain types of files from a directory with the following one-liner:

```
$mkdir /mnt/common/test
$ find /mnt/common/moby-dick/ -name "*.txt" | xargs tar -zcpf reports.tar.gz
$ find /mnt/common/moby-dick/ -name "*.txt" | xargs tar -jcpf reports.tar.bz2
```

Now check

untar in a different directory

If you've got a gzipped tarball and you want to untar it in a directory other than the one you're in, do the following:

$ cd /mnt/backup

$ zcat reports.tar.gz | (cd ./otherdir; tar zxvf -)

$ ls

Understand the above cmd , note -: “-“ is used in after the arguments given to tar.

Extract individual files from a tarball

If you need a file that you've put into a tarball and you don't want to extract the whole file, you can do the following.

First, get a list of the files and find the one you want

$ cd /mnt/common/moby-dick

$ tar -zltf moby1.tar.gz

Then extract the one you want

$ tar zxvf moby1.tar.gz file-anaconda.txt

Backup everything with tar

To make a backup of everything in a particular directory, first do this

$ cd /mnt/common/moby-dick/

$ ls -a > backup.all

If you don't really want *everything*, you can also edit backup.all and get rid of things you don't want

To make the tarball, just do this:

$ tar -cvf newtarfile.tar `cat backup.all`

(remember, those are backtics)

Extracting Specific Files

Extract a file called etc/default/sysstat from config.tar.gz tarball:

$ tar -cvzf /opt/test/config.tar.gz /mnt/backup/ramu

$ tar -ztvf config.tar.gz

$ tar -zxvf config.tar.gz <any file>

$ tar -xvf {tarball.tar} {path/to/file}

Some people prefers following syntax:

$ tar --extract --file={tarball.tar} {file}

Extract a directory called css from cbz.tar:

$ tar --extract --file=cbz.tar css

Wildcard based extracting

You can also extract those files that match a specific globbing

pattern (wildcards). For example, to extract from cbz.tar all files that begin with pic, no matter their directory prefix, you could type:

Note before attempting the following you have to create tar files as cbz.tar with he files you are going to extract.

```
$ tar -xf cbz.tar --wildcards --no-anchored 'pic*'
```

To extract all php files, enter:

```
$ tar -xf cbz.tar --wildcards --no-anchored '*.php'
```

-x: instructs tar to extract files.

-f: specifies filename / tarball name.

-v: Verbose (show progress while extracting files).

-j : filter archive through bzip2, use to decompress .bz2 files.

-z: filter archive through gzip, use to decompress .gz files.

--wildcards: instructs tar to treat command line arguments as globbing patterns.

--no-anchored: informs it that the patterns apply to member names after any / delimiter.

Have you ever seen this error when using tar?

```
$ tar -czf etc.tgz /etc
```

Removing leading `/' from member names

Tar is removing the leading / from the archive file, and warning you about it. Although you can redirect STDERR to /dev/null, doing so can result in missed errors. Instead, use tar with the -P or --absolute-names switch. They do the same thing: leave the leading / in the archived files.

```
$ tar -czPf etc.tgz /etc
```

When you untar the archive without -P, the leading / will still equate to your current working directory. Use the -P when untarring to restore from archive to the absolute path name. For example:

The following creates ./etc (dot, slash, etc)

```
$ tar -xzf etc.tgz
```

This overwrites /etc (slash, etc)!

```
$ tar -xzPf etc.tgz
```

PATH is an *environmental variable* in Linux and other Unix-like operating systems that tells the shell which directories to search for executable files (i.e., ready-to-run programs) in response to commands issued by a user. It increases both the convenience and the safety of such operating systems and is widely considered to be the single most important environmental variable.

Environmental variables are a class of *variables* (i.e., items whose values can be changed) that tell the shell how to behave as the user works at the command line (i.e., in a text-only mode) or with shell scripts (i.e., short programs written in a shell programming language). A shell is a program that provides the traditional, text-only user interface for Unix-like operating systems; its primary function is to read commands that are typed in at the command line and then *execute* (i.e., run) them.

Practical - Setting Path

Login as root

$id

$echo $PATH

$useradd john

$passwd john

$su - john

$id

Verify john's PATH

$echo $PATH

you cant find :/sbin:/usr/sbin so u cant run cmnd's fdisk, shred under the same.

$fdisk -l

will get command not found.

So u can set path, **but it's temporary for the shell.**

$PATH=$PATH=:/sbin:/usr/sbin

To set under environment run

$export PATH

For permanent

you can locate the above two cmnds under /etc/profile file, which run's always after login.

Now chk you will get the above added dir under john’s path.

$echo $PATH

Now try

$ fdisk -l

Note-: The cmd is executed but fdisk binary will work only by uid 0 (root), bcoz it's programmed like that.

So search for the cmd in /sbin & /usr/sbin , which can run by other uid's.

Now create a testscript under /opt and execute the script

$vi /opt/testscript

#Append the following

echo " THIS IS MY SCRIPT"

#Save

$cd /opt

set execute permisson

$chmod +x /opt/testscript

$./testscript # (./ means current path execution)

But what if u want to run the script from any other directories under your filesystem hiriearchy.

Then set the /opt dir to the users path as mentioned above or copy the script under the following PATH . (which is already set)

set. For eg-:

$PATH=$PATH:/opt

$cd /

$testscript

or

$cp /opt/testscript /bin or /usr/local/bin etc...

Now try running the script

$cd /

$testscript

Daemons&Process

Application Daemon are those which can be killed & will have no effect to the sysytem

$ kill -15 <appd-pid>
For eg. - firefox, openoffice, X server, etc...

System Daemons are those which can be killed & will effect the system.
$ kill -9 <sysd-pid>

For eg - init, kerneld, ksoftirqd, khelper, kthread, kblockd

OBJECTIVES

Defining a process
Process states
Process management
Job control
System Information
Performance Related Information

What is a Process?

A process has many components and properties

- exec thread
- PID
- priority
- memory context
- environment
- file descriptors
- security credentials

How Processes Are Created

One process “forks” a child, pointing to the same pages of memory, and marking the area as read-only.Then, the child “execs” the new command, causing a copy-on-write fault, thus copying to a new area of memory. A process can exec, without forking. The child maintains the process ID of the parent.

Process Ancestry

init is the first process started at boot time - always has PID 1.
Except init, every process has a parent.
Processes can be both a parent and a child at the same time.
Understand the Multiuser Environment.

One of the goals of UNIX was to enable a number of users to use the system simultaneously (multiuser capability). Because several users might also want to use several different programs simultaneously, mechanisms must be available to allow these programs to run simultaneously (multitasking capability).
The implementation of a multiuser and multitasking system appears to be simultaneous in a single processor system, but this is only possible in a multiprocessor system.

Even in a single-processor system, advantages can be gained through multitasking because waiting times for input or output from processes can be used for other processes.

UNIX implements preemptive multitasking—each process is allowed a maximum time with which it can work. When this time has expired, the operating system takes processor time away from the process and assigns it to another process waiting to run. Other operating systems (such as versions older than the MAC OS version X) do not intervene in this process cycle. Instead, control over the processor must be released by the running process before another process can run.

This can lead to one process hijacking the processor, leaving other processes without processing time and blocking the system. The operating system coordinates access to the resources available in the system (hard drives, tapes, interfaces). If there is competition among processes, e.g., for access to a tape device, only one process can be granted access. The others must be rejected. This coordination task is very complex and no operating system is able to implement an ideal solution. The classic problem involves a situation in which two or more processes exclusively need the same resources, as illustrated in the following resource conflict:

The following describes the resource conflict:
Process A needs resources Res.1 and Res.2.
Process B needs resources Res.2 and Res.1.
Process A has received access to Res.1 and would now also like access to Res.2. In the meantime, however, B has already gained access to Res.2 and, in turn, would like access to Res.1 as well.

If these two processes wait until what they need is available, nothing more will happen-they are deadlocked.

Multithreading is an extension of multitasking, and helps solve this problem. In multithreading, a number of parts independent from one another (threads) can be produced within a process.

Multithreading increases the level of parallel processes with each thread needing to be administered, which makes the use of a multiprocessor system more valuable.

A clear distinction should be made here between programs and processes: as a rule, a program exists only once in the system, but there can be several processes that perform the same program.If a number of users are active, both programs and processes can be used independently of one another (such as a program used to display directories).

Processes and Multitasking

Terminology can be confusing

Multiuser: system can simultaneously service more than one online terminal
Multiprogramming: the system can execute more than one program at the “same” time
Multitasking: system can execute two or more tasks at the “same” time

In common usage, these all refer to the same thing

Multitasking Operating Systems

Multitasking OSs are designed to perform a complex juggling trick They must:

- Allocate resources, such as CPU cycles and memory, and assign priorities so each process receives adequate attention
- Higher priority jobs need more or larger CPU time-slices without neglecting lower priority jobs
- Jobs that are waiting for some resource (such as user input, input from disk, or a shared output such as a printer) need to handled without wasting CPU time

Multitasking on a Single CPU

Obviously, a single CPU cannot run multiple process simultaneously. The OS simulates simultaneity by switching between tasks at a high rate. Each switch is a “time-slice” Since thousands or hundreds of thousands of CPU cycles can go by between user keystrokes, this gives the appearance of simultaneous operation.

This resource allocation, priority processing, and time-slicing is all done by the scheduler Unix Scheduling Algorithm

Unix schedules tasks in this order:

- Highest priority task that is Ready-to-Run and loaded in memory and preempted
- Ties for priority are broken by time spent waiting (also known as Round-Robin scheduling)
- If no one is ready to run, the kernel idles until the next time-slice Unix Images and Processes

Each process receives a unique numerical process identifier (pid) when it is started. Even if the same program is run multiple times, each instance will have a unique PID. A process has an image in RAM.

Forks and Spawns:
- When a process A is running, it can spawn another process B
- It does this using the fork system call
- B is said to be the child of A and A is known as the parent of B
- Initially, the child and parent are virtually identical

They each start with identical but independent copies of the RAM image, but being separate processes, they have unique PIDs.

The child then calls the system call exec using the command name and arguments inherited from the parent. From this point on, the child and parent can go their separate ways. However, since they both have access to the same open files and pipes, there is a potential for communication between them (interprocess communication). The shell is the parent of most of your processes The Shell is a Process. The principle process you interact with is the shell. The shell can run some commands (builtins) itself but for most commands, it forks a separate process. It usually waits for the command process to finish and then gives you a new shell prompt.

What if you could tell the shell not to wait? You could then instruct the shell to do something else while the first command was running in the background Voila! Multiprocessing in action!

Redhat Linux comparing with other Unices/Linuces, its shipped with plethora of options for monitoring system, utilization with regards to CPU, Memory and Disk etc.

```
$ uptime
 18:18:16 up 3 days, 7:37, 5 users, load average: 0.00, 0.00, 0.00
 Tells you exactly how long your system is been running from

 1mt 5mt 15mt
 load average: 0.00, 0.00, 0.00

$ cat /proc/meminfo
 /proc
```

Virutal Directory created in RAM. It runs whenever the system is running. It represents real time information and values stored in are accurate. It doesn't occupy space on the disk

$ cat /proc/cpuinfo → CPU Information A process has many components and properties.

Display and update information about the top cpu processes

$ top

Top displays the top 10 processes on the system and periodically updates this information. Top command is a combination of various commands to display CPU stats, memory, real time processes running in the system Top refresh every 5 seconds Process States. Unix uses several process states to determine the current condition of a process.

- Runnable
- Stopped
- Page Wait
- Non-Interruptable wait

Typically for disk I/O or NFS requests
- Sleeping
- Idle
- Terminated

OPTIONS

-q Renice top to -20 so that it will run faster. This can be used when the system is being very sluggish to improve the possibility of discovering the problem.
-dcount Show only count displays, then exit. A display is considered to be one update of the screen. This option allows the user to select the number of displays he wants to see before top automatically exits. For intelligent terminals, no upper limit is set. The default is 1 for dumb terminals.
-stime Set the delay between screen updates to time seconds. The default delay between updates is 5 seconds.

INTERACTIVE MODE

h or ? Display a summary of the commands (help screen). Version information is included in this display.
Q Quit top.
K Send a signal ("kill" by default) to a list of processes
R Change the priority (the "nice") of a list of processes.
S Change the number of seconds to delay between displays (prompt

for new number).
O Change the order in which the display is sorted. This command is not available on all systems. The sort key names vary from system to system but usually include: "cpu", "res", "size", "time". The default is cpu.

THE DISPLAY

PID every process runs have the process ID USER owner of the process
PRI Current priority of the process.
NICE Nice amount in the range -20 to 20, as established by the use of the command nice.
RES Resident memory: current amount of process memory that resides in physical memory, given in kilobytes.
STATE Current state (typically one of "sleep", "run", "idl", "zomb", or "stop").
TIME Number of system and user cpu seconds that the process has used.
SIZE Amount of memory the process needs
CPU Percentage of available cpu time used by this process.
COMMAND Name of the command that the process is currently running

PROCESS STATE CODES

Here are the different values that the s, stat and state output specifiers (header "STAT" or "S") will display to describe the state of a process.
D Uninterruptible sleep (usually IO)
R Running or runnable (on run queue)
S Interruptible sleep (waiting for an event to complete)
T Stopped, either by a job control signal or because it is being traced.
W paging (not valid since the 2.6.xx kernel)
X dead (should never be seen)
Z Defunct ("zombie") process, terminated but not reaped by its parent. zombie -- dead process

For BSD formats and when the stat keyword is used, additional characters may
be displayed:
high-priority (not nice to other users)

N low-priority (nice to other users)
L has pages locked into memory (for real-time and custom IO)
s is a session leader
l is multi-threaded (using CLONE_THREAD, like NPTL pthreads do)
+ is in the foreground process group

Each process has a unique identification number (PID) which characterises the process. The command top allows you to kill

processes using the “k” interactive command and entering the PID of the relevant process. To leave top to just press the “q” key.

Free

```
$ free -m
$ free -c 5 -s 3
$ free -m
total used free shared buffers cached
Mem: 1003 981 22 0 91 688
-/+ buffers/cache: 201 802
Swap: 1058 0 1058
```

As you can see, my system has 1 GB of ram and 981 MB are in use leaving 22MB free. If you look at the cached column, it shows 688 MB free. This is a good thing as cached memory is basically free memory. This is where programs a user may have used earlier and then quit are stored, just on the off chance that the user might start up the program again. On the other hand, if the user starts up a new program, this cache can be replaced for the new program that is running. It should be mentioned that the caching works not just for recently loaded programs but also for data, i.e. recently used files and directories. Program loading is just a special case of loading a file.

The -/+ buffers/cache section is will show you what is really going on. In my example, it shows that only 201 MB are in use and that 802 MB are free. The rest is just cached. What a user really needs to worry about is that last line. If you start seeing the swap file go into use that means that you are out of free ram and you are now using space on your hard disk to help out. If this starts happening, the best thing to do is run the top command and see what is taking up all the memory. Then, if it is an unneeded program, shut it down.

Signals

Signals are a software mechanism that are similar to a message of some sort. They can be trapped and handled or ignored

Signals operate through two different system calls
1) The kill system call
2) The signal system call

1) The kill System Call
The kill system call sends a signal to a process kill is generally used to terminate a process. It requires the PID of the process to be terminated and the signal number to send as arguments.

2) The Signal System Call

The signal system call is much more diverse. When a signal occurs, the kernel checks to see if the user had executed a signal system call and was therefor expecting a signal. If the call was to ignore the signal, the kernel returns
Otherwise, it checks to see if it was a trap or kill signal If not, it processes the signal If it was a trap or kill signal, the kernel checks to see if core should be dumped and then calls the exit routine to terminate the user process.

Common Unix Signals

```
$kill -l

SIGHUP              Hang-up
SIGINT              Interrupt
SIGQIT              Quit
SIGINS              Illegal Instruction
SIGTRAP             Trace Trap
SIGKILL             Kill
SIGSYS              Bad argument to system call
SIGPIPE             Write on pipe with no one to read it
SIGTERM             Software termination signal from kill
SIGSTOP             Stop signal
```

See /usr/include/sys/signal.h

Signal Acceptance

There are a couple of possible actions to take when a signal occurs

Ignore it
Process it
Terminate

The superuser can send signals to any process.
Normal users can only send signals to their own processes

Process Termination

A process is terminated by executing an exit system call or as a result of a kill signal. When a process executes an exit system call, it is first placed in a zombie state. In this state, it doesn't exist anymore but may leave timing information and exit status for its parent process. A zombie process is removed by

executing a wait system call by the parent process.

Process Cleanup

The termination of a process requires a number of cleanup actions
These actions include:

Releasing all memory used by the process
Reducing reference counts for all files used by the process
Closing any files that have reference counts of zero
Releasing shared text areas, if any
Releasing the associated process table entry, the proc structure

This happens when the parent issues the wait system call, which returns the
terminated child's PID

kill - signal a process
kill is somewhat strangely named
Sends the specified signal to a process

Syntax: kill [-sig_no] pid

kill -l (display list of signals)
-sig_no - signal number to send
pid - process id of process to receive signal

Default signal is TERM sig_no is 15, or request-process-termination
kill -9 pid terminates the process with extreme prejudice. As usual, you can only kill your own processes unless you are the superuser.

$ kill -9 <PID>
$ kill -l -> lists all available signals
$ killall
$ pidof <pidname>
$ pgrep <pidname>
$ pkill <pidname>

Job Control

Job control refers to the ability to selectively stop (suspend) the execution of processes and continue (resume) their execution at a later point. A job is one or more processes started from a single command line. By default, only one job can be run in the foreground. This means that when a job is being executed in the foreground the command line is unavailable. When the job
has finished executing the command prompt is reissued.

It is also possible to suspend jobs and/or run multiple jobs in the background, in which case the command line is still available in the foreground, although any output from running background jobs will still be displayed at the terminal. You can see the jobs currently running or stopped in the background using the jobs command.

The syntax for the jobs command is shown below:
jobs option(s)
Common jobs options are:

Option Explanation:
l Shows the job number, its PID, its status, and its name
p Shows just the PID of running jobs

Issuing the jobs command without any options will show a list of all running, stopped and
suspended background jobs.
An example of using the job command is illustrated below:

```
$ jobs -l
[1]- 1229 Running tail -n5 -f /var/log/secure
[2]+ 1230 Stopped joe fred
```

In the above example there are two jobs in the background, one running and one stopped.

File Permissions

File permissions are assigned to:

1. the owner of a file
2. the members of the group the file is assigned to
3. all other users
4. Permissions under Linux are configured for each file and directory.

There are three levels of permissions:

1. The permissions that apply to the owner of the file. The owner of a file is by default the user that created the file1.
2. The permissions that apply to all members of the group that is associated with the file.
3. The permissions that apply to all other users on the system.
4. Permissions can only be changed by the owner, and root of course.

For a file, these permissions mean the following:

read allow the user to read the contents of the file, for instance with cat or less.

write allow the user to modify the contents of the file,for instance with vi.

execute allow the user to execute the file as a program, provided that the file is indeed an executable program (such as a shell script).

For a directory, these permissions have a slightly different meaning:

read allow the user to view the contents of the directory, for instance with ls.

write allow the user to modify the contents of the directory. In other words: allow the user to create and delete files, and to modify the names of the files. Note: Having write permissions on a directory thus allows you to delete files, even if you have no write permissions on that file!

execute allow the user to use this directory as its current working directory. In other words: allow the user to cd into it.

r - read
w - write
x - execute
• u for the owner (user) of the file
• g for the group assigned to the file
• o for all other users
• a for all (owner+group+others)

<operator> can be:
• + to add permissions
• - to delete permissions
• = to clear all permissions and set to the permissions specified

Symbolic way

```
$ useradd sachin
$ passwd sachin
$ useradd dhoni
$ passwd dhoni
$ groupadd market;usermod -G market dhoni
$ useradd shewag
$ passwd shewag
$ groupadd market;usermod -G market shewag
$ mkdir /opt/perm/;touch /opt/perm/file{1..6}
$ mkdir /opt/perm/{data1,data2}
$ cd /opt/perm
$ ll -d data1
 drwxr-xr-x 2 root root 4096 Jul 29 20:15 data1
$ chown sachin data1
$ ll -d data1
$ chgrp market data1
$ ll -d data1
$ chmod u-w data1
$ ll -d data1
$ chmod g+w data1
$ ll -d data1
$ chmod o+w,o-rx data1
$ ll -d data1
$ ll -d data2
 drwxr-xr-x 2 root root 4096 Jul 29 20:15 data2
$ chown -Rv sachin.market data2
$ ll -d data2
$ chmod u-rwx data2
$ ll -d data2
$ chmod g+w,g-x data2
$ ll -d data2
$ chmod -Rv o+w,o-r data2
$ ll -d data2
```

Octal way

```
$ ll file1
 -rw-r--r-- 1 root root 0 Jul 29 20:15 file1
$ chmod 777 file1
$ ll file1
$ chmod 666 file2
$ ll file1
$ chmod 467 file3
$ ll file1
$ chmod 541 file4
$ ll file1
$ chmod 724 file5
$ ll file1
$ chmod 000 file6
$ chmod 0 file6
```

This table shows what numeric values mean:

Octal digit	Text equivalent	Binary value	Meaning
0	---	000	All types of access are denied
1	--x	001	Execute access is allowed only
2	-w-	010	Write access is allowed only
3	-wx	011	Write and execute access are allowed
4	r--	100	Read access is allowed only
5	r-x	101	Read and execute access are allowed
6	rw-	110	Read and write access are allowed
7	rwx	111	Everything is allowed

Umask
User Mask

New files should not be created with 666! To avoid this problem a permission mask exists. It is obviously important to know with what permissions new files and directories are created. Under Linux, it's not really easy to tell, since the default permissions can be modified by setting a umask (with the umask command).

If no umask were set (which never happens, by the way), a file would always be created with permissions 666 (rw-rw-rw-) and a directory would get 777 (rwxrwxrwx). In actual practice however, a umask is set, and this number is subtracted from these permissions.

So, with a umask of 022, the default permissions for a file will become 644 (rw-r--r--, 666-022) and the default permissions for a directory will become 755 (rwx-r-xr-x, 777-022).

The default umask depends on your distribution, and whether your distribution uses something called "User Private Groups".

• Red Hat assigns a umask of 002 to regular users, and 022 to root.
• SUSE assigns a umask of 022 to all users, including root.
- What is your current default permission (umask)
- How do you set your default permission?
- Umask defines what permissions, in octal, cannot be set
- Umask stands for user file creation mode mask
- In essence, system sets the default permission on the file and directory
- If i would have no "umask:, the default permission on the file would be "777"
- Usually set in a login script
- it is the inverse of the normal octal permissions
- "umask -S" shows your umask in symbolic form
- linux removes the "x" permissions (or the 1) so 777 is the same as 666

- here are the common umask values:
 --> 000 = full access (r+w) to everyone, or 666
 --> 006 = no access to other, or 660
 --> 022 = full access (r+w) to user and r to g and 0, or 644
 --> 066 = full access (r+w) to user and no access to g + o, or 600
-

Normally, you can subtract from 666 but be very careful as it may

be 777. In Fedora Linux, it is 666 but lets test it out...
--> View the current umask setting
$umask

--> shows your umask in symbolic form
$ umask -S

- Umask on directory should be subtract from 777

```
  777
- 022
 ------
 755
```

System-wide umask for all users in /etc/profile
Individual umask in $HOME/.bash_profile or $HOME/.profile

Default value of umask is:
For root 022
For user 002 (if user private groups are used) or 022 (otherwise)
The umask specifies what permission bits will be set on a new file when it is created. The umask is an octal number that specifies the which of the permission bits will not be set. On Task

I
change Symbolic way
1.Give 704 to abc file
2.Give 417 to abc file
3.Give 006 to abc file
4.give 707 to abc file

II
change Octal way
1.change to octal mode r-xrw-r-x to abc chmod 565
2.change to octal mode --xr-xr-- to abc chmod 154
3.change to octal mode rw----rwx to abc chmod 607
4.change to octal mode ---r-x--- to abc chmod 050

III
symbolic way
1.change r-xrw-r-x to rw--wxrwx to abc chmod u+w,u-x,g-r,g+x,o+w
2.change --xr-xr-- to rwxrwxrw- to abc chmod u+rw,g+w,o+w
3.change rw----rwx to --x----wx to abc chmod u-rw,u+x,o-r
4.change ---r-x--- to rwx-w-rwx to abc chmo u+rwx,g-rx,g+w,o+rwx

Administarative cmnds and Lowlevel cmnds

Lowlevel

/bin This directory contains executable programs which are needed in
single user mode and to bring the system up or repair it.

Administrative

/sbin Like /bin, This directory holds commands needed to boot the sys-
tem, but which are usually not executed by normal users.

Lowlevel

/usr/bin This is the primary directory for executable programs. Most
programs executed by normal users which are not needed for boot-
ing or for repairing the system and which are not installed
locally should be placed in this directory.

Administrative

/usr/sbin This directory contains program binaries for system administra-
tion which are not essential for the boot process, for mounting
/usr, or for system repair.

Understanding UNIX / Linux file system

A conceptual understanding of file system, especially data structure and related terms will help you become a successful system administrator. I have seen many new Linux system administrator w/o any clue about file system. The conceptual knowledge can be applied to restore file system in an

emergency situation.

What is a File?

File are collection of data items stored on disk. Or it's device which can store the information, data, music (mp3), picture, movie, sound, book etc. In fact what ever you store in computer it must be inform of file. Files are always associated with devices like hard disk ,floppy disk etc. File is the last object in your file system tree. See Linux/UNIX – rules for naming file and directory names.

What is a directory?

Directory is group of files. Directory is divided into two types: Root directory – Strictly speaking, there is only one root directory in your system, which is denoted by / (forward slash). It is root of your entire file system and can not be renamed or deleted.

- Sub directory – Directory under root (/) directory is subdirectory which can be created, renamed by the user.

Directories are used to organize your data files, programs more efficiently.

Linux supports numerous file system types

3. Ext2: This is like UNIX file system. It has the concepts of blocks, inodes and directories.
4. Ext3: It is ext2 filesystem enhanced with journalling capabilities. Journalling allows fast file system recovery. Supports POSIX ACL (Access Control Lists).
5. Isofs (iso9660): Used by CDROM file system.
6. Sysfs: It is a ram-based filesystem initially based on ramfs. It is use to exporting kernel objects so that end user can use it easily.
7. Procfs: The proc file system acts as an interface to internal data structures in the kernel. It can be used to obtain information about the system and to change certain kernel parameters at runtime using sysctl command. For example you can find out cpuinfo with following command:

What is a UNIX/Linux File system?

A UNIX file system is a collection of files and directories stored. Each file system is stored in a separate whole disk partition. The following are a few of the file system:

- / – Special file system that incorporates the files under several directories including /dev, /sbin, /tmp etc
- /usr – Stores application programs
- /var – Stores log files, mails and other data
- /tmp – Stores temporary files

Exploring Linux File System Hierarchy

A typical Linux system has the following directories:

=> **/** : This is the root directory.

=> **/bin** : This directory contains executable programs which are needed in single user mode and to bring the system up or repair it.

=> **/boot** : Contains static files for the boot loader. This directory only holds the files which are needed during the boot process.

=> **/dev** : Special or device files, which refer to physical devices such as hard disk, keyboard, monitor, mouse and modem etc

=> **/etc** : Contains configuration files which are local to the machine. Some larger software packages, like Apache, can have their own subdirectories below /etc i.e. /etc/httpd. Some important subdirectories in /etc:

=> **/home** : Your sweet home to store data and other files. However in large installation yhe structure of /home directory depends on local administration decisions.

=> **/lib** : This directory should hold those shared libraries that are necessary to boot the system and to run the commands in the root filesystem.

=> **/lib64** : 64 bit shared libraries that are necessary to boot the system and to run the commands in the root filesystem.

=> **/mnt** : This directory contains mount points for temporarily mounted filesystems

=> **/opt** : This directory should contain add-on packages such as install download firefox or static files

=> **/proc** : This is a mount point for the proc filesystem, which provides information about running processes and the kernel.

=> **/root** : This directory is usually the home directory for the root user.

=> **/sbin** : Like /bin, this directory holds commands needed to boot the system, but which are usually not executed by normal users, root / admin user specific commands goes here.

=> **/tmp** : This directory contains temporary files which may be deleted
with no notice, such as by a regular job or at system boot up.

=> **/usr** : This directory is usually mounted from a separate partition.
It should hold only sharable, read-only data, so that it can be mounted by various machines run ning Linux (useful for diskless client
or multiuser Linux network such as university network). Programs, libraries, documentation etc. for all user-related programs.

=> **/var** : This directory contains files which may change in size, such
as spool and log files.

=> **/lost+found** : Every partition has a lost+found in its upper directory. Files that were saved during failures are here,

for e.g

ext2/ext3 fsck recovery.

- **/etc/skel** : When a new user account is created, files from this directory are usually copied into the user's home directory.

- **/etc/X11** : Configuration files for the X11 window system .

* **/etc/sysconfig** : Important configuration file used by SysV script stored in /etc/init.d and /etc.rcX directories

/etc/cron.* : cron daemon configuration files which is used to execute scheduled commands

Common Linux log files name and usage

* /var/log/message: General message and system related stuff
* /var/log/auth.log: Authenication logs
* /var/log/kern.log: Kernel logs
* /var/log/cron.log: Crond logs (cron job)
* /var/log/maillog: Mail server logs
* /var/log/qmail/ : Qmail log directory (more files inside this directory)
* /var/log/httpd/: Apache access and error logs directory
* /var/log/lighttpd: Lighttpd access and error logs directory
* /var/log/boot.log : System boot log
* /var/log/mysqld.log: MySQL database server log file
* /var/log/secure: Authentication log
* /var/log/utmp or /var/log/wtmp : Login records file
* /var/log/yum.log: Yum log files

Go to /var/logs directory:#

$cd /var/logsView common log file /var/log/messages using any one of the

following command:

```
$ tail -f /var/log/messages
$ less /var/log/messages
$ more -f /var/log/messages
$ vi /var/log/messagesOutput:
```

Device Driver character,block,socket

. Type field: The first character in the field indicates a file type of one of the following:

* d = directory.

* l = symbolic link.

* s = socket – sockets are special files offering a type of network interface.

* p = named pipe – handling other programme other than kernel driver.

* - = regular file.

* c= character (unbuffered) device file special.

* b=block (buffered) device file special.

*D=door A door is a special file for inter-process communication between a client and server.

Ref –

http://www.securityfocus.com/infocus/1872

http://tldp.org/LDP/Linux-Filesystem-Hierarchy/html/index.html

http://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard

http://www.comptechdoc.org/os/linux/howlinuxworks/linux_hlfilesystems.html

FSTAB

fstab is 9th out of the 10 most critical and important configuration files which is stored in /etc directory, where all the configuration files are stored.

fstab stands for "File System TABle" and this file contains information of hard disk partitions and removeable devices in the system. It contains information of where the partitions and removeable devices are mounted and which device drivers are used for mounting them, which filesystem they are using and what permissions are assigned to them.

The file fstab contains descriptive information about the various file systems. fstab is only read by programs, and not written; it is the duty of the system administrator to properly create and maintain this file. Each filesystem is described on a separate line; fields on each line are separated by tabs or spaces. Lines starting with '#' are comments. The order of records in fstab is important because fsck, mount, and umount sequentially iterate through fstab doing their thing.

Example of a fstab file content :

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
LABEL=/         /                 ext3      defaults            1
1
LABEL=/boot      /boot              ext3     defaults            1
2
none             /dev/pts           devpts   gid=5,mode=620      0
0
LABEL=/home      /home              ext3     defaults            1
2
none             /proc              proc     defaults            0
0
none             /dev/shm           tmpfs    defaults            0
0
LABEL=/tmp       /tmp               ext3     defaults            1
2
LABEL=/u01       /u01               ext3     defaults            1
2
LABEL=/usr       /usr               ext3     defaults            1
2
LABEL=/var       /var               ext3     defaults            1
2
/dev/hda6        swap               swap     defaults            0
0
/dev/cdrom       /mnt/cdrom         udf,iso9660   noauto,ro      0
0
/dev/fd0         /mnt/floppy        auto     noauto,owner,kudzu  0
0
/dev/sda1        /mnt/usb_hdd       vfat     noauto              0
0
\________/       \____________/   \_________/\_____________/     \_/
\_/
    |                  |             |                  |         |
|
   1st                2nd            3rd                4th      5th
6th
```

There are total six columns in the fstab file separated by spaces or tabs. Each column holds different information about the device. For adding any new device add a fresh row. Each row stands for a partition or removeable device in the system.

1st Column :

~~~~~~~~~~

The first column contains the partitions's label, eg. "LABEL=/boot" or driver's path, eg. "/dev/cdrom". Device driver's path tells the system to mount the device with the mentioned device driver.

2nd Column :
~~~~~~~~~~

The second field (fs_file) describes the mount point for the filesystem.
For swap partitions, this field should be specified as `none'. If the name of
the mount point contains spaces these can be escaped as `\040'.

The second column shows the mount point specified for a device in the fstab file. The mount points actually is the directory where that particular device(mentioned in the first column) will be mounted and through which we can view and modify the content of that partition. You can change the default mount point listed in the column, if you are not satisfied with the one your system has given you.

3rd Column :
~~~~~~~~~~

The third column in the file specifies the file system type of the device or partition.Many diffrent file systems are supported by Linux and most common ones are,

1) autofs
2) devpts
3) ext2
4) ext3
5) iso9660
6) nfs
7) ntfs
8) proc
9) swap
10) tmpfs
11) udf
12) ufs
13) vfat
14) xfs

If you are not sure of the file system type of the device then set the value to "auto" and the system will itself determine the file system type and will mount the device with that file system.

4th Column :
~~~~~~~~~~

The fourth column is for permissions to be given to the partition at the time of booting. There are many options which constitutes the forth column. They are as follows : -

1) **ro** - Read Only

2) **rw** - Read Write

3) **auto** - Mount on startup

4) **noauto** - Do not mount on startup

5) **user** - Any user can mount, but only unmount device mounted by him

6) **nouser** - Only root can mount & unmount the device

7) **users** – Every user can mount and also unmount the device mounted by others

8) **owner** – Same as user (above no. 5)

9) **dev** – User can use device driver to mount the device

10) **nodev** – User cannot use device driver to mount the device

11) **exec** – Users can execute binaries on the partition

12) **noexec** – Users cannot execute binaries on the partition

13) **async** – Asynchronous, whenever a file is saved it will be first saved in the RAM and after 30 seconds all the queued files will be written on the hard disk

14) **sync** – Synchronous, whenever a file is saved it will be directly written to the hard disk

15) **suit** – Allow set-user-identifier for the device where users are allowed to run binaries even though they do not have execute permissions. These binaries are temporarily made available to them to perform certain tasks

16) **nosuid** – Do not allow set-user-identifier

17) **defaults** – **auto, rw, dev, async, suid, exec & nouser**

5th Column :
~~~~~~~~~~

The 5th column is for backup option. This column contains either 0 or 1. Where "0" stands for "NO" and "1" stands for "YES". The system checks it at the time of booting, if it's "0", dump will ignore that filesystem but if its "1" then it will enable backup option.Backup is supported on only ext3 file system, hence only for ext3 file system it should be enabled and for rest of the file systems it should be disabled.

6th Column :
~~~~~~~~~~

The 6th column is for "fsck" option. fsck stands for file system check. This column defines the order in which the system should scan the partitions on start up. The / partition is assigned top priority i.e. 1 and the rest of the partitions are assigned second priority i.e. 2. If value is set to 0 means no scanning will be done at the time of startup.If same number is given to different partitions then the partitions are scanned together with equal priority. This minimizes error because if a link is present on one partition with higher priority and the source file in another partition with a priority lower than the link, it will give an error.

The *dmesg* command is used to write the *kernel messages* in Linux and other Unix-like operating systems to standard output (which by default is the display screen).

A kernel is the core of an operating system. It is the first part of the operating system that is loaded into memory when a computer boots up (i.e., starts up), and it controls virtually everything on a system. The numerous messages generated by the kernel that appear on the display screen as a computer boots up show the hardware devices that the kernel detects and indicate whether it is able to configure them.

dmesg obtains its data by reading the *kernel ring buffer*. A buffer is a portion of a computer's memory that is set aside as a temporary holding place for data that is being sent to or received from an external device, such as a hard disk drive (HDD), printer or keyboard. A *ring buffer* is a buffer of fixed size for which any new data added to it overwrites the oldest data in it.

dmesg can be very useful when troubleshooting or just trying to obtain information about the hardware on a system. Its basic syntax is `dmesg [options]`

Invoking dmesg without any of its options (which are rarely used) causes it to write all the kernel messages to standard output. This usually produces far too many lines to fit into the display screen all at once, and thus only the final messages are visible. However, the output can be redirected to the *less* command through the use of a pipe (designated by the vertical bar character), thereby allowing the startup messages to be viewed one screenful at a time:

```
dmesg | less
```

less allows the output to be moved forward one screenful at a time by pressing the SPACE bar, backward by pressing the *b* key and removed by pressing the *q* key. (The *more* command could have been used here instead of the *less* command; however, *less* is newer than *more* and has additional functions, including the ability to return to previous pages of the output.)

When a user encounters a problem with the system, it can be convenient to write the output of dmesg to a file and then send that file by e-mail to a system administrator or other knowledgeable person for assistance. For example, the output could be redirected to a file named *boot_messages* using the *output redirection operator* (designated by a rightward facing angle bracket) as follows:

```
dmesg > boot_messages
```

Because of the length of the output of dmesg, it can be convenient to pipe its output to grep, a filter which searches for any lines that contain the string (i.e., sequence of characters) following it. The *-i* option can be used to tell grep to ignore the *case* (i.e., lower case or upper case) of the letters in the string. For example, the following command lists all references to USB (universal serial bus) devices in the kernel messages:

```
dmesg | grep -i usb
```

And the following tells dmesg to show all serial ports (which are represented by the string *tty*):

```
dmesg | grep -i tty
```

The dmesg and grep combination can also be used to show how much *physical memory* (i.e., RAM) is available on the system:

```
dmesg | grep -i memory
```

The following command checks to confirm that the HDD(s) is running in DMA (direct memory access) mode:

```
dmesg | grep -i dma
```

The output of dmesg is maintained in the log file */var/log/dmesg*, and it can thus also be easily viewed by reading that file with a text editor, such as vi or gedit, or with a command such as cat, e.g.,

```
cat /var/log/dmesg | less
```

http://linuxgazette.net/issue59/nazario.html

lspci is a command on Unix-like operating systems that prints detailed information about all PCI buses and devices in the system. It is based on a common portable library *libpci* which offers access to the PCI configuration space on a variety of operating systems.

Example output on a Linux system:

```
# lspci
00:00.0 Host bridge: Intel Corporation 82815 815 Chipset Host
Bridge and Memory Controller Hub (rev 11)
00:02.0 VGA compatible controller: Intel Corporation 82815 CGC
[Chipset Graphics Controller] (rev 11)
00:1e.0 PCI bridge: Intel Corporation 82801 Mobile PCI Bridge (rev
03)
00:1f.0 ISA bridge: Intel Corporation 82801BAM ISA Bridge (LPC)
(rev 03)
00:1f.1 IDE interface: Intel Corporation 82801BAM IDE U100 (rev 03)
00:1f.2 USB Controller: Intel Corporation 82801BA/BAM USB (Hub #1)
(rev 03)
00:1f.3 SMBus: Intel Corporation 82801BA/BAM SMBus (rev 03)
00:1f.4 USB Controller: Intel Corporation 82801BA/BAM USB (Hub #2)
(rev 03)
00:1f.5 Multimedia audio controller: Intel Corporation 82801BA/BAM
AC'97 Audio (rev 03)
01:03.0 CardBus bridge: O2 Micro, Inc. OZ6933/711E1
CardBus/SmartCardBus Controller (rev 01)
01:03.1 CardBus bridge: O2 Micro, Inc. OZ6933/711E1
CardBus/SmartCardBus Controller (rev 01)
01:0b.0 PCI bridge: Actiontec Electronics Inc Mini-PCI bridge (rev
11)
02:04.0 Ethernet controller: Intel Corporation 82557/8/9 [Ethernet
Pro 100] (rev 08)
02:08.0 Communication controller: Agere Systems WinModem 56k (rev
01)
```

If many devices are shown as unknown (e.g. "Unknown device 2830 (rev 02)), issuing the command 'update-pciids' will usually do the trick.

Detail Information

```
$lspci -vv
```

To update pci-ids information to /usr/share/hwdata/pci.ids **$update-pciids**

Bash

Bash

Descended from the Bourne Shell, Bash is a GNU product, the "Bourne Again SHell." It's the standard command line interface on most Linux machines. It excels at interactivity, supporting command line editing, completion, and recall. It also supports configurable prompts - most people realize this, but don't know how much can be done.

Bash converts the text script to binary (0,1).

This chapter is based on Chapters 6 through 8 of the Siever book, *Linux in a Nutshell* [Siever 2003] .

Figure 1 illustrates some of the shells found on UNIX/Linux systems.

Shell	Description
bash	Bourne-again shell (GNU)
csh	C shell (BSD)
jsh	Job control shell (SVR4)
ksh	Korn shell (Bell Labs)
rc	Plan 9 shell (Bell Labs)
rsh	Remote shell (TCP/IP)
sh	Bourne shell (UNIX 7th Edition)
tcsh	Popular extension of the C shell
zsh	Popular extension of the Korn shell

Figure 1: Some UNIX/Linux Shells

Standard GNU/Linux systems use bash as the default shell. Some distributions, e.g. Red Hat Linux, have /bin/sh as a symbolic link to /bin/bash and /bin/csh as a symbolic link to /bin/tcsh.

Common Features

Figure 2 illustrates some features that are common to both bash and tcsh.

Symbol	Description
>	Redirect output
>>	Append output to a file
<	Redirect input
<<	Redirect input ("Here" document)

Symbol	Description
|	Pipe output
&	Run process in background
;	Separate commands on one line
*	Match character(s) in filename
?	Match single character in filename
!*n*	Repeat command number *n*
[...]	Match any characters enclosed
(...)	Execute commands in a subshell
"..."	Quote allowing variable and command expansion
'...'	Literal string
`...`	Command substitution
\	Quote following character
$*var*	Variable expansion
$$	Process ID
$0	Command name
$*n*	*n*th argument (0...9)
$*	All arguments
$?	Exit status
	Begin comment

Figure 2: Common symbols

In addition to these symbols, both shells have some common commands, as illustrated in Figure 3.

Command	Description
bg	Background execution
break	Break out of a loop
cd	Change directory
continue	Resume a loop
echo	Display output
eval	Evaluate arguments
exec	Execute a new program
fg	Foreground execution
jobs	Show active jobs
kill	Terminate running job(s)
shift	Shift positional parameters
stop	Suspend a background job
suspend	Suspend a foreground job
umask	Set or list file permissions
unset	Erase variable or function definition
wait	Wait for a background job to finish

Refrence

http://en.wikipedia.org/wiki/Bash

Practical

BASH

Login root

passwd *****

when you login u get vcs (virtual konsole) with the help of tty driver (/dev/tty*) and a shell (/bin/bash)

In Linux default shell is bash (/bin/bash)

To check the shells supported by your OS

$cat /etc/shells

To swap in other shell

$sh

$ksh etc....

check your bash

$ps

to start another bash just run the following, which is inherited

$bash

Now chk with

$ps you will have two bash (parent and child) if u will kill the child bash it wont effect to parent but if u do viceversa then chk what happens.

$List the bash shell pid

$ps -el | grep bash

Now try loading bash and can kill with the cmd

$kill -9 <pid of bash>

Bash

Here's a neat Bash-prompt trick. At a basic Bash prompt, press the up-arrow key and you'll see the last command you typed in. Press again and again to rotate through all the commands you typed previously, stored for you in Bash history.

You will only see the commands you typed in for your login, whether that's for a specific user or for root.

Here are some additional Bash tips, all of which are commands that you type at the Bash prompt:

To display a full numbered list of all stored commands, type:
history
To retrieve the eighth command previously entered, type:
!8

To get the last command that started with the letter V, type:
!v

Bash history isn't lost when you reboot or shutdown either. Clever isn't it?

Bash Shortcuts

To go along with Bash basics above, here are some basic shorthand commands:

To go back one step in the directory tree, type:

```
cd ..
```

To change to the /home/{logged in username} directory, type:

```
cd ~
```

To change to the directory of a specific user when you have more than one, type the previous command followed by the name of the user:

```
cd ~bruno
cd ~anna
```

To change the directory /home/{logged in username}/Downloads/Backgrounds, type:

```
cd ~/Downloads/Backgrounds
```

For really fast typing don't forget to use the Tab-key for auto-completion.

Typing the following does the same as the previous example, a lot faster:

```
cd ~/D {press Tab Key} /B {press Tab key}
```

Bash Script

You probably know that the "rm" command removes (or deletes) a file permanently. Wouldn't it be nice if we could move it to the recycle bin with a simple command instead? You can. To do that, you can make your own command called Del with a brief script.

To build the script, open a terminal and type the following lines:

```
su
{type your root password}  (Note: you should see the # prompt)
kedit /usr/bin/del
```

This opens a new window in the keditor into which you should type the following script:

```
#!/bin/bash

mv $1 ~/Desktop/Trash
#End script
```

The next step is to save the file using kedit's File, Save As menu command. Then, back at the Bash prompt logged in as root, type this line to make the new script executable:

```
$chmod 0775 /usr/bin/del
```

Now whenever you type the del command, it will run your script. For example, if you came across the "tessst" file and you wanted to move it to the trash, you could just type this at the Bash prompt:

```
$del tessst
```

That will perform the same action as:

```
$mv tessst /home/{logged in username}/Desktop/Trash
```

Sure this was a very short example, a three-line script, it only holds one command, but you could add as many lines to the script as

you wanted to and execute it with a simple three-letter word. If there are more commands in the script it will execute them in the order that they appear. Because /usr/bin is in your path you only have to type "del" to execute it from anywhere in the file system.

Tab Completion Tip

Did you know you can use the Tab key to auto-complete commands on the command line? Just type a few characters that start a command and press the Tab key. The command or name of an existing directory or file will be completed.
Try this. Type the following and then press the Tab key:
$ cd /u

Now add an "s" and press Tab, type "h" and press Tab. The result should be:
$ cd /usr/share/

Now type "f" "o" "n" and press Tab, "t" press Tab, "d" Tab, and press the Enter key. That should put you in:
/usr/share/fonts/ttf/decoratives

Type the following and press Enter:
ls

That'll bring up a list of all the fancy ttf fonts on your system. So next time you have to type a long command like this:
cp synthesis.hdlist.update_source.cz
/var/lib/urpmi/synthesis.hdlist.update_source.cz
... try it this way instead:

cp sy (Tab key), /v (Tab key), li (Tab key), u (Tab key), sy (Tab key)
And because the full command is on your screen, the light will go on if it hasn't already! (Note: This command works only if the file "synthesis.hdlist.update_source.cz" is in your /home directory)

How about a little more on the Tab key and commands. If you don't remember exactly how a command was written, type in the first character or two and hit the Tab key. You'll get a list of all the commands that start with the same character(s).
If you wish to know what a certain command does -- say, mkmanifest -- use the whatis command, like this:
$ whatis mkmanifest
mkmanifest (1) - Makes list of file names and their DOS 8+3 equivalents.

Introduction to BASH

* Developed by GNU project.
* The default Linux shell.
* Backward-compatible with the original sh UNIX shell.
* Bash is largely compatible with sh and incorporates useful features from the Korn shell ksh and the C shell csh.
* Bash is the default shell for Linux. However, it does runs on every version of Unix and a few other operating systems such as ms-

dos, os/2, and Windows platforms.

Quoting from the official Bash home page:

Bash is the shell, or command language interpreter, that will appear in the GNU operating system. It is intended to conform to the IEEE POSIX P1003.2/ISO 9945.2 Shell and Tools standard. It offers functional improvements over sh for both programming and interactive use. In addition, most sh scripts can be run by Bash without modification.
The improvements offered by BASH include:

The Bash syntax is an improved version of the Bourne shell syntax. In most cases Bourne shell scripts can be executed by Bash without any problems.

* Command line editing.
* Command line completion.
* Unlimited size command history.
* Prompt control.
* Indexed arrays of unlimited size (Arrays).
* Integer arithmetic in any base from two to sixty-four.
* Bash startup files - You can run bash as an interactive login shell, or interactive non-login shell. See Bash startup files for more information.
* Bash conditional expressions: Used in composing various expressions for the test builtin or [[or [commands.
* The Directory Stack - History of visited directories.
* The Restricted Shell: A more controlled mode of shell execution.
* Bash POSIX Mode: Making Bash behave more closely to what the POSIX standard specifies.

In Linux, a lot of work is done using a command line shell. Linux comes preinstalled with Bash. Many other shells are available under Linux:

* tcsh - An enhanced version of csh, the C shell.
* ksh - The real, AT&T version of the Korn shell.
* csh - Shell with C-like syntax, standard login shell on BSD systems.
* zsh - A powerful interactive shell.
* scsh- An open-source Unix shell embedded within Scheme programming language.

Shell Scripting

Starting a Script With #!

1. It is called a shebang or a "bang" line.
2. It is nothing but the absolute path to the Bash interpreter.
3. It consists of a number sign and an exclamation point character (#!), followed by the full path to the interpreter such as /bin/bash.
4. All scripts under Linux execute using the interpreter specified on a first line[1].
5. Almost all bash scripts often begin with #!/bin/bash (assuming that Bash has been installed in /bin)
6. This ensures that Bash will be used to interpret the script, even if it is executed under another shell[2].
7. The shebang was introduced by Dennis Ritchie between Version 7 Unix and 8 at Bell Laboratories. It was then also added to the BSD line at Berkeley [3].

Ignoring An Interpreter Line (shebang)

* If you do not specify an interpreter line, the default is usually the /bin/sh. But, it is recommended that you set #!/bin/bash line.

/bin/sh

For a system boot script, use /bin/sh:

#!/bin/sh

sh is the standard command interpreter for the system. The current version of sh is in the process of being changed to conform with the POSIX 1003.2 and 1003.2a specifications for the shell.
Did you know?

* It is the shell that lets you run different commands without having to type the full pathname to them even when they do not exist in the current directory.
* It is the shell that expands wildcard characters, such as * or ?, thus saving you laborious typing.
* It is the shell that gives you the ability to run previously run commands without having to type the full command again by pressing the up arrow, or pulling up a complete list with the history command.
* It is the shell that does input, output and error redirection.

Why shell scripting?

* Shell scripts can take input from a user or file and output them to the screen.
* Whenever you find yourself doing the same task over and over again you should use shell scripting, i.e., repetitive task automation.
    - Creating your own power tools/utilities.
    - Automating command input or entry.

- Customizing administrative tasks.
- Creating simple applications.
- Since scripts are well tested, the chances of errors are reduced while configuring services or system administration tasks such as adding new users.

Practical examples where shell scripting actively used

* Monitoring your Linux system.
* Data backup and creating snapshots.
* Dumping Oracle or MySQL database for backup.
* Creating email based alert system.
* Find out what processes are eating up your system resources.
* Find out available and free memory.

List of command bash keywords and built in commands

* JOB_SPEC &
* ((expression))
* . filename
* [[:]]
* [arg...]
* expression
* alias
* bg
* bind
* builtin
* caller
* case
* command
* compgen
* complete
* continue
* declare
* dirs
* disown
* echo
* enable
* eval
* exec
* exit
* export
* false
* fc
* fg

```
command1 && command2
```

OR

```
First_command && Second_command
```

command2 is executed if, and only if, command1 returns an exit status of zero (true). In other words, run command1 and if it is successfull, then run command2.
Example

Type the following at a shell prompt:

```
$rm /tmp/filename && echo "File deleted."
```

The echo command will only run if the rm command exits successfully with a status of zero. If file is deleted successfully the rm command set the exit stats to zero and echo command get executed.

Lookup a username in /etc/passwd file

```
grep "^champu" /etc/passwd && echo "champu found in /etc/passwd"
```

Exit if a directory /tmp/foo does not exist

```
test ! -d /tmp/foo && { read -p "Directory /tmp/foo not found. Hit [Enter] to exit..." enter; exit 1; }
```

Syntax:

```
command1 || command2
```

OR

```
First_command || Second_command
```

command2 is executed if, and only if, command1 returns a non-zero exit status. In other words, run command1 successfully or run command2.

Example

```
$cat /etc/shadow 2>/dev/null || echo "Failed to open file"
```

The cat command will try to display /etc/shadow file and it (the cat command) sets the exit stats to non-zero value if it failed to open /etc/shadow file. Therefore, 'Failed to open file' will be displayed cat command failed to open the file.

Find username else display an error

```
$grep "^champu" /etc/passwd || echo "User champu not found in /etc/passwd"
```

How Do I Combine Both Logical Operators?

Try it as follows:

```
$cat /etc/shadow 2>/dev/null && echo "File successfully opened." || echo "Failed to open file."
```

Make sure only root can run this script:

```
$test $(id -u) -eq 0  && echo "You are root" || echo "You are NOT root"
```

OR

```
$test $(id -u) -eq 0  && echo "Root user can run this script." ||
echo "Use sudo or su to become a root user."
```

Shell functions

* Sometime shell scripts get complicated.
* To avoid large and complicated scripts use functions.
* You divide large scripts into a small chunks/entities called functions.
* Functions makes shell script modular and easy to use.
* Function avoids repetitive code. For example, is_root_user() function can be reused by various shell scripts to determine whether logged on user is root or not.

* Function performs a specific task. For example, add or delete a user account.
* Function used like normal command.
* In other high level programming languages function is also known as procedure, method, subroutine, or routine.

Writing the hello() function

Type the following command at a shell prompt:

```
hello() { echo 'Hello world!' ; }
```

Invoking the hello() function

hello() function can be used like normal command. To execute, simply type:

```
hello
```

Passing the arguments to the hello() function

You can pass command line arguments to user defined functions. Define hello as follows:

```
hello() { echo "Hello $1, let us be a friend." ; }
```

You can hello function and pass an argument as follows:

```
hello champu
```

Sample outputs:

```
Hello champu, let us be a friend.
```

* One line functions inside { ... } must end with a semicolon. Otherwise you get an error on screen:

```
$xrpm() { rpm2cpio "$1" | cpio -idmv }
```

Above will not work. However, the following will work (notice semicolon at the end):

```
$xrpm() { rpm2cpio "$1" | cpio -idmv; }
```

To display defined function names use the declare command. Type the following command at a shell prompt:

```
$declare -f
```

Sample outputs:

```
declare -f command_not_found_handle
declare -f genpasswd
declare -f grabmp3
declare -f hello
declare -f mp3
declare -f xrpm
```

Display Function Source Code

To view function names and source code, enter:

```
declare  -f
```

OR

```
declare -f | less
```

The test command is used to check file types and compare values. Test is used in conditional execution. It is used for:

* File attributes comparisons
* Perform string comparisons.
* Arithmetic comparisons.

test command syntax

```
test condition
```

OR

```
test condition && true-command
```

OR

```
test condition || false-command
```

OR

```
test condition && true-command || false-command
```

Type the following command at a shell prompt (is 5 greater than 2?):

```
$test 5 > 2 && echo "Yes"
$test 1 > 2 && echo "Yes"
```

Sample Output:

```
Yes
Yes
```

Rather than test whether a number is greater than 2, you have used redirection to create an empty file called 2 (see shell redirection). To test for greater than, use the -gt operator (see numeric operator syntax):

```
test 5 -gt 2 && echo "Yes"
test 1 -gt 2 && echo "Yes"
```

```
Yes
```

You need to use the test command while make decision. Try the following examples and note down its output:

```
$test 5 = 5 && echo Yes || echo No
$test 5 = 15 && echo Yes || echo No
$test 5 != 10 && echo Yes || echo No
$test -f /etc/resolv.conf && echo "File /etc/resolv.conf found." ||
echo "File /etc/resolv.conf not found."
test -f /etc/resolv1.conf && echo "File /etc/resolv1.conf found."
|| echo "File /etc/resolv1.conf not found."
```

Write Scripts

1.

```
#!/bin/bash
read -p "Enter # 5 : " number
if test $number == 5
then
    echo "Thanks for entering # 5"
fi
if test $number != 5
then
    echo "I told you to enter # 5. Please try again."
fi
```

2.

```
#!/bin/bash
clear
echo -e "What is your name : \c"
read name
echo hello $name. Welcome to Shell programming

sleep 2
clear
echo -e "Would you like to see a listing of your files ? [y/n]: \c"
read yn
if [ $yn = y ]
```

```
then
ls
fi


sleep 1
echo -e "Would you like to see who all are logged in  ? [y/n]: \c"
read yn
if [ $yn = y ]
then

who
fi

sleep 1
echo Would you like to see which dir you are in \?
read yn
if [ $yn = y ]
then
   pwd
fi
```

3.

```
#!/bin/sh
clear
echo Enter file name to copy
read apple
echo Enter file name to copy to
read mango
if cp $apple $mango > /dev/null 2>&1
then
     echo Files copied ok Congrats!!
else
        echo Error !!!!!!!!!!!!!! Contact Mr ABC at Ext 101
fi
```

4.

```
#!/bin/bash
# -lt, -le, -gt, -ge, -ne, -eq   : Use this for numerical
comparisions
#  <, <=, >, >=, <>, =           : Use this for String comparisions
clear
tput cup 10 10
echo -e "Enter a no from 1 to 5 : \c"
read num
if test $num -lt 6
then
   tput cup 12 10
   echo "Good"
else
   tput cup 12 10
   echo "Sorry only between 1 to 6"
fi
```

5.

```
#!/bin/bash
## see man test
```

```
clear
echo Enter file name
read filename
if [ -z $filename ]
then
     echo You have to enter some file name
     echo Exiting....
     sleep 2
     exit
fi
if [ -f $filename ]
then
     echo The filename you entered exists !!
     echo Deleting $filename .....
     sleep 2
     rm -f $filename
     echo Deleted $filename .....
     sleep 1
     cls
else
     echo The filename you entered does not exist !!!
fi
```

6.

```
#!/bin/bash
read -p "Enter a number : " n
if [ $n -gt 0 ]; then
  echo "$n is a positive."
elif [ $n -lt 0 ]
then
  echo "$n is a negative."
elif [ $n -eq 0 ]
then
  echo "$n is zero number."
else
  echo "Oops! $n is not a number."
fi
```

7.

```
#!/bin/bash

clear
echo -e "Enter a number from 1 to 3 : \c"
read num

case $num in
     1) echo You have entered 1
        ;;
     2) echo You have entered 2
        ;;
     3) echo You have entered 3
        ;;
     *) echo Between 1 to 3 only !!
```

```
        ;;
esac
```

8.

```
#!/bin/sh
echo Enter dog/cat/parrot
read animal
case $animal in
     cat|kat) echo You have entered cat
     ;;
     dog) echo You have entered dog
     ;;
     parrot|crow) echo You have entered parrot or crow
     ;;
     *) echo Invalid entry !!
     ;;
esac
```

RPM

Rpm is a powerful Package Manager for Red Hat, Suse and Fedora Linux. It can be used to build, install, query, verify, update, and remove/erase individual software packages. A Package consists of an archive of files, and package information, including name, version, and description:

The RPM Package Manager

RPM is a recursive acronym for RPM Package Manager.
It used to be called the Red Hat Package Manager, but Red Hat changed its name to emphasis that other distributions use it too. The new official name is RPM Package Manager, and yes, that’s a self-referencing acronym (SRA), just like GNU.

- RPM is the default package manager for Red Hat Linux systems.
- RPM system consists of a local database, the rpm executable, rpm package files.
- It deals with .rpm files, which contain the actual programs as well as various bits of meta-information about the package: what it is, where it came from, version information and info about package dependencies.
- RPMs are the files (called packages) which contain the installable software; typically they have
the .rpm suffix.

RPM FACTS

---

1. RPM is free - GPL

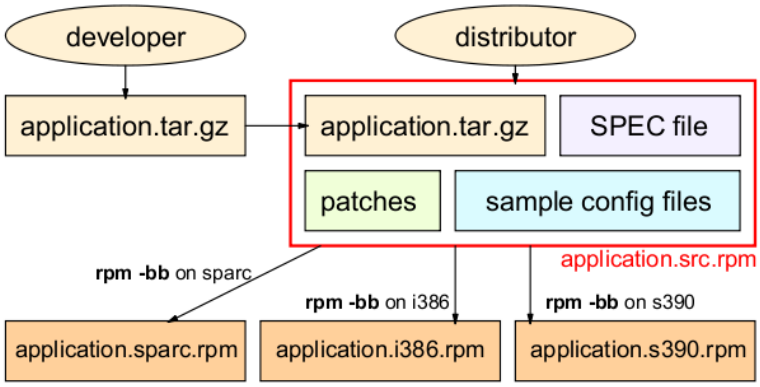
The RPM Package Manager or RPM is a tool which was developed by Red Hat Software, who still maintain it, but released under the GNU General Public Licence (GPL) and has proven to be so popular, that a lot of other distribution manufacturers use it as well.

RPM is a very versatile program which solves a lot of problems that a distributor of software typically faces:
• Management of source files
• Management of the build process
• A distribution method and format for binary files, including pre- and
postinstall scripts. RPMs can be created by anyone, not only the manufacturer of your distribution.

2. stores info about packages in a database /var/lib/rpm

/var/lib/rpm contains all the database necessary for managing all of the packages installed on your system in the form of rpm
The database stores information about installed packages such as file attributes and package prerequisites.

When a certain system uses RPMs to install packages, a database of installed packages is stored in /var/lib/rpm. The database itself is in rpm format too, so it cannot be read directly. You will have to access the database using the rpm command.



Note: RPM v4 uses **rpmbuild** instead of **rpm** for building RPMs

Where to get RPMs

–
http://rpmseek.com
http://rpmfind.net
http://www.redhat.com
http://freshrpms.net
http://rpm.pbone.net
http://dag.wieers.com
http://rpmforge.net
http://filewatcher.com

Common Build Procedures

- source code install – tarball (.tar, .tar.gz, .tgz, tar.bz, tar.tbz)
- Configure/make/make install

- Binary RPMs (.rpm)
- Source RPMs (.srpm)

Some Query Options

```
$ rpm -ivh {rpm-file}    Install the package
$ rpm -ivh mozilla-mail-1.7.5-17.i586.rpm
$ rpm -ivh --test mozilla-mail-1.7.5-17.i586.rpm
$ rpm -Uvh {rpm-file}    Upgrade package
$ rpm -Uvh mozilla-mail-1.7.6-12.i586.rpm
$ rpm -Uvh --test mozilla-mail-1.7.6-12.i586.rpm
$ rpm -Fvh upgrades to a later version
$ rpm -ev {package}      Erase/remove/ an installed package
$ rpm -ev mozilla-mail
$ rpm -ev --nodeps {package} Erase/remove/ an installed package
without checking for dependencies
$ rpm -ev --nodeps mozilla-mail
$ rpm -qa      Display list all installed packages    rpm -qa
$ rpm -qa | less
$ rpm -qi {package}      Display installed information along with
package version and short description
$ rpm -qi mozilla-mail
$ rpm -qf {/path/to/file}    Find out what package a file belongs
to i.e. find what package owns the file
$ rpm -qf /etc/passwd
$ rpm -qf /bin/bash
$ rpm -qc {pacakge-name}     Display list of configuration file(s)
for a package
$ rpm -qc httpd
$ rpm -qcf {/path/to/file}   Display list of configuration files
for a command
$ rpm -qcf /usr/X11R6/bin/xeyes
$ rpm -qa --last    Display list of all recently installed RPMs
$ rpm -qa --last
$ rpm -qa --last | less
$ rpm -qpR {.rpm-file}
$ rpm -qR {package}      Find out what dependencies a rpm file has
$ rpm -qpR mediawiki-1.4rc1-4.i586.rpm
$ rpm -qR bash
$ rpm -qlp foo.rpm Which files are installed with foo.rpm?
$ rpm -ivh --nodeps pants.rpm Installing package Ignoring
Dependencies
$ rpm -e foo ('e' for erase)
$ rpm -i --prefix /new/directory package.rpm The --prefix and --
relocate options should make the rpm command relocate a package to
a new
location.
$ rpm -k <.rpm> we could verify the MD5 is OK
$ rpm --rebuilddb
```

Task

Download xmms-1.2.10-1.i386.rpm & try to install

```
$ rpm -ivh xmms-1.2.10-1.i386.rpm
```

Will ask for dependency, Download dep from the given sites above & install the same.

Download

glib-1.2.10-627.i586.rpm
gtk-1.2.10-926.i586.rpm
gtk-32bit-1.2.10-926.x86_64.rpm

UserAdministration

Only root (i.e. system administrator)can use adduser command
To create new users. It is not allow to other users.

Adduser is symlink of Useradd which is binary in /usr/sbin.
We(root)can
 customise adduser by using another word(champu) & make it
 symlink of useradd.
 Let's see

```
[root@localhost root]$ cd /usr/sbin
[root@localhost sbin]$ ln -s useradd uad
```

Now uad is symlink of useradd.

There are 3 types of users

<1> Superuser : At the time of linux installation it is create.
 He has right to make other users & his`userid'& `groupid' is zero
in `/etc/Passwd' file.

 <2> Systemuser: These users create by System. They can't login
 becoz their shell `sbin/noloin' is default in seventh field in
`/etc/passwd' file.

 <3> Normaluser: These users create by superuser.

 Let's see how superuser make normaluser :

```
[root@localhost root]$ adduser john
[root@localhost root]$ passwd john
Changing password for user john.
New password:(user password)
BAD PASSWORD: it is too short (if password is less than six
character but it doesn't affect so no need to worry)
Retype new password:(user password)
Passwd: all authentication tokens updated succesfully.
```

[root@localhost root]$ userdel john ---> `userdel' command delete

only name of the user from
/home directory but it's
data remain there. It's
/usr/sbin/userdel

[root@localhost root]$ userdel -r john
 ---->userdel -r delete name of user as well as data.

[root@localhost root]$ usermod -G groupname username
i.e.
[root@localhost root]$ usermod -G john eric

 ---->`usermod -G' command makes the user eric member
 of the group john. /usr/sbin/usermod.

 su ----> with the help of this command root can work as
 substitute user.

 su -r ---->with the help of this command root come out from
 subtitute user.

 The information of adduser refers 2 files & updates 4 files.

Config. files

Refers
|----/etc/login.defs
|
|----/etc/default/useradd

Updates
|----/etc/passwd
|
|----/etc/group
|
|----/etc/shadow
|
|----/etc/gshadow

/etc/login.defs
<1> /etc/login.defs : It keep the information of directory where mailboxes reside or name of file relative to the home directory, Password duration & how many users can login.

"Passwd file" & "Group file" get the information of userid & groupid from this file.
"shadow file" & "Gshadow file" get the information of user login & password duration of user from this file.

Min/max values for automatic uid selection in useradd.

UID-MIN 500
UID-MAX 60000

The id of user start from 500 & max it is 60000 which is default according to REDHAT but we can customise it.

If there are two department ACCOUNTANT & MARKETING in one office then I can start userid to ACCOUNTANT from 1000 & to MARKETING from 2000 which is reliable.

Similar way to Groupid

GID-MIN 500
GID-MAX 60000

PASSWORD AGING CONTROLS:

1. PASS-MAX-DAYS 99999 : The maximum number of days a password can be used. i.e max 99999 days.
2. PASS-MIN-DAYS 0 : The minimum number of days allowed between password can change.
3. PASS-MIN-LEN 5 : The minimum length of the password. i.e. 5 character.
4. PASS-WARN-AGE 7 : Specifies the number of days warning given to user before the password expire. ie 7 days.

The above PASSWORD AGING information is default according to REDHAT which we can customise it.

/etc/default/useradd

<2> /etc/default/useradd : It has information of no. of groups, directory
of users & user using which shell in following way.

1. Group=100 ----> It's default no. of groups according to Redhat which can customise.

2. Home=/home ----> It's default dir of user as Redhat say to which we can give any name i.e. we can make `ghar'instead of `home' by making directory under /

3. Inactive ----> It's number of days after password expire of user.

4. Expire ----> It's number of days for the account of user will expire.

5. Shell=/bin/bash --> It's path of user shell.
Skel=/etc/skel ---> When user create there is zero dir or file but when give command `l.' it shows some hidden files which comes from /etc/skel.

/etc/passwd

<3> /etc/passwd : * It keeps the record of new user when create by superuser. Each line is entry of new user. It is text file & has details of all system users.

* It has 7 fields for each user in each line so it is called `system passwd database' & each field is separted : (colon) also called "Internal field separator".

```
champu:x:500:500::/home/champu:/bin/bash
\____/\_/\__/\_/||\____________/\______/
   |   |   |  | ||     |          |
   1   2   3  4 5      6          7
```

1. field (champu) : It is username

2. field (x) : It contain user password which is somewhere else if exist.
If we put * inplace of x then user can't login.
If we keep second field blank then user can login without password.
i.e. (x) --- password somewhere else.
(*) --- user can't login.
( ) --- user can login without passwd.

3. field (500) : It contain userid which is unique. Further userid's are just one greater than last user.

4. field (500) : It contain groupid which is always same as userid. It's group of users.

5. field () : It is comment field or GECOS(General electric compressive operating system) user can keep his information by using command `chfn'in this field such as

```
$ chfn
Name []:
office []:
office phone []:
Home phone []:
```

6. field (/home/champu) : It's home of champu. /home is directory where all users store.

7. field (/bin/bash) : It contain the full path of shell used by user. Through shell we can convert shell script into binary format & whatever get from kernal convert into text format.

/etc/group

<4> /etc/group : This file keep the information of group. It has four field of each group of each line so it is called `system group database'.

Member of group has right to enter other member's of system who is member of same group.

line in this field like follow

```
Accounts:x:500:
    |    |  |  |
    1    2  3  4
```

1. field (accounts) : It contain name of group which is always same as the first member username.

2. field (x) : It contain group password which is somewhere else if exist & it's password is same of first member of group.

3. field (500) : It contain group id which is same of first member's id of group.

4. field : It contains list of members of group. By default Redhat it is blank but user can fill it by put the name of members of group.

One user can makes members of his group by using command `usermod -G' which is run by only root.

```
$usermod -G groupname username
when system admin first time creates users he can  send message
like `Thanku for using redhat linux'  through this & user get this
mail whenever he login.
```

Command line options

Option	**Description**
-c *comment*	Comment for the user
-d *home-dir*	Home directory to be used instead of default /home/*username*/
-e *date*	Date for the account to be disabled in the format YYYY-MM-DD
-f *days*	Number of days after the password expires until the account is disabled. (If 0 is specified, the account is disabled immediately after the password expires. If -1 is specified, the account is not be disabled after the password expires.)
-g *group-name*	Group name or group number for the user's default group (The group must exist prior to being specified here.)
-G *group-list*	List of additional (other than default) group names or group numbers, separated by commas, of which the user is a member. (The groups must exist prior to being specified here.)
-m	Create the home directory if it does not exist
-M	Do not create the home directory
-n	Do not create a user private group for the user
-r	Create a system account with a UID less than 500 and without a home directory
-p *password*	The password encrypted with crypt
-s	User's login shell, which defaults to /bin/bash
-u *uid*	User ID for the user, which must be unique and greater than 499

```
groupadd <group-name>
```

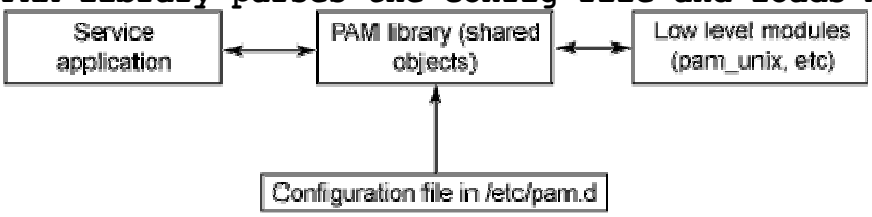
Command line options

Option	Description
-g *gid*	Group ID for the group, which must be unique and greater than 499
-r	Create a system group with a GID less than 500
-f	Exit with an error if the group already exists (The group is not altered.) If -g and -f are specified, but the group already exists, the -g option is ignored

Password aging
$chage -l root
$chage -d 0 *username*
Change shell
$chsh <username>
FingerInformation
$chfn <username>
$finger

PAM

PAM library parses the config file and loads modules to it



What operating systems support PAM?

PAM was first developed by Sun Microsystems in 1995 and is supported by the following operating system versions (and higher):

RedHat 5.0
SUSE 6.2
Debian 2.2
Mandrake 5.2
Caldera 1.3
TurboLinux 3.6

PAM is the Pluggable Authentication Module, invented by Sun. It's a beautiful concept, but it can be confusing and even intimidating at first. We're going to look at it on a RedHat system, but other Linuxes will be similar - some details may vary, but the basic ideas will be the same.

The first thing to understand is that PAM is NOT something like tcpd (tcp wrappers) or xinetd that encloses and restricts access to some service. An application needs to be "PAM aware"; it needs to have been written and compiled specifically to use PAM. There are tremendous advantages in doing so, and most applications with any interest in security will be PAM aware.

PAM is about security - checking to see that a service should be used or not. Most of us first learned about PAM when we were told that login was using it, but PAM can do much more than just validate passwords. A lot of applications now use PAM - even things like SAMBA can call on PAM for authentication.

The big advantage here is that security is no longer the application's concern: if PAM says its OK, its OK. That makes things easier for the application, and it makes things easier for the system administrator. PAM consults text configuration files to see what security actions to take for an application, and the administrator can add and subtract new rules at any time. PAM is also extensible: should someone invent a device that can read your brain waves and determine ill intent, all we need is a PAM module that can use that device. Change a few files, and login now reads your mind and grants or denies access appropriately. We're a bit away from that feature, but there are a tremendous number of available PAM modules that administrators can use.

Configuration Files

On modern RedHat systems, the configuration files are found in /etc/pam.d, one file for each PAM aware application (plus a special "other" file we'll get to later). One word of warning: changes to these files take effect instantly. You aren't going to get logged out if you make a mistake here. but if you DO screw up and blithely log out, you may not be able to log back in. So test changes before you exit.

We're going to use a very simple example to get started here. In a number of articles here, we've talked about SSH Security. Most of those articles have been about changes to ssh's configuration files, but here we'll use PAM to add some additional restriction: the time of day you are allowed to use ssh. To do this, we need a PAM module called pam_time.so - it's probably in your /lib/security/ directory already. It uses a configuration file "/etc/security/time.conf". That file is pretty well commented, so I'm not going to go into detail about it and will just say that I added the line

```
sshd;*;*;!Al2200-0400
```

which says that sshd cannot be used between 10:00 PM and 4:00 AM. I'm usually rather soundly asleep between those times, so why let ssh be used? I could still login at the console if I woke up with an urgent need to see an ls of my /tmp directory, but I couldn't ssh in, period. Configuring the time.conf file by itself doesn't

affect ssh; we need to add the pam module to /etc/pam.d/sshd. My file ends up looking like this:

```
#%PAM-1.0
account    required     pam_time.so
auth       required     pam_stack.so service=system-auth
auth       required     pam_nologin.so
account    required     pam_stack.so service=system-auth
password   required     pam_stack.so service=system-auth
session    required     pam_stack.so service=system-auth
session    required     pam_limits.so
session    optional     pam_console.so
```

I put the time.so module first so that it is the very first thing that is checked. If that module doesn't give sshd a green light, that's the end of it: no access. That's the meaning of "required": the module HAS to say that it is happy. The "account" type is specified here. That's a bit of a confusing thing: we have "account", "auth", "password" and "session". The man page isn't all that helpful:

> account - provide account verification types of service: has the user's password expired?; is this user permitted access to the requested service?
>
> authentication - establish the user is who they claim to be. **Typically** this is via some challenge-response request that the user must satisfy: if you are who you claim to be please enter your password. Not all authentications are of this type, there exist hardware based authentication schemes (such as the use of smart-cards and biometric devices), with suitable modules, these may be substituted seamlessly for more standard approaches to authentication - such is the flexibility of **Linux-PAM**.
>
> **password -** this group's responsibility is the task of updating authen-tication mechanisms. Typically, such services are strongly coupled to those of the auth group. Some authentication mechanisms lend themselves well to being updated with such a function. Standard UN*X password-based access is the obvious example: please enter a replacement pass-word.
>
> **session -** this group of tasks cover things that should be done prior to a service being given and after it is withdrawn. Such tasks include the maintenance of audit trails and the mounting of the user's home direc- tory. The session management group is important as it provides both an opening and closing hook for modules to affect the services available to a user.

I think that the distinction between account and session in that

man page is a little confusing. I think it would be quite reasonable to think you should use "session" for this module. Now, sometimes you have a man page for the module that shows you what to use, but pam_time doesn't help us there. Technically, it's not up to the library: the application is the one that is checking with account or session, but keep this in mind: session happens AFTER authentication. I liked the older PAM manual better, which said:

> auth modules provide the actual authentication, perhaps asking
> for and checking a password, and they set "credentials" such as group membership or kerberos "tickets."
>
> account modules check to make sure that the authentication is allowed (the account has not expired, the user is allowed to log in at this time of day, and so on).
>
> password modules are used to set passwords.
>
> session modules are used once a user has been authenticated to allow them to use their account,perhaps mounting the user's home directory or making their mailbox available.

For me, that was more clear.

Stacking

In this case, I only wanted to apply this restriction to ssh. If I'm physically at the box, I want no time restrictions. If I DID want these same restrictions, I'd make the same change to /etc/pam.d/login. But what if there are a whole bunch of things I want to apply the same rules to? RedHat has a special module "pam_stack". It functions much like an "include" statement in any programming language. We saw it in my /etc/pamd/sshd file:

```
auth       required     pam_stack.so service=system-auth
```

That says to look in /etc/pam.d/system-auth for other modules to use. Both login and sshd have this line (as does just about every other file in /etc/pam.d/), so we can look in system-auth to see what gets called by them:

```
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.

auth        required      /lib/security/$ISA/pam_env.so
auth        sufficient    /lib/security/$ISA/pam_unix.so
likeauth nullok
auth        required      /lib/security/$ISA/pam_deny.so
auth required /lib/security/$ISA/pam_tally.so
no_magic_root onerr=fail
account     required      /lib/security/$ISA/pam_unix.so
account     required      /lib/security/$ISA/pam_tally.so
onerr=fail file=/var/log/faillog deny=1 no_magic_root
even_deny_root_account
```

```
password    required      /lib/security/$ISA/pam_cracklib.so
retry=3 type=
password    sufficient    /lib/security/$ISA/pam_unix.so nullok
use_authtok md5 shadow
password    required      /lib/security/$ISA/pam_deny.so

session     required      /lib/security/$ISA/pam_limits.so
session     required      /lib/security/$ISA/pam_unix.so
```

Therefor, if we really wanted our time restrictions to apply to just about everything, we could add it to system-auth. Note the warning about authconfig though, and also consider that you will be making sudden sweeping changes to a LOT of applications and services.

Other

What if a PAM aware app doesn't have a file in /etc/pam.d? In that case, it uses the "other" file, which looks like this by default:

```
#%PAM-1.0
auth     required       /lib/security/$ISA/pam_deny.so
account  required       /lib/security/$ISA/pam_deny.so
password required       /lib/security/$ISA/pam_deny.so
session  required       /lib/security/$ISA/pam_deny.so
```

That "deny" module is a flat-out no access, red light, stop you dead right here module that is always going to say no. That's excellent from a security point of view, but can be a bit harsh should you accidentally delete something like "login". Login would now use the "other" file, and you couldn't login. That could be unpleasant.

There are many, many useful and clever PAM modules. While our brain wave interpreter doesn't exist yet, many other possibilities are available to you. There are modules to automatically black list hosts that have many failed logins, and much more. See http://www.kernel.org/pub/linux/libs/pam/modules.html.

Use of pam_listfile.so module

This PAM module authenticates users based on the contents of a specified file. For example, if username exists in a file /etc/sshd/ssh.allow, sshd will grant login access.

How do I configure pam_listfile.so module to deny access?

You want to block a user, if user-name exists in a file /etc/sshd/sshd.deny file.

Open /etc/pam.d/ssh (or /etc/pam.d/sshd for RedHat and friends)
vi /etc/pam.d/ssh

Append following line:
auth required pam_listfile.so item=user sense=deny
file=/etc/sshd/sshd.deny onerr=succeed

Save and close the file

Now add all usernames to /etc/sshd/sshd.deny file. Now a user is denied to login via sshd if they are listed in this file:
vi /etc/sshd/sshd.deny

Append username per line:
user1
user2
...

Restart sshd service:
/etc/init.d/sshd restart

Understanding the config directives:

- **auth required pam_listfile.so** : Name of module required while authenticating users.
- **item=user** : Check the username
- **sense=deny** : Deny user if existing in specified file
- **file=/etc/sshd/sshd.deny** : Name of file which contains the list of user (one user per line)
- **onerr=succeed** : If an error is encountered PAM will return status PAM_SUCCESS.

How do I configure pam_listfile.so module to allow access?

You want to ALLOW a user to use ssh, if user-name exists in a file /etc/sshd/sshd.allow file.
Open /etc/pam.d/ssh (or /etc/pam.d/sshd for RedHat and friends)
vi /etc/pam.d/ssh

Append following line:
auth required pam_listfile.so item=user sense=allow
file=/etc/sshd/sshd.allow onerr=fail

Save and close the file.

Now add all usernames to /etc/sshd/sshd.allow file. Now a user is allowed to login via sshd if they are listed in this file.
vi /etc/sshd/sshd.allow

Append username per line:
tony
om
rocky

Restart sshd service (optional):
/etc/init.d/sshd restart

Now if paul try to login using ssh he will get an error:
Permission denied (publickey,keyboard-interactive).

Following log entry recorded into my log file (/var/log/secure or

/var/log/auth.log file)
tail -f /var/log/auth.log

Output:

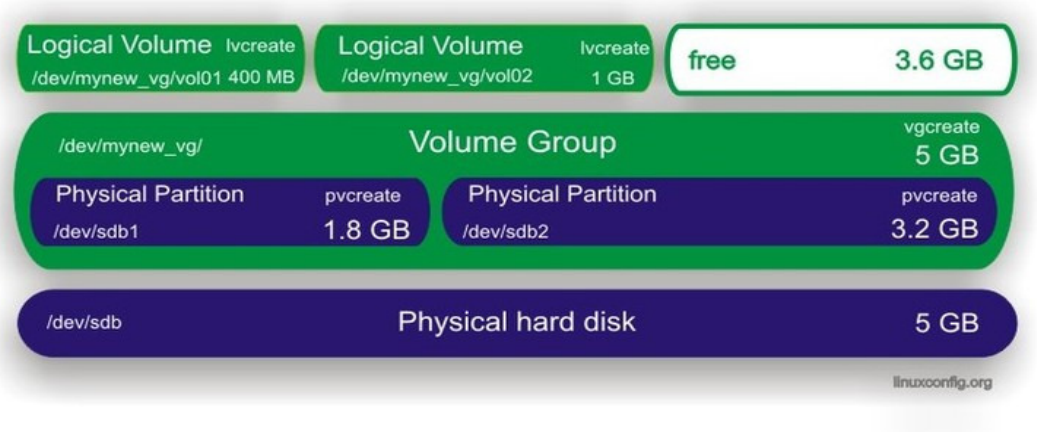
Jul 30 23:07:40 p5www2 sshd[12611]: PAM-listfile: Refused user paul for service ssh
Jul 30 23:07:42 p5www2 sshd[12606]: error: PAM: Authentication failure for paul from 125.12.xx.xx

Understanding the config directives:

8. **auth required pam_listfile.so** : Name of module required while authenticating users.
9. **item=user** : Check or specify the username
10. **sense=allow** : Allow user if existing in specified file
11. **file=/etc/sshd/sshd.allow** : Name of file which contains the list of user (one user per line)
12. **onerr=fail** : If filename does not exists or username formatting is not coreect it will not allow to login.

http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/
http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/Linux-PAM_MWG.html

LVM



Create Partitions

For this Linux lvm example you need an unpartitioned hard disk /dev/sdb. First you need to create physical volumes. To do this you

need partitions or a whole disk. It is possible to run pvcreate command on /dev/sdb, but I prefer to use partitions and from partitions I later create physical volumes.

```
linuxconfig.org# fdisk -l

Disk /dev/sda: 4294 MB, 4294967296 bytes
255 heads, 63 sectors/track, 522 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1          25      200781   83  Linux
/dev/sda2              26         522     3992152+   5  Extended
/dev/sda5              26         217     1542208+  83  Linux
/dev/sda6             218         299      658633+  83  Linux
/dev/sda7             300         327      224878+  82  Linux swap / Solaris
/dev/sda8             328         342      120456   83  Linux
/dev/sda9             343         522     1445818+  83  Linux

Disk /dev/sdb: 5368 MB, 5368709120 bytes
255 heads, 63 sectors/track, 652 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Disk /dev/sdb doesn't contain a valid partition table
linuxconfig.org# cfdisk /dev/sdb
```

Use your preferred partitioning tool to create partitions. In this example I have used cfdisk.

```
page for additional information.
linuxconfig.org# fdisk -l

Disk /dev/sda: 4294 MB, 4294967296 bytes
255 heads, 63 sectors/track, 522 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1          25      200781   83  Linux
/dev/sda2              26         522     3992152+   5  Extended
/dev/sda5              26         217     1542208+  83  Linux
/dev/sda6             218         299      658633+  83  Linux
/dev/sda7             300         327      224878+  82  Linux swap / Solaris
/dev/sda8             328         342      120456   83  Linux
/dev/sda9             343         522     1445818+  83  Linux

Disk /dev/sdb: 5368 MB, 5368709120 bytes
255 heads, 63 sectors/track, 652 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1         243     1951866   83  Linux
/dev/sdb2             244         652     3285292+  83  Linux
linuxconfig.org# []
```

Partitions are ready to use.

3. Create physical volumes

Use the pvcreate command to create physical volumes.

```
$ pvcreate /dev/sdb1
$ pvcreate /dev/sdb2
```

The pvdisplay command displays all physical volumes on your system.

```
$ pvdisplay
```

Alternatively the following command should be used:

```
$ pvdisplay /dev/sdb1
```

```
linuxconfig.org# pvcreate /dev/sdb1
  Physical volume "/dev/sdb1" successfully created
linuxconfig.org# pvcreate /dev/sdb2
  Physical volume "/dev/sdb2" successfully created
linuxconfig.org# pvdisplay
  --- NEW Physical volume ---
  PV Name               /dev/sdb1
  VG Name
  PV Size               1.86 GB
  Allocatable           NO
  PE Size (KByte)       0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               rFn4cW-L1QZ-ZfIG-rPxg-Wtw0-8XFf-ROypxN

  --- NEW Physical volume ---
  PV Name               /dev/sdb2
  VG Name
  PV Size               3.13 GB
  Allocatable           NO
  PE Size (KByte)       0
  Total PE              0
  Free PE               0
  Allocated PE          0
  PV UUID               FqDSca-6z8F-RbI7-6apo-31LN-iCRp-7J2vng

linuxconfig.org#
```

4. Create Virtual Group

At this stage you need to create a virtual group which will serve as a container for your physical volumes. To create a virtual group with the name "mynew_vg" which will include /dev/sdb1 partition, you can issue the following command:

```
$ vgcreate mynew_vg /dev/sdb1
```

To include both partitions at once you can use this command:

```
$ vgcreate mynew_vg /dev/sdb1 /dev/sdb2
```

```
linuxconfig.org# vgcreate mynew_vg /dev/sdb1
  Volume group "mynew_vg" successfully created
linuxconfig.org# vgdisplay
  --- Volume group ---
  VG Name               mynew_vg
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                1
  Act PV                1
  VG Size               1.86 GB
  PE Size               4.00 MB
  Total PE              476
  Alloc PE / Size       0 / 0
  Free  PE / Size       476 / 1.86 GB
  VG UUID               OtLsp7-dMvt-G60t-qHM0-Ev3W-YgmY-ZjhZOU

linuxconfig.org#
```

Feel free to add new physical volumes to a virtual group by using the vgextend command.

```
$ vgextend mynew_vg /dev/sdb2
```

```
linuxconfig.org# vgextend mynew_vg /dev/sdb2
  Volume group "mynew_vg" successfully extended
linuxconfig.org# vgdisplay
  --- Volume group ---
  VG Name               mynew_vg
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  2
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               4.99 GB
  PE Size               4.00 MB
  Total PE              1277
  Alloc PE / Size       0 / 0
  Free  PE / Size       1277 / 4.99 GB
  VG UUID               OtLsp7-dMvt-G60t-qHM0-Ev3W-YgmY-ZjhZOU

linuxconfig.org#
```

5. Create Logical Volumes

From your big cake (virtual group) you can cut pieces (logical volumes) which will be treated as a partitions for your linux system. To create a logical volume, named "vol01", with a size of 400 MB from the virtual group "mynew_vg" use the following command:

- create a logical volume of size 400 MB -L 400
- create a logical volume of size 4 GB -L 4G

```
$ lvcreate -L 400 -n vol01 mynew_vg
```

```
linuxconfig.org# lvcreate -L 400 -n vol01 mynew_vg
  Logical volume "vol01" created
linuxconfig.org# lvdisplay
  --- Logical volume ---
  LV Name                /dev/mynew_vg/vol01
  VG Name                mynew_vg
  LV UUID                CVolJV-4oN7-uBga-OeWB-TUOp-dem3-OBxD1G
  LV Write Access        read/write
  LV Status              available
  # open                 0
  LV Size                400.00 MB
  Current LE             100
  Segments               1
  Allocation             inherit
  Read ahead sectors     0
  Block device           254:0

linuxconfig.org#
```

With a following example you will create a logical volume with a size of 1GB and with the name vol02:

```
$ lvcreate -L 1000 -n vol02 mynew_vg
```

```
linuxconfig.org# lvcreate -L 1G -n vol02 mynew_vg
  Logical volume "vol02" created
linuxconfig.org# lvdisplay
  --- Logical volume ---
  LV Name                /dev/mynew_vg/vol01
  VG Name                mynew_vg
  LV UUID                CVolJV-4oN7-uBga-OeWB-TUOp-dem3-OBxD1G
  LV Write Access        read/write
  LV Status              available
  # open                 0
  LV Size                400.00 MB
  Current LE             100
  Segments               1
  Allocation             inherit
  Read ahead sectors     0
  Block device           254:0

  --- Logical volume ---
  LV Name                /dev/mynew_vg/vol02
  VG Name                mynew_vg
  LV UUID                RQbUjW-wFuV-hZo9-ZFET-52ks-AJD8-Xe6jzk
  LV Write Access        read/write
  LV Status              available
  # open                 0
  LV Size                1.00 GB
  Current LE             256
  Segments               1
  Allocation             inherit
  Read ahead sectors     0
  Block device           254:1

linuxconfig.org# []
```

```
$ lvremove /dev/mynew_vg/vol02
```

```
linuxconfig.org# lvremove /dev/mynew_vg/vol02
Do you really want to remove active logical volume "vol02"? [y/n]: y
  Logical volume "vol02" successfully removed
linuxconfig.org# lvdisplay
  --- Logical volume ---
  LV Name                /dev/mynew_vg/vol01
  VG Name                mynew_vg
  LV UUID                ZONBw5-1Muk-InQl-7WcR-wPin-WaPd-sDCKNV
  LV Write Access        read/write
  LV Status              available
  # open                 0
  LV Size                1.17 GB
  Current LE             300
  Segments               3
  Allocation             inherit
  Read ahead sectors     0
  Block device           254:0

linuxconfig.org# []
```

More workAround

1. After Creating all the Partition,change the ID of that particular partition from ID 83 to ID 8e which is assign for LVM
2. Don’t format LVM partition

3. As we can see that to access the partition in Linux, we have to go through /dev/had, Similarly in LVM one cannot access the partition directly, you have to go through
4. Pv—Physical Volume
5. Since we have /dev/hda5 is our /home LVM partition so we have to create physical volume of /dev/hda5
6. $ pvdisplay
7. $ pvcreate /dev/hda5
8. $ vgscan
9. $ vgcreate myvol /dev/hda5
vgdisplay ---
10. $ lvcreate -L <+lvsize> -n lv1 myvol
11. $ lvdisplay
12. $ mke2fs -j /dev/myvol/lv1
13. $ mount /dev/myvol/lv1 /home
14. $ df -h

More Workaround

1. Add another HDD which is connecting to motherboard as Primary Slave
Therefore that disk should be readable by Linux as /dev/hdb
2. $ fdisk /dev/hdb
3. Create a single primary partition the size of entire disk and change the ID of that partition to 8e.
4. Create PV of new drive which will be
$ pvcreate /dev/hdb1
5. Add this new PV into a existing VG (MYVOL)
$ vgextend myvol /dev/hdb1
6. Extend your Logical Volume into a existing LV
$ lvextend -L +2000M /dev/myvol/lv1
$resize2fs /dev/myvol/lv1

7. The above command will extend Logical Volume by 2GB, which mean our /home is above 2gb as its mounting on it by

/dev/myvol/lv1 /home ext3 default 1 2

8. $mount -a
9. $ df -h

(a) **Increase Your LVM size up to 6GB**

Note: We have exiting value is 2GB, therefore to increases LVM up to 6GB

```
$ lvextend -L +4000M /dev/myvol/lv1
$ resize2fs /dev/myvol/lv1
```

(b) **Reduce Your LVM size up to 1GB**

Note: we have now LVM up to is 6GB, which have to decrease up to 1GB, therefore

```
$ umount /home
$ e2fsck -yc /dev/myvol/lv1
$ resize2fs /dev/myvol/lv1 1000M
$ lvreduce -L 1000M /dev/myvol/lv1
$ mount -a
$ df -h
```

The Linux Schedulers **cron-cronology-sequence cronological order-date-wise**

Cron job are used to schedule commands to be executed periodically i.e. to setup commands which will repeatedly run at a set time, you

can use the cron jobs.

crontab is the command used to install, deinstall or list the tables used to drive the cron daemon in Vixie Cron. Each user can have their own crontab, and though these are files in /var/spool/cron/crontabs, they are not intended to be edited directly. You need to use crontab command for editing or setting up your own cron jobs.

To edit your crontab file, type the following command:

```
$ crontab -e
```

Syntax of crontab

Your cron job looks like as follows:

1 2 3 4 5 /path/to/command arg1 arg2

Where,

- 1: Minute (0-59)
- 2: Hours (0-23)
- 3: Day (0-31)
- 4: Month (0-12 [12 == December])
- 5: Day of the week(0-7 [7 or 0 == sunday])
- /path/to/command - Script or command name to schedule

Same above five fields structure can be easily remembered with following diagram:

```
* * * * * command to be executed

- - - - -

| | | | |

| | | | ----- Day of week (0 - 7) (Sunday=0 or 7)

| | | ------- Month (1 - 12)

| | --------- Day of month (1 - 31)

| ----------- Hour (0 - 23)

------------- Minute (0 - 59)
```

Example(s)

If you wished to have a script named /root/backup.sh run every day at 3am, my crontab entry would look like as follows:

```
crond*                  -----> Binary or App server daemon

/etc/rc.d/init.d/crond  -----> Initscript to start crond server
```

/etc/crontab -----> System crontab file

mins hrs DOM MOY DOW

00-59 00-23 1-31 1-12 0-7 (0=Sun 1=Mon, 2=Tue, 3=Wed,4=Thu, 5=Fri, 6=Sat and 7=Sun)

Each of the time-related fields may contain:

- A '*', which matches everything, or matches any value
- A single integer, which matches exactly
- Two integers seperated by a dash, matching a range of values ie
8-10 in the hr field would match 8am,9am and 10am.
8-10,13 would match 8am,9am,10am and 1pm
- A comma-seperated series of ints or ranges, matching any listed value
- */2 in the hr field refers to midnote, 2am, 4am and so forth ie the cmd is executed every 2 hrs
- 0-10/2 in the hr field refers to midnite, 2am, 4am, 6am, 8am and 10am

Note:

- A crontab entry is considered to match the current time when the min and hr fields match the curr time and the mth field matches the current month
- An entry is considered to match the current date when the day of month field [3rd] matches the current day of the mth OR the day of week [5th] field matches the current day of the week:

IT IS NOT NECESSARY THAT BOTH THE DAY OF THE MTH AND DAY OF THEWEEK MATCH!

- If both the time and date match the current time and date the cmd is executed!
- Never put a '*' in the first field unless u want the cmd to run every minute
- You MAY hand-edit this file but it is never necessary since run-parts does everything. Simply put a shell script in the appropriate /etc/cron.*/ dirs

Also the crond* daemon need not be restart. It will do just that every minute anyway

Example: Users often forget to shutdown their machines and go home. Hence, machine should auto shutdown at 11 pm

/etc/crontab

Install your cronjob:# crontab -e
00 23 * * * root /sbin/shutdown -h now

b) Append following entry:
0 3 * * * /root/backup.sh

Run five minutes after midnight, every day:
5 0 * * * /path/to/command

Run at 2:15pm on the first of every month:
15 14 1 * * /path/to/command

Run at 10 pm on weekdays:
0 22 * * 1-5 /path/to/command

Run 23 minutes after midnigbt, 2am, 4am ..., everyday:
23 0-23/2 * * * /path/to/command

Run at 5 after 4 every sunday:
5 4 * * sun /path/to/command

If you run many sites, you can use this tip to make managing your cron jobs easier. To minimize the clutter, create a /etc/cron.5min directory and have crontab read this directory every five minutes.

*/5 * * * * root run-parts /etc/cron.5min

45 * * * * /usr/bin/lynx -source http://example.com/cron.php

45 * * * * /usr/bin/wget -O - -q -t 1 http://www.example.com/cron.php

45 * * * * curl --silent --compressed http://example.com/cron.php

00 11,16 * * * /home/sadhiq/bin/incremental-backup

- **00** – 0th Minute (Top of the hour)
- **11,16** – 11 AM and 4 PM
- ***** – Every day
- ***** – Every month

00 09-18 * * * /home/ramesh/bin/check-db-status

- ***** – Every day of the week
- **00** – 0th Minute (Top of the hour)
- **09-18** – 9 am, 10 am,11 am, 12 am, 1 pm, 2 pm, 3 pm, 4 pm, 5 pm, 6 pm
- ***** – Every day
- ***** – Every month

- ***** – Every day of the week

```
*/10 * * * * /home/sadhiq/check-disk-space
```

Cron jobs saved in to /var/spool/cron/$username

$ crontab -l --> To list your crontab jobs

$ crontab -r --> To remove or erase all crontab jobs

Use special string to save time

Instead of the first five fields, you can use any one of eight special strings. It will not just save your time but it will improve readability.

Special string	Meaning
@reboot	Run once, at startup.
@yearly	Run once a year, "0 0 1 1 *".
@annually	(same as @yearly)
@monthly	Run once a month, "0 0 1 * *".
@weekly	Run once a week, "0 0 * * 0".
@daily	Run once a day, "0 0 * * *".
@midnight	(same as @daily)
@hourly	Run once an hour, "0 * * * *".

Run ntpdate every hour:

```
@hourly /path/to/ntpdate
```

Typical /etc/crontab file entries:

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/
$ run-parts
```

```
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

Directory	Description
/etc/cron.d/	Put all scripts here and call them from /etc/crontab file.
/etc/cron.daily/	Run all scripts once a day
/etc/cron.hourly/	Run all scripts once an hour
/etc/cron.monthly/	Run all scripts once a month
/etc/cron.weekly/	Run all scripts once a week

How do I use above directories to put scripts?

Here is a sample shell script (clean.cache) to clean up cached files every 10 days. This script is directly created at /etc/cron.daliy/ directory i.e. create a file called /etc/cron.daily/clean.cache:

```
 #!/bin/bash


CROOT="/tmp/cachelighttpd/"

DAYS=10

LUSER="lighttpd"

LGROUP="lighttpd"


 # start cleaning

/usr/bin/find ${CROOT} -type f -mtime +${DAYS} | xargs -r /bin/rm


# if directory deleted by some other script just get it back

if [ ! -d $CROOT ]
```

```
then
        /bin/mkdir -p $CROOT
        /bin/chown ${LUSER}:${LGROUP} ${CROOT}
fi
```

Cron Access Perms

/etc/cron.allow and /etc/cron.deny

If a user is only in /etc/cron.allow, then all others are denied

If a user is only in /etc/cron.deny then all others are allowed/not affected

If cron.deny is touched, then no users is allowed to create a crontab

If cron.allow is touched, then no users is allowed to create a crontab

AT

'at' executes a command once on a particular day, at a particular time. *at* will add a particular command to be executed.

Examples:

$ at 21:30

You then type the commands you want executed then press the end-of-file character (normally **CTRL-D**). Also try:

$ at now + time

This will run at at the current time + the hours/mins/seconds you specify (use *at now + 1 hour* to have command(s) run in 1 hour from now...)

You can also use the *-f* option to have at execute a particular file (a shell script).

$ at -f shell_script now + 1 hour

This would run the shell script 1 hour from now.

atq Will list jobs currently in queue for the user who executed it, if root executes at it will list all jobs in queue for the at daemon. Doesn't need or take any options.

atrm Will remove a job from the 'at' queue.

Command syntax:

$ atrm job_no

Will delete the job "job_no" (use *atq* to find out the number of the job)

$ at -f myjobs.txt now + 1 hour

$ at -f myjob now + 1 min

$ at 10 am tomorrow

$ at 11:00 next month

$ at 22:00 today

$ at now + 1 week

$ at noon

Anacron

anacron is another tool designed for systems which are not always on, such as home computers.

While *cron* will not run if the computer is off, *anacron* will simply run the command when the computer is next on (it catches up with things).

Quota

Important Note:

1. Quotas can only be created for partitions.

2. Quota is of two types, user and group.

3. If 1 MB quota is set for the partition /home, then every directory under /home or every user on the system, since each directory in /home represents an user, can use a max of 1 MB.

Enabling Quotas

1. Go to /etc/fstab and in the permissions field, enter "usrquota" followed by a "," for the partition where you want to enable quota in our case /home.
Note: If you want to enable group quota then enter "grpquota" instead of "usrquota"
Reboot and directly jump to step 5! else...

2. Unmount and mount /home for the changes to take effect
```
$ umount /home
$ mount /home
  or
$ mount -o remount /home
```

Note:If the system is rebooted after step 2 skip step 3&4 and jump to step 5.

3. To scan /home and enable quota
```
$ quotacheck -vcu /home
```

4. To turn on quota on /home
```
$ quotaon -v /home
```

5. To check if quota is on or not
```
$ repquota -a
```

Implementing Quotas

6. To edit quota for a user
```
$ edquota -u <username>
```

Note: u stands for user, for group type g and give groupname

7. To edit grace period
```
$ edquota -t
```

8. To copy a quota setting of one user to another user
```
$ edquota -p <source_user> <user> OR
```

For all users

```
$ edquota -p <source_user> `awk -F: '$3>499 {print $1}' /etc/passwd`
```

Repairing aquota.user file

9. Boot in single mode

10. Turn off quotas
    $ quotaoff -v /home

Enable Quota on Filesytem -> /home

Cond: if there is no /home partition, imply quota on / filesystem

Practical

$ vi /etc/fstab

    /dev/hda7   /home   ext3   defaults,usrquota   0 0

Remount the /home filesystem with usrquota parameters

$ mount -o remount /home

Confirm whether usrquota is implied

$ mount

It should like this:

/dev/hda7 on /home type ext3 (rw,usrquota)

Create quota database file i.e aquota.user on /home

$ quotacheck -cuv /home --> This creates aquota.user under /home

Enable the quota on /home

$ quotaon /home

Set user level quota on user neo restricting the size below 70k

$ edquota -u neo

This opens up a temp file under /tmp and vi as a editor

Disk quotas for user neo (uid 529):

<---- file size quota ----------> | <------ No. of files quota -->

Filesystem	blocks	soft	hard	inodes	soft	hard
/dev/hda7	11	50	69	11	0	0

Quota Implemented for the user gets updated in /home/aquota.user

Confirm quota really works or not

Login as neo

$ su - neo

    $ dd if=/dev/zero of=/home/neo/data.tmp bs=1k count=70

    This should show the below error

```
-------------------------------------------------------

warning, user block quota exceeded.

dd: writing data.tmp: Disk quota exceeded

-------------------------------------------------------
```

If user neo wants to view his own quota

```
$ quota
```

As a root user you would be interested in viewing the quota statistics on user level basis.

```
# repquota -a
```

How to enable grpquota i.e. Group Quota

```
$ vi /etc/fstab

   /dev/hda7       /home   ext3    defaults,usrquota,grpquota  0 0
```

Remount the /home filesystem with usrquota and grpquota parameters

```
$ mount -o remount /home
```

Confirm whether usrquota is implied

```
$ mount
```

It should look like this:

```
     /dev/hda7 on /home type ext3 (rw,usrquota,grpquota)
```

Create quota database file i.e aquota.group, aquota.user on /home

```
$ quotacheck -cugv /home
```

--> This creates aquota.group, aquota.user under /home

How to set grpquota

```
$ edquota -g ADMINS
```

How to disable quota

```
# quotaoff /home
```

How to imply the quota settings meant for user neo onto user champu

```
# edquota -p neo jane
```

Commands

- quota – display disk usage and limits
- rquota – implement quotas on remote machines
- fstab – static information about the filesystems
- edquota – edit user quotas
- setquota – set disk quotas (Command line editor)
- quotacheck – scan a filesystem for disk usage, create, check and repair quota files
- quotaon – turn filesystem quotas on
- quotaoff – turn filesystem quotas off

Kernel Compilation

if you want to update the kernel from new source code you have downloaded, or you have applied a patch to add new functionality or hardware support, you will need to compile and install a new kernel to actually use that new functionality. Compiling the kernel involves translating the kernel's contents from human-readable code to binary form. Installing the kernel involves putting all the compiled files where they belong in /boot and /lib and making changes to the bootloader.

The process of compiling the kernel is almost completely automated by the make utility as is the process of installing. By providing the necessary arguments and following the steps covered next, you can recompile and install a custom kernel for your use.

Basically, there are three types of kernel:

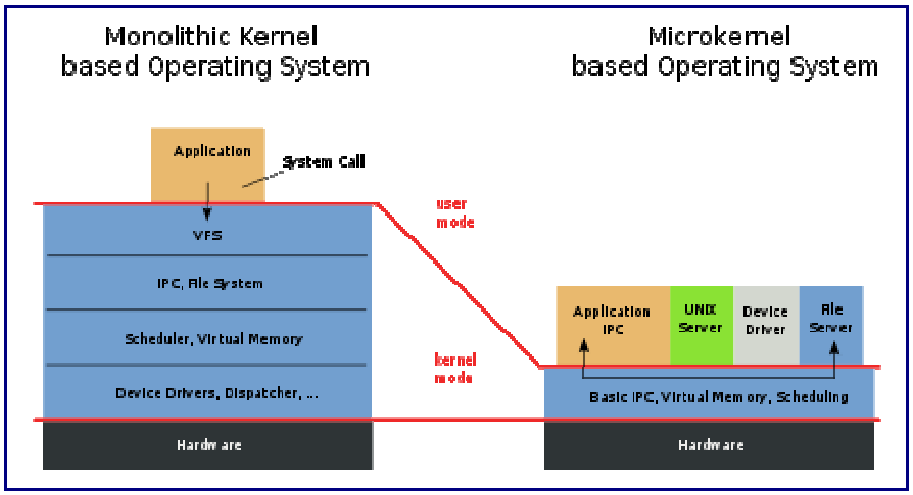
- **Monolithic Kernel – Micro Kernel – ExoKernel**

Monolithic: As the name itself suggests, the kernel has every services like, FS Management, MM, Process Management, etc. in the kernel space. It does not run as a seperate process. So, as you guess, there is no context switching, when you ask for a service. But, the probability of a monolithic kernel getting struck is more. Because, if there is a bug in the kernel itself, nothing can rescue it. Linux and Windows are good examples of Monolithic kernel. Linux, being a monolithic kernel, you can insert modules into the kernel dynamically using insmod command.

Micro Kernel: Micro kernel runs all the services as a daemon in the user space. So, if a problem occurs in any of the service, the kernel will be able to decide what to do next. But, you pay-off the time to switch to a service in this type of kernel. Micro kernels are some what difficult to design and build than the monolithic kernel. There are always a discussion over the internet, talking about the advantage and disadvantages of monolithic and micro

kernel.

Exo Kernel: Exo kernel is not yet stabilized. It's under design and research. The user mode processes running in this type of kernel has the ability to access kernel resources like process tables, etc directly.



Structure of monolithic and microkernel-based operating systems, respectively

Compilation
Steps to compile kernel - Redhat 9

Install dep
kernel-source-2.4.20-8.i386.rpm
binutils-2.13.90.0.18-9.i386.rpm
glibc-kernheaders-2.4-8.10.i386.rpm
cpp-3.2.2-5.i386.rpm
gcc-3.2.2-5.i386.rpm
glibc-2.3.2-11.9.i386.rpm
  libgcc-3.2.2-5.i386.rpm
glibc-common-2.3.2-11.9.i386.rpm
ncurses-5.3-4.i386.rpm
glibc-devel-2.3.2-11.9.i386.rpm
ncurses-devel-5.3-4.i386.rpm

Once these all dependencies are installed:

1) Go to /usr/src/linux-2.4/

2) edit Makefile [Top-Level]
  parameter EXTRAVERSION=-8champu

3) make mrproper [delete the .config]
architecture
4) cp /usr/src/linux-2.4/configs/kernel-2.4.18-i686.config [see uname -m]
/usr/src/linux-2.4/.config
  or simply

  cp -p configs/kernel-2.4.18-i686.config .config

5) make oldconfig - To update the .config with running kernel parameters

6) make config / make menuconfig (for Text) / make xconfig - make necessary changes - enable ntfs - disable sound & bluetooth

7) make dep - checks dependecies & construts MAKEFILE.

8) make clean - cleans unwanted files form memory loaded by above commands.

9) make bzImage - Actual kernel compilation process

10) make modules - Actual KLM compilation process

11) make modules_install #check in /lib/modules/2.4.20-8champu/

12) cp /usr/src/linux-2.4.20-8/arch/i386/boot/bzImage
/boot/vmlinuz-2.4.20-8champu

13) cp /usr/src/linux-2.4.18-14/System.map
/boot/System.map-2.4.20-8champu

14) cp /usr/src/linux-2.4.20-8/.config /boot/config-2.4.20-8champu [OPTIONAL]

15) mkintrd /boot/initrd-2.4.20-8champu.img 2.4.20-8champu

16) vi /etc/grub.conf #Add the new customized kernel entries

```
title REDHAT 9champu (customized)
        root (hd0,8)
        kernel /vmlinuz-2.4.20-8champu  ro root=/dev/hda11 rhgb
quiet
        initrd /initrd-2.4.20-8champu.img
```

note - hd0,8 for boot partition - (/dev/hda9 -1=hda8) (/de/hda11 is “/”)

17) reboot

Centos 5 Steps to compile kernel 2.6 :-

1> copy kernel tarball file ino /usr/src/kernels/ location & untar into that location

```
2>tar -jxvf /usr/src/kernels/ linux-2.6. 18.2.tar. bz2
3>cd /usr/src/kernels/ linux-2.6. 18.2
4>make gconfig (graphical)
make menuconfig (text)
5> make clean
6> make bzImage
7> make modules
8> make modules_install
9>cp arch/i386/boot/ bzImage /boot/vmlinuz- 2.6.18-2
10>cp /usr/src/kernels/ linux-2.6. 18.2/System. map /boot/System.
map-2.6.18- 2
11>ln -s /boot/System. map-2.6.18- 2 /boot/System. map
12> Create initrd :--
$first check into /lib/modules/ 2.6.18.2 --> this is created or not
then execute next command
mkinitrd /boot/initrd- 2.6.18.2. img 2.6.18.2
Final Steps
$vi /etc/grub.conf
default=0
timeout=77
splashimage= (hd0,0)/grub/ splash.xpm. gz
title Red Hat Enterprise Linux AS (2.6.9-34.EL)
root (hd0,0)
kernel /vmlinuz-2.6. 9-34.EL ro root=LABEL=/ rhgb quiet
initrd /initrd-2.6. 9-34.EL.img
title Red Hat Enterprise Linux AS (2.6.18.2)
root (hd0,0)
kernel /vmlinuz-2.6. 18-2 ro root=LABEL=/ rhgb quiet
initrd /initrd-2.6. 18.2.img
```

Kernel Definition
http://www.linfo.org/kernel.html

Kernel compilation
http://www.cyberciti.biz/tips/compiling-linux-kernel-26.html
http://book.opensourceproject.org.cn/distrib/ubuntu/unleashed/opensource/0672329093/ch35lev1sec7.html
http://wiki.centos.org/HowTos/Custom_Kernel

This is one the essential and important task. Many time we upgrade our kernel and some precompiled drivers won't work with Linux. Especially if you have weird hardware; then vendor may send you driver code aka C files to compile. Or even you can write your own Linux kernel driver. Compiling kernel driver is easy. Kernel 2.6.xx makes it even much more easier. Following steps are required to compile driver as module:

1) You need running kernel source code; if you don't have a source code download it from kernel.org. Untar kernel source code (tar ball) in /usr/src using tar command:
```
$ tar -zxvf kernel* -C /usr/src
```

To be frank kernel headers are more than sufficient to compile kernel modules / drivers. See how to install kernel headers under Debian / Ubuntu Linux or RHEL / CentOS / Fedora Linux.

Next go to your kernel module source code directory and simply create the Makefile file as follows (assuming your kernel module name is foo):

```
$ vi Makefile
```

3) Add following text to it:

```
obj-m = foo.o
KVERSION = $(shell uname -r)
all:
	make -C /lib/modules/$(KVERSION)/build M=$(PWD) modules
clean:
	make -C /lib/modules/$(KVERSION)/build M=$(PWD) clean
```

4) Compile module using make command (module build can be done by any user) :

```
$ make
```

It will finally creates the foo.ko module in current directory. You can see all actual compile command stored in .foo* files in same directory.

5) Once module compiled successfully, load it using insmod or modprobe command. You need to be root user or privileged user to run insmod:

```
# insmod foo.ko
```

Example: hello.c module

1) hello.c C source code. Copy following code and save to hello.c

```
$ mkdir demo; cd demo
$ vi hello.c
```

2)Add following c source code to it:

```
#include <linux/module.h>       /* Needed by all modules */
#include <linux/kernel.h>       /* Needed for KERN_INFO */
#include <linux/init.h>         /* Needed for the macros */

static int __init hello_start(void)
{
printk(KERN_INFO "Loading hello module...\n");
printk(KERN_INFO "Hello world\n");
return 0;
}

static void __exit hello_end(void)
{
printk(KERN_INFO "Goodbye Mr.\n");
}

module_init(hello_start);
module_exit(hello_end);
```

This is an example modified from original source for demonstration purpose.

3) Save the file. Create new Makefile as follows:

```
$ vi Makefile
```

Append following make commands:

```
obj-m = hello.o
KVERSION = $(shell uname -r)
all:
        make -C /lib/modules/$(KVERSION)/build M=$(PWD) modules
clean:
        make -C /lib/modules/$(KVERSION)/build M=$(PWD) clean
```

4) Save and close the file.

5) Compile hello.c module:

```
$ make
```

6) Become a root user (use su or sudo) and load the module:

```
$ su -
$ insmod hello.ko
```

Note you can see message on screen if you are logged in as root under run level 3.

7) Verify that module loaded:

```
$ lsmod | less
```

8) See message in /var/log/message file:

```
$ tail -f /var/log/message
```

9) Unload the module:

```
$ rmmod hello
```

10) Load module when Linux system comes up. File /etc/modules use to load kernel boot time. This file should contain the names of kernel modules that are to be loaded at boot time, one per line. First copy your module to /lib/modules/$(uname -r)/kernel/drivers. Following are suggested steps:

(a) Create directory for hello module:

```
$ mkdir -p /lib/modules/$(uname -r)/kernel/drivers/hello
```

(b) Copy module:

```
$ cp hello.ko /lib/modules/$(uname -r)/kernel/drivers/hello/
```

(c) Edit /etc/modules file under Debian Linux:

```
$ vi /etc/modules
```

(d) Add following line to it:

```
hello
```

(e) Reboot to see changes. Use lsmod or dmesg command to verify module loaded or not.

```
$ cat /proc/modules
OR
$ lsmod | less
```

Most people have a fairly recent kernel. But since the kernel is constantly being updated, people on modems (such as myself) don't like downloading the whole source everytime a new version of the kernel comes out... It is a pain to download 14+ megs of stuff when

95% of it is the same stuff that you already have in your kernel source diectory.

For this reason, kernel patches are released. Kernel patches contain only the files that have changed since the last kernel, hence making it less of a pain to upgrade.

It is a good idea to back up your old kernel tree before you do anything to it, just in case something messes up. To do this, do the following:

Become root and then go into your kernel source directory (for me it was /usr/src/linux-2.2.10) and do a 'make clean' to clean it up so you don't compress a lot of crap you don't need as follows

```
# cd /usr/src/linux-2.2.10
# make clean
```

Now you need to go to backup the tree, I did this by doing the following:

```
# cd /usr/src/
# tar zcvf linux-2.2.10-tree.tar.gz linux-2.2.10
```

Now with that backed up, you can go ahead and change the stuff with less worrying...

If you have kernel 2.2.10, like I did, and 2.2.12 is the current stable release (or at least it is as I am writing this) you need all of the patch files after 2.2.10. So in my case, I needed to get patch-2.2.11.gz and patch-2.2.12.gz

http://www.kernelnotes.org is where I got mine from, but I'm sure there are mirrors where you can get the patches from, more on this is on www.kernelnotes.org.

Note: When I downloaded this file using netscape, it un-gzipped it for me as it was downloading... so I didn't have to do the following step that you would have to do if you were using a program such as 'ftp'

un-gzip the file by doing the following:

```
# gzip -d patch-2.2.11.gz
# gzip -d patch-2.2.12.gz
```

This will leave you with patch-2.2.11 and patch-2.2.12 (unless you downloaded the file with netscape, and this step would already have been done for you)

Now move the files to your kernel source directory (using the mv command,

mv patch-2.2.* /usr/src/linux-2.2.10

Now change into your kernel source directory (/usr/src/linux-2.2.10 in my case)

Now you need to apply the patch the the source... Order is important here. Start with the lowest and go to the highest, like the following:

```
# patch -p1 < patch-2.2.11
# patch -p1 < patch-2.2.12
```

Both of these commands will give you lots of output telling you what files are being patched, etc.

After I applied the patches, I went ahead and renamed my source directory to reflect the patches applied (mv /usr/src/linux-2.2.10 /usr/src/linux-2.2.12) and then I removed the old /usr/src/linux link and replaced it with the new location (rm /usr/src/linux and then ln -s /usr/src/linux-2.2.12 /usr/src/linux)

Now just compile your kernel

Kernel Patch with .KO
http://wiki.centos.org/HowTos/BuildingKernelModules
http://www.cyberciti.biz/tips/compiling-linux-kernel-module.html

Patch
http://www.cyberciti.biz/tips/how-to-patch-running-linux-kernel.html

Patch O Matic
http://www.fifi.org/doc/iptables-dev/html/netfilter-extensions-HOWTO-2.html

Patch without reboots - Ksplice
http://www.cyberciti.biz/tips/debian-centos-redhat-hotfix-patch-linux-kernel.html

Faq
http://kernelnewbies.org/FAQ

Kernel Tuning

Kernel tuning with sysctl
The Linux kernel is flexible, and you can even modify the way it

works on the fly by dynamically changing some of its parameters, thanks to the sysctl command. Sysctl provides an interface that allows you to examine and change several hundred kernel parameters in Linux or BSD. Changes take effect immediately, and there's even a way to make them persist after a reboot. By using sysctl judiciously, you can optimize your box without having to recompile your kernel, and get the results immediately.

To start getting a taste of what sysctl can modify, run sysctl -a and you will see all the possible parameters. The list can be quite long: in my current box there are 712 possible settings.

```
$ sysctl -a
kernel.panic = 0
kernel.core_uses_pid = 0
kernel.core_pattern = core
kernel.tainted = 129

...many lines snipped...
```

If you want to get the value of just a single variable, use something like sysctl vm.swappiness, or just sysctl vm to list all variables that start with "vm." Add the -n option to output just the variable values, without the names; -N has the opposite effect, and produces the names but not the values.

You can change any variable by using the -w option with the syntax sysctl -w *variable=value*. For example, sysctl -w net.ipv6.conf.all.forwarding=1 sets the corresponding variable to true (0 equals "no" or "false"; 1 means "yes" or "true") thus allowing IP6 forwarding. You may not even need the -w option -- it seems to be deprecated. Do some experimenting on your own to confirm that.

sysctl values are loaded at boot time from the /etc/sysctl.conf file. This file can have blank lines, comments (lines starting either with a "#" character or a semicolon), and lines in the "*variable=value*" format. For example, my own sysctl.conf file is listed below. If you want to apply it at any time, you can do so with the command sysctl -p.

```
# Disable response to broadcasts.
net.ipv4.icmp_echo_ignore_broadcasts = 1

# enable route verification on all interfaces
net.ipv4.conf.all.rp_filter = 1

# enable ipV6 forwarding
net.ipv6.conf.all.forwarding = 1

# increase the number of possible inotify(7) watches
fs.inotify.max_user_watches = 65536
```

sysctl and the /proc directory

The /proc/sys virtual directory also provides an interface to the sysctl parameters, allowing you to examine and change them. For example, the /proc/sys/vm/swappiness file is equivalent to the vm.swappiness parameter in sysctl.conf; just forget the initial "/proc/sys/" part, substitute dots for the slashes, and you get the

corresponding sysctl parameter. (By the way, the substitution is not actually required; slashes are also accepted, though it seems everybody goes for the notation with the dots instead.) Thus, echo 10 >/proc/sys/vm/swappiness is exactly the same as sysctl -w vm.swappiness=10. But as a rule of thumb, if a /proc/sys file is read-only, you cannot set it with sysctl either.

linux network optimize with sysctl

Disabling the TCP options reduces the overhead of each TCP packet and might help to get the last few percent of performance out of the server. Be aware that disabling these options most likely decreases performance for high-latency and lossy links.
* net.ipv4.tcp_sack = 0
* net.ipv4.tcp_timestamps = 0

Increasing the TCP send and receive buffers will increase the performance a lot if (and only if) you have a lot of large files to send.

* net.ipv4.tcp_wmem = 4096 65536 524288
* net.core.wmem_max = 1048576

If you have a lot of large file uploads, increasing the receive buffers will help.

* net.ipv4.tcp_rmem = 4096 87380 524288
* net.core.rmem_max = 1048576

```
# These ensure that TIME_WAIT ports either get reused or closed fast.
net.ipv4.tcp_fin_timeout = 1
net.ipv4.tcp_tw_recycle = 1
# TCP memory
net.core.rmem_max = 16777216
net.core.rmem_default = 16777216
net.core.netdev_max_backlog = 262144
net.core.somaxconn = 262144

net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_max_orphans = 262144
net.ipv4.tcp_max_syn_backlog = 262144
net.ipv4.tcp_synack_retries = 2
net.ipv4.tcp_syn_retries = 2

# you shouldn't be using conntrack on a heavily loaded server anyway, but these are
# suitably high for our uses, insuring that if conntrack gets turned on, the box doesn't die
net.ipv4.ip_conntrack_max = 1048576
net.nf_conntrack_max = 1048576

# increase Linux TCP buffer limits
echo 8388608 > /proc/sys/net/core/rmem_max
echo 8388608 > /proc/sys/net/core/wmem_max

# increase Linux autotuning TCP buffer limits
echo "4096 87380 8388608" > /proc/sys/net/ipv4/tcp_rmem
echo "4096 65536 8388608" > /proc/sys/net/ipv4/tcp_wmem

#echo 65536 > /proc/sys/fs/file-max # physical RAM * 256/4
```

```
echo "1024 65000" > /proc/sys/net/ipv4/ip_local_port_range
#echo 1 > /proc/sys/net/ipv4/tcp_syncookies
echo 8192 > /proc/sys/net/ipv4/tcp_max_syn_backlog
# Decrease the time default value for tcp_fin_timeout connection
#echo 30 > /proc/sys/net/ipv4/tcp_fin_timeout
#echo 3 > /proc/sys/net/ipv4/tcp_syn_retries
#echo 2 > /proc/sys/net/ipv4/tcp_retries1
# Decrease the time default value for tcp_keepalive_time connection
#echo 1800 >/proc/sys/net/ipv4/tcp_keepalive_time
# Turn off tcp_window_scaling
echo 0 >/proc/sys/net/ipv4/tcp_window_scaling
#echo "67108864" > /proc/sys/kernel/shmmax
# Turn off the tcp_sack
echo 0 >/proc/sys/net/ipv4/tcp_sack # This disables RFC2018 TCP
Selective Acknowledgements
#Turn off tcp_timestamps
echo 0 >/proc/sys/net/ipv4/tcp_timestamps # This disables RFC1323
TCP timestamps
echo 5 > /proc/sys/kernel/panic # reboot 5 minutes later then
kernel panic

the third:
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_syncookies = 1
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
```

Refrence

http://shebangme.blogspot.com/2010/07/kernel-sysctl-configuration-file-for.html
swappiness
http://www.linux.com/archive/feature/146599

Device driver
http://en.wikipedia.org/wiki/Device_driver
Lsmod
http://en.wikipedia.org/wiki/Lsmod
Modprobe
http://en.wikipedia.org/wiki/Modprobe
Oracle + sysctl
http://www.puschitz.com/TuningLinuxForOracle.shtml
http://www.puschitz.com/TuningLinuxForOracle.shtml#SettingSHMMAXParameter
http://www.puschitz.com/TuningLinuxForOracle.shtml#TheSEMMSLParameter

Refrence
http://www.linux.com/archive/feature/126718
http://www.fcicq.net/wp/?p=197
http://www.cyberciti.biz/tips/linux-procfs-file-descriptors.html
http://en.opensuse.org/Kernel_module_configuration

http://www.cyberciti.biz/tips/blade-server-disable-floppy-driver-module.html

Blacklist
Just open your /etc/modprobe.conf file and turn of auto loading using
following syntax:
alias driver-name off
If you are using Debian / Ubuntu Linux...

open /etc/modprobe.d/blacklist file and add drivername using following syntax:
blacklist driver-name

Linux Kernel Magic SysRq keys

Kernel offers you something that allows you to recover your system from a crash or at the least lets you to perform a proper shutdown using the Magic SysRq Keys. The magic SysRq key is a select key combination in the Linux kernel which allows the user to perform various low level commands regardless of the system's state using the SysRq key. It is often used to recover from freezes, or to reboot a computer without corrupting the filesystem.

How do I use the magic SysRq keys in emergency?

You need to use following key combination in order to reboot/halt/sync file system etc:
ALT+SysRq+COMMAND-KEY

The 'SysRq' key is also known as the 'Print Screen' key. COMMAND-KEY can be any one of the following (all keys need to hit simultaneously) :

- **'b'** : Will immediately reboot the system without syncing or unmounting your disks.
- **'o'** : Will shutdown your system off (if configured and supported).
- **'s'**: Will attempt to sync all mounted filesystems.
- **'u'** : Will attempt to remount all mounted filesystems read-only.
- **'e'** : Send a SIGTERM to all processes, except for init.
- **'h'**: Show help, indeed this the one you need to remember.

So whey you need to tell your Linux computer to reboot or when your X server is crashed or you don't see anything going across the screen then just press:

ALT+SysRQ+s : (Press and hold down ALT, then SysRQ (Print Screen) key and press 's') -Will try to syn all mounted system

ALT+SysRQ+r : (Press and hold down ALT, then SysRQ (Print Screen)

key and press 'r') -Will reboot the system.

If you wish to shutdown the system instead of reboot then press following key combination:
ALT+SysRQ+o

ipt_sysrq is a new iptables target that allows you to do the same as the magic sysrq key on a keyboard does, but over the network. Sometimes a remote server hangs and only responds to icmp echo request (ping). Every administrator of such machine is very unhappy because (s)he must go there and press the reset button. It takes a long time and it's inconvenient. So use the Network Magic SysRq and you will be able to do more than just pressing a reset button. You can remotely sync disks, remount them read-only, then do a reboot. And everything comfortably and only in a few seconds. Please see Marek Zelem page to enableIP Tables network magic SysRq function.

The magic Sysrq key basically has a key combination of **<ALT> + <SysRq or Prnt Scrn> + <Command key>**.

The command key can be one of the following providing a specific functionality

'b' *- Will immediately reboot the system without syncing or unmounting your disks.*

'c' *- Will perform a kexec reboot in order to take a crashdump.*

'd' *- Shows all locks that are held.*

'e' *- Send a SIGTERM to all processes, except for init.*

'f' *- Will call oom_kill to kill a memory hog process.*

'g' *- Used by kgdb on ppc and sh platforms.*

'h' *- Will display help (actually any other key than those listed here will display help. but 'h' is easy to remember*

'i' *- Send a SIGKILL to all processes, except for init.*

'k' *- Secure Access Key (SAK) Kills all programs on the current virtual console. NOTE: See important comments below in SAK section.*

'm' *- Will dump current memory info to your console.*

'n' *- Used to make RT tasks nice-able*

'o' *- Will shut your system off (if configured and supported).*

'p' *- Will dump the current registers and flags to your console.*

'q' *- Will dump a list of all running timers.*

'r' *- Turns off keyboard raw mode and sets it to XLATE.*

's' *- Will attempt to sync all mounted filesystems.*

't' *- Will dump a list of current tasks and their information to your console.*

'u' *- Will attempt to remount all mounted filesystems read-only.*

'v' *- Dumps Voyager SMP processor info to your console.*

'w' *- Dumps tasks that are in uninterruptable (blocked) state.*

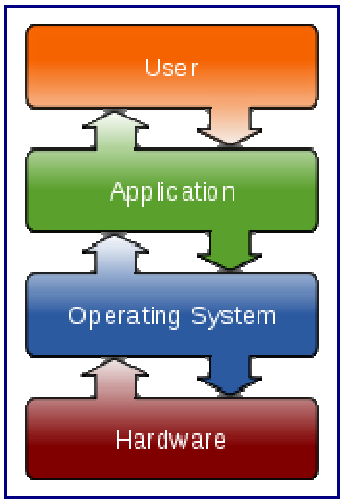
'x' *- Used by xmon interface on ppc/powerpc platforms.*

'0•–'9• *- Sets the console log level, controlling which kernel messages will be printed to your console. ('0•, for example would make it so that only emergency messages like PANICs or OOPSes would make it to your console.)*

Ref -
http://www.susegeek.com/general/linux-kernel-magic-sysrq-keys-in-opensuse-for-crash-recovery/
http://www.cyberciti.biz/tips/reboot-linux-box-after-a-kernel-panic.html
http://www.cyberciti.biz/tips/reboot-or-halt-linux-system-in-emergency.html

In computing, a **device driver** or **software driver** is a computer program allowing higher-level computer programs to interact with a hardware device.

A driver typically communicates with the device through the computer bus or communications subsystem to which the hardware connects. When a calling program invokes a routine in the driver, the driver issues commands to the device. Once the device sends data back to the driver, the driver may invoke routines in the original calling program. Drivers are hardware-dependent and operating-system-specific. They usually provide the interrupt handling required for any necessary asynchronous time-dependent hardware interface.

Operating systems 

The mknod command

MAKEDEV is the preferred way of creating device files which are not present. However sometimes the **MAKEDEV** script will not know about the device file you wish to create. This is where the **mknod** command comes in. In order to use **mknod** you need to know the major and minor node numbers for the device you wish to create. The devices.txt file in the kernel source documentation is the canonical source of this information.

To take an example, let us suppose that our version of the **MAKEDEV** script does not know how to create the /dev/ttyS0 device file. We need to use **mknod** to create it. We know from looking at the devices.txt file that it should be a character device with major number 4 and minor number 64. So we now know all we need to create the file.

```
# mknod /dev/ttyS0 c 4 64
# chown root.dialout /dev/ttyS0
# chmod 0644 /dev/ttyS0
# ls -l /dev/ttyS0
crw-rw----   1 root dialout    4,   64 Oct 23 18:23 /dev/ttyS0
```

As you can see, many more steps are required to create the file. In this example you can see the process required however. It is unlikely in the extreme that the ttyS0 file would not be provided by the **MAKEDEV** script, but it suffices to illustrate the point.

$mknod /opt/champu b 3 10

$mount /opt/champu /home

1. **lsmod**
2. **insmod**

3. **rmmod**
4. **modprobe**
5. **modinfo**
6. **depmod**

lsmod is a command on Linux systems which prints the contents of the /proc/modules file. It shows which loadable kernel modules are currently loaded.

Abridged example output:

```
# lsmod
Module                  Size  Used by
af_packet              27392  2
8139too                30592  0
snd_cs46xx             96872  3
snd_pcm_oss            55808  1
snd_mixer_oss          21760  2 snd_pcm_oss
ip6table_filter         7424  1
ip6_tables             19728  1 ip6table_filter
ipv6                  290404  22
xfs                   568384  4
sis900                 18052  5
libata                169920  1 pata_sis
scsi_mod              158316  3 usb_storage,sd_mod,libata
usbcore               155312  6 ohci_hcd, usb_storage, usbhid
```

lsmod

First column is Module name and second column is size of modules i..e
the output format is module name, size, use count, list of referring
modules.

modprobe is a Linux program originally written by Rusty Russell used to add a loadable kernel module (LKM) to the Linux kernel or remove an LKM from the kernel. It is commonly used indirectly as udev relies upon modprobe to load drivers for automatically detected hardware.

Networking

Tools

$ifconfig

$neat-tui

$/etc/sysconfig/network-scripts/ifcfg-eth0

$netconfig

$ethtool

$ip r l

$telnet

$nmap

$netstat

$ping

$route

$traceroute

$tcpdump - n/w traffic tool

$iptraf   - Monitor n/w traffic.curses-based tool - Self-explanatory

$etheral -   Network Analyzers which does data capture and filtering

$tethral - Captures and displays only the high level protocols

$ ifconfig --> Status of all interfaces

```
eth0      Link encap:Ethernet  HWaddr 00:50:FC:2A:2C:48
          inet addr:192.0.34.7  Bcast:192.0.34.255
Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 b)  TX bytes:240 (240.0 b)
          Interrupt:11 Base address:0xf000
eth1      Link encap:Ethernet  HWaddr 00:60:CC:AA:2C:9C
          inet addr:192.168.0.20  Bcast:192.168.0.255
Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 b)  TX bytes:240 (240.0 b)
          Interrupt:11 Base address:0xc000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
```

```
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1407 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1407 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:149180 (145.6 Kb)  TX bytes:149180 (145.6 Kb)


$ ifconfig eth0         --> Status of eth0 interface


eth0      Link encap:Ethernet  HWaddr 00:50:FC:2A:2C:48
          inet addr:192.0.34.7  Bcast:192.0.34.255
Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:0 (0.0 b)  TX bytes:240 (240.0 b)
          Interrupt:11 Base address:0xf000
$ ifconfig eth0 IP      --> Set eth0 to IP
$ ifconfig eth0 IP:x    --> Set eth0 to multiplexed IP
$ ifconfig eth0 down    --> Bring eth0 down
$ ifdown eth0           --> ditto
$ ifconfig eth0 up      --> Bring eth0 up
$ ifup eth0             --> ditto
$ ifconfig eth0 -arp    --> Disable use of arp protocol on this
interface


$ ifconfig eth0 -allmulti
```

Enable or disable all-multicast mode. If selected, all multi-cast packets on the network will be received by the interface.

```
$ ifconfig eth0 -promisc
```

Turn off promiscuous mode of the interface eth0. If on, tells the interface to send all traffic on the NW to the kernel, not just traffic addressed to the m/c Check with ifconfig or netstat -i

```
$ ifconfig eth0 hw ether CC:CC:CC:CC:CC:CC
```

Changes the MAC address. Do a 'ifconfig eth0 down' first,change, then 'ifconfig eth0 up'. MAC addr is changed.

```
$ ifconfig eth0 172.16.1.77 broadcast 172.16.1.255 netmask
255.255.0.0
```

Changes IP/BC/netmask all in one go!

```
$ ifconfig eth0 mtu 800
```

Change mtu to 800

ethtool - Display or change ethernet card settings

```
$ ethtool ethX

$ ethtool -h

$ ethtool -a ethX

$ ethtool -A ethX [autoneg on|off] [rx on|off] [tx on|off]

$ ethtool -c ethX

$ ethtool -C ethX [adaptive-rx on|off] [adaptive-tx on|off] [rx-
usecs N]       [rx-frames N] [rx-usecs-irq N] [rx-frames-irq N]
[tx-usecs N] [tx-frames N] [tx-usecs-irq N] [tx-frames-irq N]
[stats-block-usecs N] [pkt-rate-low N] [rx-usecs-low N] [rx-frames-
low N] [tx-usecs-low N] [tx-frames-low N] [pkt-rate-high N]
[rx-usecs-high N] [rx-frames-high N] [tx-usecs-high N]
[tx-frames-high N] [sample-interval N]

$ ethtool -g ethX

$ ethtool -G ethX [rx N] [rx-mini N] [rx-jumbo N] [tx N]

$ ethtool -i ethX

$ ethtool -d ethX

$ ethtool -e ethX

$ ethtool -k ethX

$ ethtool -K ethX [rx on|off] [tx on|off] [sg on|off]

$ ethtool -p ethX [N]

$ ethtool -r ethX
```

$ ethtool -S ethX

$ ethtool -t ethX [offline|online]

$ man ethtool

ping - TCP/IP Diagnostic Tool

Send ICMP ECHO_REQUEST to network hosts

There are two types of ping -

The std Unix ping which sends a ICMP ECHO REQUEST and receives a ICMP ECHO REPLY frim the remote host if it is UP and running

The other is to send a UDP or TCP pkt to port 7 [echo] of the remote host and see that whatever you type is echoed back. The host is UP.

$ telnet remote-host echo or 7

and whatever you type will be echoed back to you. system is alive !

$ ping -c -a -n IP/Hostname [Count/AudiblePing/No Name Resolution]

ping send a packet of 64 bytes by def. The size if 56 ICMP data bytes + 8 bytes for the header data.

$ ping -s 1600 203.12.10.20

Send a larger pkt size than the MTU of Ethernet [1500], you can force fragmentation. You can then identify low-level media issue or a congested NW. Since ping works at the IP layer, no server process [HTTP/DNS] is reqd to be running on the target host. Just a running kernel.

Check the ICMP seq no to see that no pkts are dropped and are in sequence.

Run

$ traceroute --> to get the path the pkt is taking and then track down the

offending mid-way routers by pinging each in succession.

```
$ route ['add'/'del'] [-net|-host] 'addr' {gw 'IP'} {netmask
'mask'}

        'interface'
```

Default route:

/etc/sysconfig/network

```
GATEWAY=IP
or
route add default gw gateway-IP-addr
```

Routing determines path a pkt takes from its source thru a maze of NWs to dest.

Like asking for directions in an unfamiliar place. A person may point you tothe right city, another to a street, another to the right bldg.

Routing is done at the IP layer.

When a pkt bound for some other host arrives, the path is found by matching the dest IP addr against the Kernel Routing Table [KRT]. If it matches a route in the KRT, the pkt is fwd'ed to the 'next-hop gateway' IP addr associated with the route.

Two special cases are possible here:

Case I: pkt may be destined for some host on a directly connected NW. In this case the 'next-hop gateway' IP addr in the KRT will be one of the localhosts own interfaces and the pkt is sent directly to its dest. The type of route is what you normally do with the ifconfig cmd when you configure and interface.

Case II: No route in the KRT matches the dest addr that the pkt wishes to reach. The default route [Gateway] is invoked. Or an error. Most NWs have only one way out and that is the default route. On the Internet backbone, the routers do not have default routes. The buck stops here. If they do not have a routing entry for a dest, the dest cannot be reached and a "network unreachable" ICMP error is sent to the sender

The KRT contains info like "To get to NW X from m/c Y, send pkt to m/c Z with a cost of 1 [metric], alongwith TTL and reliability values for that route.

Routing Policy:

- Static routes : For small unconnected NWs
- Dynamic routes : Many subnets, large NWs, connected to the Internet
- Static/Dyn :

```
$ route
```

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.0.34.0	0.0.0.0.	255.255.255.0	U	0	0	0	eth0
192.168.0.0	0.0.0.0.	255.255.255.0	U	0	0	0	eth1
127.0.0.1	0.0.0.0	255.255.255.0	U	0	0	0	lo
0.0.0.0.	192.0.34.1	0.0.0.0	UG	0	0	0	eth0

$ route -n

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
1. 132.236.227.0	132.236.227.93	255.255.255.0	U	0	0	0	eth0
2. 132.236.212.0	132.236.212.1	255.255.255.192	U	0	0	0	eth1
3. 127.0.0.1	0.0.0.0	255.255.255.0	U	0	0	0	lo
4. default	132.236.227.1	0.0.0.0	UG	0	0	0	eth0
5. 132.236.220.64	132.236.212.6	255.255.255.192	UG	0	0	0	eth1

Routes 1 and 2 were added by ifconfig when the eth0 and eth1 interfaces were configured at bootup

This means to reach machine 132.236.227.93 on the NW 132.236.227.0 the GW is machine 132.236.227.93 – the machine itself is its GW which implies it can be reached directly on this NW and one has to go to no other m/c to consult.

Ditto for the next one.

Route 3 is the loopback interface, a pseudo-device that prevents pkts sent from the host to itself from going out on the NW; instead, they are transferred directly route add default gw 132.236.227.1 eth0

Route 4 is the default route.

It says :

Pkts not explicitly addressed to any of the 3 NWs listed [or to the m/c itself] will be sent to the default GW host, 132.236.227.1

Route 5 says :

To reach NW 132.236.220.64/26, pkts must be sent GW host 132.236.212.6 thru eth1.

netstat – Monitoring your TCP/IP NW

Print network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.

$ netstat –a :

Displays status of all active connections, including Inactive [listening] servers waiting for connects

$ netstat –l :

Show only inactive or listening connections, not establised

$ netstat –p :

Show the PID and name of the program to which each socket belongs

$ netstat –o :

Include information related to networking timers

$ netstat –r :

Show the kernel routing table

$ netstat –vatnp | grep <servicename>

$ netstat –tulnp | grep <servicename>

State : TCP/IP connection [socket] state

ESTABLISHED

The socket has an established connection.

SYN_SENT

The socket is actively attempting to establish a connection to the remote host

Debug Note :

If you find a connection that stays in this state, then a local process is trying very hard to contact a non-existent or inaccessible NW server.

SYN_RECV

A connection request has been received from a remote host and is being initialized

FIN_WAIT1

The socket is closed, and the connection is shutting down.

FIN_WAIT2

Connection is closed, and the socket is waiting for a shutdown from the remote end.

TIME_WAIT

The socket is waiting after close to handle packets still in thenetwork.

CLOSED The socket is not being used.

CLOSE_WAIT

The remote host end has shut down its connection, and the localhost is waiting for the socket to close.

LAST_ACK

The remote end has shut down, and the socket is closed. Waiting for acknowledgement.

LISTEN The socket is listening for incoming connections.Specify -l option to see this.

CLOSING

Both sockets are shut down but we still donâ€™t have all our data sent.

UNKNOWN

The state of the socket is unknown.

USER The login ID of the user who owns the socket

FTP

Active FTP

Passive FTP

Users

Regular FTP

Anonymous FTP

Vsftpd.conf

```
anon_root=/data/directory

# Allow anonymous FTP?

anonymous_enable=YES

# The directory which vsftpd will try to change into after an
anonymous login. (Default = /var/ftp)

anon_root=/data/directory

# Uncomment this to allow local users to log in.

local_enable=YES

# Uncomment this to enable any form of FTP write command.

# (Needed even if you want local users to be able to upload files)

write_enable=YES

# Uncomment to allow the anonymous FTP user to upload files. This
only

# has an effect if global write enable is activated. Also, you will

# obviously need to create a directory writable by the FTP user.

# anon_upload_enable=YES

# Uncomment this if you want the anonymous FTP user to be able to
create

# new directories.

# anon_mkdir_write_enable=YES

# Activate logging of uploads/downloads.

xferlog_enable=YES

# You may override where the log file goes if you like.

# The default is shown below.

xferlog_file=/var/log/vsftpd.log
```

Other vsftpd.conf Options

There are many other options you can add to this file:

- Limiting the maximum number of client connections (max_clients)
- Limiting the number of connections by source IP address (max_per_ip)
- The maximum rate of data transfer per anonymous login. (anon_max_rate)
- The maximum rate of data transfer per non-anonymous login. (local_max_rate)

Descriptions on this and more can be found in the vsftpd.conf man pages.

Anonymous upload

```
mkdir /var/ftp/pub/upload

chmod 722 /var/ftp/pub/upload

ftpd_banner= New Banner Here

write_enable = NO
```

Check files under the following

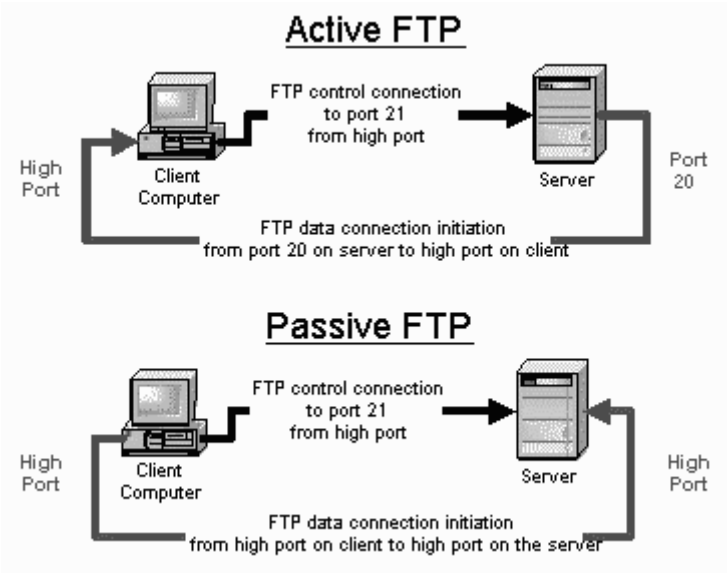
```
$ cd /etc/vsftpd/

$ ls
```

ftpusers users_list vsftpd.conf vsftpd.conf_migrate.sh

Types of FTP

From a networking perspective, the two main types of FTP are active and passive. In active FTP, the FTP server initiates a data transfer connection back to the client. For passive FTP, the connection is initiated from the FTP client. These are illustrated in Figure 15-1.

Figure 15-1 Active and Passive FTP Illustrated



From a user management perspective there are also two types of FTP: regular FTP in which files are transferred using the username and password of a regular user FTP server, and anonymous FTP in which general access is provided to the FTP server using a well known universal login method.

Take a closer look at each type.

Active FTP

The sequence of events for active FTP is:

1. Your client connects to the FTP server by establishing an FTP control connection to port 21 of the server. Your commands such as 'ls' and 'get' are sent over this connection.

2. Whenever the client requests data over the control connection, the server initiates data transfer connections back to the client. The source port of these data transfer connections is always port 20 on the server, and the destination port is a high port (greater than 1024) on the client.

3. Thus the ls listing that you asked for comes back over the port 20 to high port connection, not the port 21 control connection.

FTP active mode therefore transfers data in a counter intuitive way to the TCP standard, as it selects port 20 as it's source port (not a random high port that's greater than 1024) and connects back to the client on a random high port that has been pre-negotiated on the port 21 control connection.

Active FTP may fail in cases where the client is protected from the Internet via many to one NAT (masquerading). This is because the firewall will not know which of the many servers behind it should receive the return connection.

Passive FTP

Passive FTP works differently:

1. Your client connects to the FTP server by establishing an FTP control connection to port 21 of the server. Your commands such as ls and get are sent over that connection.

2. Whenever the client requests data over the control connection, the client initiates the data transfer connections to the server. The source port of these data transfer connections is always a high port on the client with a destination port of a high port on the server.

Passive FTP should be viewed as the server never making an active attempt to connect to the client for FTP data transfers. Because client always initiates the required connections, passive FTP works better for clients protected by a firewall.

As Windows defaults to active FTP, and Linux defaults to passive, you'll probably have to accommodate both forms when deciding upon a security policy for your FTP server.

Regular FTP

By default, the VSFTPD package allows regular Linux users to copy files to and from their home directories with an FTP client using their Linux usernames and passwords as their login credentials.

VSFTPD also has the option of allowing this type of access to only a group of Linux users, enabling you to restrict the addition of new files to your system to authorized personnel.

The disadvantage of regular FTP is that it isn't suitable for general download distribution of software as everyone either has to get a unique Linux user account or has to use a shared username and password. Anonymous FTP allows you to avoid this difficulty.

Anonymous FTP

Anonymous FTP is the choice of Web sites that need to exchange files with numerous unknown remote users. Common uses include downloading software updates and MP3s and uploading diagnostic information for a technical support engineers' attention. Unlike regular FTP where you login with a preconfigured Linux username and password, anonymous FTP requires only a username of anonymous and your email address for the password. Once logged in to a VSFTPD server, you automatically have access to only the default anonymous FTP directory (/var/ftp in the case of VSFTPD) and all its subdirectories.

Good GUI ftp clients

- 1.1. kasablanca
- 1.2. ftpcube

- 1.3. gftp
- 1.4. iglooftp
- 1.5. konqueror
- 1.6. filezilla

Console ftp clients

- 2.1. GNU Midnight Commander
- 2.2. ftp
- 2.3. yafc
- 2.4. ncftp

Problems With FTP And Firewalls

FTP frequently fails when the data has to pass through a firewall, because firewalls are designed to limit data flows to predictable TCP ports and FTP uses a wide range of unpredictable TCP ports. You have a choice of methods to overcome this.

Note: The Appendix II, "Codes, Scripts, and Configurations", contains examples of how to configure the VSFTPD Linux firewall to function with both active and passive FTP.

Client Protected By A Firewall Problem

Typically firewalls don't allow any incoming connections at all, which frequently blocks active FTP from functioning. With this type of FTP failure, the active FTP connection appears to work when the client initiates an outbound connection to the server on port 21. The connection then appears to hang, however, as soon as you use the ls, dir, or get commands. The reason is that the firewall is blocking the return connection from the server to the client (from port 20 on the server to a high port on the client). If a firewall allows all outbound connections to the Internet, then passive FTP clients behind a firewall will usually work correctly as the clients initiate all the FTP connections.

- Solution

  Table shows the general rules you'll need to allow FTP clients through a firewall:

Client Protected by Firewall – Required Rules for FTP

method	Source Address	ce Port	Destination Address	Destination Port	Connection Type
Allow outgoing control connections to server					
Channel	FTP	High[1]	FTP server[2]	21	New
	FTP server[2]	21	FTP client/network	High	Established[3]
Allow the client to establish data channels to remote server					

Active FTP	FTP server[2]	20	FTP client / network	High	New
	FTP client/network	High	FTP server [2]	20	Established[3]
FTP	FTP [illegible]	High	FTP server [2]	High	New
	FTP server[2]	High	FTP client/network	High	Established [3]

[1] Greater than 1024.

[2] In some cases, you may want to allow all Internet users to have access, not just a specific client server or network.

[3] Many home-based firewall/routers automatically allow traffic for already established connections. This rule may not be necessary in all cases.

Server Protected By A Firewall Problem

Typically firewalls don't let any connections come in at all. When a an incorrectly configured firewall protects an FTP server, the FTP connection from the client doesn't appear to work at all for both active and passive FTP.

- Solution

Table 15-2 Rules needed to allow FTP servers through a firewall.

Method	Source Address	source Port	Destination Address	Destination Port	Connection Type
Allow incoming control connections to server					
control Channel	FTP client/network[2]	High[1]	FTP server	21	New
	FTP server	21	FTP client/network[2]	High	Established[3]
Allow server to establish data channel to remote client					
FTP	FTP server	20	FTP client/network[2]	High	New
	FTP client/network[2]	High	FTP server	20	Established[3]
Passive FTP	FTP client/network[2]	High	FTP server	High	New
	FTP server	High	FTP client/network[2]	High	Established[3]

[1] Greater than 1024.

[2] In some cases, you may want to allow all Internet users to have access, not just a specific client server or network.

[3] Many home-based firewall/routers automatically allow traffic for already established connections. This rule may not be necessary in all cases.

chrooted ftp users

uncomment & edit vsftpd.conf
local_enable=YES
chroot_local_user=YES
the above line will enable for users to be chrooted under their home
this is for chroot list
local_enable=YES
chroot_local_user=NO
chroot_list_enable=YES
chroot_list_file=/etc/vsftpd.chroot_list

create vsftpd.chroot_list file under /etc and add the users which u want to chroot

NFS

NFSv2 uses the User Datagram Protocol (UDP) to provide a stateless network connection between the client and server. NFSv3 can use either UDP or Transmission Control Protocol (TCP) running over an IP network

/lib/modules/2.4.20-8/kernel/fs/nfsd/nfsd.

NFS port – 2049

NFS relies on Remote Procedure Calls (RPC) to route requests between clients and servers. RPC services under Linux are controlled by the portmap service. To share or mount NFS file systems, the following services work together:

- nfs – Starts the appropriate RPC processes to service requests for shared NFS file systems.

- nfslock – An optional service that starts the appropriate RPC processes to allow NFS clients to lock files on the server.

- portmap – The RPC service for Linux; it responds to requests for RPC services and sets up connections to the requested RPC service.

The following RPC processes work together behind the scenes to facilitate NFS services:

- rpc.mountd – This process receives mount requests from NFS clients and verifies the requested file system is currently

exported. This process is started automatically by the nfs service and does not require user configuration.

- rpc.nfsd – This process is the NFS server. It works with the Linux kernel to meet the dynamic demands of NFS clients, such as providing server threads each time an NFS client connects. This process corresponds to the nfs service.

- rpc.lockd – An optional process that allows NFS clients to lock files on the server. This process corresponds to the nfslock service.

- rpc.statd – This process implements the Network Status Monitor (NSM) RPC protocol which notifies NFS clients when an NFS server is restarted without being gracefully brought down. This process is started automatically by the nfslock service and does not require user configuration.

- rpc.rquotad – This process provides user quota information for remote users. This process is started automatically by the nfs service and does not require user configuration.

NFS and portmap

The portmap service under Linux maps RPC requests to the correct services. RPC processes notify portmap when they start, revealing the port number they are monitoring and the RPC program numbers they expect to serve. The client system then contacts portmap on the server with a particular RPC program number. The portmap service redirects the client to the proper port number so it can communicate with the requested service.

Because RPC-based services rely on portmap to make all connections with incoming client requests, portmap must be available before any of these services start.

The portmap service uses TCP wrappers for access control, and access control rules for portmap affect all RPC-based services. Alternatively, it is possible to specify access control rules for each of the NFS RPC daemons. The man pages for rpc.mountd and rpc.statd contain information regarding the precise syntax for these rul

/etc/hosts

1. /var/lib/nfs/xtab

2./var/lib/nfs/rmtab

rpc.mountd

The rpc.mountd program implements the NFS mount protocol. When receiving a MOUNT request from an NFS client, it checks the request against the list of currently exported file systems. If the client is permitted to mount the file system, rpc.mountd obtains a file handle for requested directory and returns it to the client.

Alternatively, you can export individual directories temporarily using exportfs host:/directory syntax. uid matching and readonlyfilesystem

Portmap is a server that converts RPC program numbers into DARPA protocol port numbers. It must be running in order to make RPC calls.

When an RPC server is started, it will tell portmap what port number it is listening to, and what RPC program numbers it is prepared to serve. When a client wishes to make an RPC call to a given program number, it will first contact portmap on the server machine to determine the port number where RPC packets should be sent.

$ /etc/exports

```
  /data/files           *(ro,sync)
  /home                 192.168.1.0/24(rw,sync)
  /data/test            *.my-site.com(rw,sync)
  /data/database        192.168.1.203/32(rw,sync)
```

/etc/exports option

$cat /proc/fs/nfs/exports (runtime check)

Try the following

Edit exports file

/opt/data
192.168.1.65(rw,wdelay,root_squash,no_subtree_check,anonuid=65534,anongid=65534)

192.168.1.45 (rw,no_root_squash,)

try mounting from the client, in my case client ip is 192.168.1.65 and check the permissons by root and user also try from the client 192.168.1.45 and chk permissions

Client

Mount 192.168.1.75:/opt/data /data

More options

/opt/data station1 (rw,wdelay,all_squash,anonuid=150,anongid=100)

Now try mounting from 192.168.1.65 (point station1 --> 192.168.1.65 in /etc/hosts file) and chk the permisson by login user and root.

More options

we can also restrict by domain.

/opt/deploy/stic *.example.com (rw) *.workgroup.com (ro,sync)

Here's the complete list of mapping options:

root_squash

Map requests from uid/gid 0 to the anonymous uid/gid. Note that this does not apply to any other uids that might be equally sensitive, such as user bin.

no_root_squash

Turn off root squashing. This option is mainly useful for diskless clients.

all_squash

Map all uids and gids to the anonymous user. Useful for NFS-exported public FTP directories, news spool directories, etc.The opposite option is no_all_squash, which is the default set-ting.

anonuid and anongid

These options explicitly set the uid and gid of the anonymous account. This option is primarily useful for PC/NFS clients,where you might want all requests appear to be from one user. As an example, consider the export entry for /home/joe in the exam-ple section below, which maps all requests to uid 150 (which is supposedly that of user joe).

General Options

exportfs understands the following export options:

secure

This option requires that requests originate on an internet port less than IPPORT_RESERVED (1024). This option is on by default. To turn it off, specify insecure.

rw

Allow both read and write requests on this NFS volume. The default is to disallow any request which changes the filesystem. This can also be made explicit by using the ro option.

async

This option allows the NFS server to violate the NFS protocol and reply to requests before any changes made by that request have been committed to stable storage (e.g. disc drive).Using this option usually improves performance, but at the cost that an unclean server restart (i.e. a crash) can cause data to be lost or corrupted.

sync

Reply to requests only after the changes have been committed to stable storage (see async above).In releases of nfs-utils upto and including 1.0.0, this option was the default. In this and future

releases, sync is the default, and async must be explicit requested if needed.To help make system adminstrators aware of this change, 'exportfs' will issue a warning if neither sync nor async is specified.

no_wdelay

This option has no effect if async is also set. The NFS server will normally delay committing a write request to disc slightly if it suspects that another related write request may be in progress or may arrive soon. This allows multiple write requests to be committed to disc with the one operation which can improve performance. If an NFS server received mainly small unrelated requests, this behaviour couldactually reduce performance, so no_wdelay is available to turn it off. The default can be explicitly requested with the wdelay option.

subtree_check

This option enables subtree checking, which does add another level of security, but can be unreliability in some circumstances.If a subdirectory of a filesystem is exported, but the whole filesystem isn't then whenever a NFS request arrives, the server must check not only that the accessed file is in the appropriate filesystem (which is easy) but also that it is in the exported tree (which is harder). This check is called the subtree_check.

In order to perform this check, the server must include some information about the location of the file in the "filehandle" that is given to the client. This can cause problems with accessing files that are renamed while a client has them open (though in many simple cases it will still work).

Try Mount Options in fstab

192.168.1.33:/opt/deploy/stic /mnt nfs
rw,hard,intr,rsize=8192,wsize=8192 0 0

bg

Retry mounting in the background if mounting initially fails

fg

Mount in the foreground

soft Use soft mounting

If an NFS file operation has a major timeout then report an I/O error to the calling program. The default is to continue retrying NFS file operations indefinitely.

hard Use hard mounting

If an NFS file operation has a major timeout then report "server not responding" on the console and continue retrying indefinitely. This is the default.

rsize=n

The amount of data NFS will attempt to access per read operation.

The default is dependent on the kernel. For NFS version 2, set it to 8192 to assure maximum throughput.

wsize=n

The amount of data NFS will attempt to access per write operation. The default is dependent on the kernel. For NFS version 2, set it to 8192 to assure maximum throughput.

nfsvers=n

The version of NFS the mount command should attempt to use tcp
Attempt to mount the filesystem using TCP packets: the default is UDP.

intr

If the filesystem is hard mounted and the mount times out, allow for the process to be aborted using the usual methods such as CTRL-C and the kill command.

nolock

Disable NFS locking. Do not start lockd. This has to be used with some old NFS servers that don't support lock-ing.

Some important nfs mount options in Linux.

tcp - Specifies for the NFS mount to use the TCP protocol instead of UDP.

rsize=8192 and wsize=8192 - These settings speed up NFS communication for reads (rsize) and writes (wsize) by setting a larger data block size, in bytes, to be transferred at one time. Do performance tests before changing these values.

hard or soft - Specifies whether the program using a file via an NFS connection should stop and wait (hard) for the server to come back online if the host serving the exported file system is unavailable, or if it should report an error (soft). If hard is specified, the user cannot terminate the process waiting for the NFS communication to resume unless the intrsoft, is specified, the user can set an additional timeo=<value> option, where <value> specifies the number of seconds to pass before the error is reported. option is also specified.

nolock - Disables file locking. This setting is occasionally required when connecting to older NFS servers.

noexec - Prevents execution of binaries on mounted file systems. This is useful if the system is mounting a non-Linux file system via NFS containing incompatible binaries.

intr - Allows NFS requests to be interrupted if the server goes down or cannot be reached.

nfsvers=2 or nfsvers=3 - Specifies which version of the NFS protocol to use.

nosuid - Disables set-user-identifier or set-group-identifier bits. This prevents remote users from gaining higher privileges by running a setuid program.

There are many other options. But the above ones are very important.

Practicals:

On server pc (server ip - 192.168.1.40)

First check nfs install or not

```
$ rpm -qa | grep -i nfs
  nfs-utils-1.0.9-44.el5


  nfs-utils-lib-1.0.8-7.6.el5
```

if you have installation dvd then run following command:

```
$ rpm -ivh nfs-utils-1.0.9-44.el5


$ rpm -ivh nfs-utils-lib-1.0.8-7.6.el5
```

if not then install through internet

```
$ yum install nfs*
```

Then start nfs service

```
$ service nfs start
  Starting NFS services:                                     [  OK
]


  Starting NFS quotas:                                       [  OK
]


  Starting NFS daemon:                                       [  OK
]


  Starting NFS mountd:                                       [  OK
]
```

NFS major file call /etc/exports

```
$exports --> this command became blank your exports file

$ vi /etc/exports
```

Type following line in exports file

```
/mnt/test      *(sync,rw)
/opt/funny     *(sync,rw)
```

$ exportfs -a --> this cmnd reread /etc/exports file

```
$ showmount -e localhost
  Export list for localhost:


  /mnt/test  *


  /opt/funny *
```

$nfsstat → nfs statistics.

From client side (client ip – 192.168.1.50)

$ service nfs start (as a root)

```
$ showmount -e 192.168.1.40
  /mnt/test  *

  /opt/funny *
```

Mount the shared partition as follow:

```
$ mount 192.168.1.40:/mnt/test /mnt/

$ cd /mnt/

$ ls
```

Create any file in client side and check on server pc you can see the same.

For permanent mount the directory type following entry in fstab file:

```
$ vi /etc/fstab

192.168.1.40:/mnt/test   /mnt      nfs  _netdev,defaults    0    0
192.168.1.40:/opt/funny  /opt      nfs  _netdev,defaults    0    0
```

Check with the following cmds

$ rpcinfo -p --> check which rpc based services on, -p means print

$ rpcinfo -p 192.168.1.40 --> which service on server site

$ telnet 192.168.1.40 2049 --> check nfs port open or not on server side

Network Info Service (NIS)

[Directory Services] like LDAP and DNS
by Sun MicroSystems

What does it do ?

* Provides unified login and authentication, NS for a group of machines.
* Like LDAP, it handles passwords, groups, protocols, networks, services.

Advantages of NIS :

1. Central Information Store
2. Security because of encrypted maps [dbs]
3. Performance bcos of indexed maps
4. Can be used for DNS
5. Authentication for Samba, Apache etc and also Local instead of /etc/hosts and /etc/passwd, which can then be deleted

Alert : Disable Firewalls or NIS will not work since, by default, RPC services are blocked by FWs [in RHL]
Exam Alert :

1. Make sure you put FWs off before doing anything in NIS
2. Make sure you start portmap BEFORE you start NFS
3. Make sure you use 'showmount -e' before you do any 'mount'
4. All errors are mostly due to incorrect syntax in /etc/exports and especially watch out for the space between the options ()
5. If you use 'authconfig' for client config, make sure you disable it in 'ntsysv' imasap
6. Check /etc/securenets if you are being blocked or wish to
7. Check /etc/ypserv.conf if you cannot query maps or wish to

Daemon Name	Purpose
portmap	The foundation RPC daemon upon which NIS runs.
yppasswdd	Lets users change their passwords on the NIS server from NIS clients
ypserv	Main NIS server daemon
ypbind	Main NIS client daemon
ypxfrd	Used to speed up the transfer of very large NIS maps

```
rpcinfo -p localhost
   program vers proto   port
   100000    2   tcp    111  portmapper
   100000    2   udp    111  portmapper
   100009    1   udp    681  yppasswdd
   100004    2   udp    698  ypserv
   100004    1   udp    698  ypserv
   100004    2   tcp    701  ypserv
   100004    1   tcp    701  ypserv
```

NIS Server part I

```
1.   $ vi /etc/sysconfig/network  #put NISDOMAIN=altnix
2.   $ domainname altnix
3.   Configure /etc/yp.conf
      altnix server server1.altnix.local
4.   $ service ypserv start
5.   $ service yppasswdd start  [OPTIONAL]
6.   $ domainname
7.   $ nisdomainname
      The above 2 commands MUST show the right output
8.   $ cd /var/yp/
9.   $ /usr/lib/yp/ypinit -m
      Ctrl D
```

So what is an NIS domain ?
A: A group of hosts that use the same set of maps, form an NIS domain
   All of the m/cs in an NIS domain will share the same pwd,hosts and grp
   info.

I. NFS Server part

```
1.        Configure /etc/exports
$ vi /etc/exports
/home  *(rw,sync)
$ exportfs -a
2.        $ service portmap restart
3.        $ service nfs restart
    Testing : Check if all is ok with
    $ rpcinfo -p
    $ showmount -e localhost
```

II. NIS Client part

```
1.             service portmap restart
2.             showmount -e NFSServer
3.             mount NFSserver:/home /home
4.             Configure yp.conf
       Do exactly the same as for server
5.             service ypbind restart
      or
use authconfig* which will do 4,5,6 automatically
```

Testing the NIS Server from the Client

Log in as foo

```
$ ypwhich                <-- Where is the NIS Master Database
$ ypwhich -m             <-- Where is the NIS Master Database + maps
$ ypcat -x               <-- Shows list of NIS maps from NIS server
$ ypcat passwd           <-- Shows details of NIS passwd map db
                             /var/yp/altnix/passwd.byname
$ ypmatch foo passwd     <-- Check if user foo exists in the NIS pwd
db
$ yppoll passwd.byname   <-- Info about a specific map [root only]
```

You have to have the 'service yppasswdd' running on the server to do the foll:

```
$ ypchfn foo             <-- Change finger info of foo in NIS pwd db
map
$ yppasswd foo           <-- Change foo's pwd in NIS pwd db map
```

i.e. /var/yp/altnix/passwd.byname passwd.byuid

Note : His local system login pwd changes accordingly bcos yppasswd* actually changes the passwd of foo in /etc/passwd and then pushes the changes to 2 other file :

/var/yp/altnix/passwd.byname

/var/yp/altnix /passwd.byuid

```
$ ypmatch nisuser passwd
$ getent passwd nisuser
```

III - More NIS Server Security

/etc/ypserv.conf [On NIS Server]

* on server
* This file is used to control hosts/users who can use your NIS server

syntax : host:domain:map:security

* There is one entry per line.
* All rules are tried one by one. If no match is found, access to a map is allowed.
* Following options exist:

files: 30

This option specifies, how many database files should be cached by ypserv. If 0 is specified, caching is disabled. Decreasing this number is only possible, if ypserv is restarted.

xfr_check_port: [<yes>|no]
With this option enabled, the NIS master server have to run on a port < 1024. The default is "yes" (enabled).The field descriptions for the access rule lines are:

host IP address. Wildcards are allowed.
Examples:
131.234. = 131.234.0.0/255.255.0.0
131.234.214.0/255.255.254.0

domain specifies the domain, for which this rule should be applied. An asterix as wildcard is allowed.

map name of the map, or asterisk for all maps.

security one of none, port, deny:

none always allow access.

port allow access if from port < 1024. Otherwise do not allow access.

deny deny access to this map.

EXAMPLE : Try these entries in /etc/ypserv.conf

eg 192.168.0.10:*:*:deny

eg *:*:passwd.byname:deny <-- All hosts/users will be denied access from any domain to the foll map

****

Logging In Via Telnet

Try logging into the NIS client via telnet if it is enabled

```
[root@bigboy tmp]# telnet 192.168.1.201
Trying 192.168.1.201...
Connected to 192.168.1.201.
Escape character is '^]'.
Red Hat Linux release 9 (Shrike)
Kernel 2.4.20-6 on an i686
login: nisuser
Password:
Last login: Sun Nov 16 22:03:51 from 192-168-1-100.simiya.com
[nisuser@smallfry nisuser]$

yppasswd -p nisuser
Changing NIS account information for nisuser on bigboy.my-site.com.
Please enter root password:
Changing NIS password for nisuser on bigboy.my-site.com.
Please enter new password:
Please retype new password:

The NIS password has been changed on bigboy.my-site.com.
```

NIS Troubleshooting

Troubleshooting is always required as any part of your daily routine, NIS is no exception. Here are some simple steps to follow to get it working again.

1. The rpcinfo provides a list of TCP ports that your NIS client or server is using. Make sure you can TELNET to these ports from the client to the server and vice versa. If this fails, make sure all the correct NIS daemons are running and that there are no firewalls blocking traffic on the network or on the servers themselves. These ports change from time to time, so memorizing them won't help much.

The example tests from the client to the server.

```
[root@bigboy tmp]# rpcinfo -p
     program vers proto   port
     100000    2   tcp    111  portmapper
     100000    2   udp    111  portmapper
     100024    1   udp  32768  status
     100024    1   tcp  32768  status
     391002    2   tcp  32769  sgi_fam
     100009    1   udp   1018  yppasswdd
     100004    2   udp    611  ypserv
     100004    1   udp    611  ypserv
     100004    2   tcp    614  ypserv
     100004    1   tcp    614  ypserv
     100007    2   udp    855  ypbind
     100007    1   udp    855  ypbind
     100007    2   tcp    858  ypbind
     100007    1   tcp    858  ypbind
  600100069    1   udp    874  fypxfrd
  600100069    1   tcp    876  fypxfrd


 [root@bigboy tmp]#

[root@smallfry tmp]# telnet 192.168.1.100 858
Trying 10.41.32.71...
Connected to 10.41.32.71.
Escape character is '^]'.
^]
telnet> quit
Connection closed.
[root@smallfry tmp]#
```

2. Always use the ypmatch, getent, and ypwhich commands to check your NIS connectivity. If there is any failure, check your steps over again and you should be able to find the source of your problem.

3. Do not fail to create a user's home directory, set its permissions, and copy the /etc/skel files correctly. If you forget, which is a common error, your users may have incorrect login prompts and no ability to create files in their home directories.

It can never be overemphasized that one of the best places to start troubleshooting is by looking in your error log files in the /var/log directory. You'll save a lot of time and effort if you always refer to them whenever the problem doesn't appear to be obvious.

Installation of autofs.

Install autofs using rpm package.

```
[root@tenouk ~]# mount /dev/cdrom
[root@tenouk ~]# cd /mnt/cdrom/RedHat/RPMS
[root@tenouk ~]# rpm -Uhv autofs-3.1.7-28.i386.rpm
[root@tenouk ~]# cd /
[root@tenouk ~]# umont /dev/cdrom
```

Start, stop and restart autofs.

```
[root@tenouk ~]# /sbin/service autofs start
[root@tenouk ~]# /sbin/service autofs stop
[root@tenouk ~]# /sbin/service autofs restart
```

Setting of autofs automatic start.

```
[root@tenouk ~]# /sbin/chkconfig --level 35 autofs on
```

Confirmation of autofs automatic start

```
[root@tenouk ~]# /sbin/chkconfig --list autofs
```

Setting which uses NIS

The configuration on the NIS server

```
[root@tenouk ~]# vi /etc/auto.master
/nfs /etc/auto.home --timeout 60

[root@tenouk ~]# vi /etc/auto.home
home -rw,hard,intr,nolock compaq:/home

[root@tenouk ~]# vi /var/yp/Makefile
all: passwd group hosts rpc services netid protocols mail \
shadow auto.home \
# netgrp shadow publickey networks ethers bootparams printcap \
# amd.home auto.master auto.home auto.local passwd.adjunct \
# timezone locale netmasks
```

Allow normal user to mount linux partitions, usb stick / pen device

by nixcraft ·

You need to use autofs. It is use to mount file system on demand. Usually autofs is invoked at system boot time with the start parameter and at shutdown time with the stop parameter. The autofs script can also manually be invoked by the system administrator to shut down, restart or reload the automounters.

autofs will consult a configuration file /etc/auto.master to find mount points on the system.

i) Install autofs if not installed. if you are using Debian / Ubuntu Linux, enter:

```
# apt-get install autofs
```
ii) Create dekstop group and add user jimmy to this group:
```
# groupadd desktop
# usermod -G video,desktop jimmy
# chmod -R a+rx /var/autofs/misc
```
iii) Configure autofs so that usb stick can be accessed:
```
# vi /etc/auto.misc
```
iv) Append following text to auto.misc:
```
usb -fstype=auto, user, sync, nodev, nosuid, gid=desktop, umask=002
:/dev/sda1
d -fstype=vfat, user, sync, nodev, nosuid,gid=desktop, umask=002
:/dev/hda2
```
Where,

- **usb** : Is directory name, which can be accessed via /var/autofs/misc/usb directory. User in desktop group just need to type cd command (cd /var/autofs/misc/usb) to change the directory.
- **-fstype- auto, user, sync, nodev, nosuid, gid-desktop, umask-002** :- All these are options used to mount the file system by automounter.
- **auto**: File system is automatically determined by kernel.
- **user**: Normal user are allowed to mount devices
- **nodev**: Do not interpret character or block special devices on the file system.
- **nosuid**: Do not allow set-user-identifier or set-group-identifier bits to take effect. This is security feature.
- **gid=desktop**: This allows file system mounted as as group dekstop. As we have added user jimmy to this group already.
- **umask=002**: Setup umask so that users in group desktop can write data to device.

Please note that without gid and umask option normal user cannot write data to device.

v)Restart the autofs:
```
#/etc/init.d/autofs restart
```
vi) Test it as user jimmy (make sure usb stick/pen is inserted into usb port):
```
$ ls /var/autofs/misc/usb
$ cd /var/autofs/misc/usb
$ mkdir testdir
$ ls -l
```

DHCP: Dynamic Host Configuration Protocol

*** A DHCP relay agent**

These tools all use a modular API which is designed to be sufficiently general that it can easily be made to work on POSIX-compliant operating systems and also non-POSIX systems like Windows NT and MacOS.

The DHCP server, client and relay agent are provided both as reference implementations of the protocol and as working, fully-featured sample implementations.

Both the client and the server provide functionality that, while not strictly required by the protocol, is very useful in practice. The DHCP server also makes allowances for non-compliant clients which one might still like to support.

This tutorial describes how to set up a DHCP server (ISC-DHCP) for yourlocal network.

DHCP is short for "Dynamic Host Configuration Protocol",it's a protocol that handles the assignment of IP addresses, subnet masks, default routers, and other IP parameters to client PCs that don't have a static IP address. Such computers try to find a DHCP server in their local network which in turn assigns them an IP address,gateway, etc. so that they can connect to the internet or other computers from the local network.

current situation:

* network 192.168.10.0, subnetmask 255.255.255.0, broadcast address 192.168.10.255.

* gateway to the internet is 192.168.10.10; on the gateway there's no DHCP server.

* DNS servers I can use are 202.88.130.15 and 202.88.130.67

* I have a pool of 30 IP addresses (192.168.10.200 - 192.168.10.229) that can be dynamically assigned to client PCs and that are not already in use.

IP address 192.168.10.10 which will act as DHCP server.

2 Download and Install the DHCP Package

NOTE:

By default dhcp packages are installed in RHEL

See that the following is installed on the Server: dhcp-3.0pl2-6.14

$ rpm -q dhcp

On the client side, this package "dhclient-3.0pl2-6.14" is installed.

$ rpm -q dhclient

If DHCP packages aren't installed, download it from

http://www.rpmseek.com and install.

3. Configuration Part: ISC DHCP Server

Copy the DHCP configuration filke

$cp /usr/share/doc/dhcp-3.0pl2/dhcpd.conf.sample /etc/dhcpd.conf

/etc/dhcpd.conf

ddns-update-style: You can tell the DHCP server to update a DNS server if the IP address of a server in your LAN has changed (because it has been assigned a different IP by DHCP).

As we do not run servers in our LAN or always give them static IP addresses (which is a

good idea for servers...) we don't want to update DNS records so we set this to none.

ddns-update-style interim;

ignore client-updates;

 # Define the scope

subnet 192.168.10.0 netmask 255.255.255.0

{

range dynamic-bootp 192.168.10.177 192.168.10.188;

range dynamic-bootp 192.168.10.124 192.168.10.130;

Set the amount of time in seconds that a client may keep the IP address

A client can tell the DHCP server for how long it would like to get an IP address.

If it doesn't do this, the server assigns an IP address for default-lease-time seconds;

if it does, the server grants the requested time, but only up to max-lease-time seconds.

default-lease-time 10 ; # In secs. If this is not explictly given, then the default is 1 day.

max-lease-time 12;

Set the broadcast address and subnet mask to be used by the DHCP clients

option broadcast-address 192.168.10.255;

option subnet-mask 255.255.255.0;

option routers 192.168.10.10;

option nis-domain "altnix.com";

option domain-name "altnix.com";

Set the DNS server to be used by the DHCP clients

option domain-name-servers 192.168.10.10   192.168.10.20 ;

```
# If you specify a WINS server for your Windows clients,
# you need to include the following option in the dhcpd.conf file:
option  netbios-name-servers  192.168.10.66;
option  netbios-node-type  8;



# You can also assign specific IP addresses based on the clients' ethernet MAC address
# as follows (Host's name is "fcfive.altnix.com"
host  win2k3box.altnix.com {
next-server win2k3box.altnix.com  # You could assign your own
hostname
hardware ethernet   12:34:56:78:AB:CD;
fixed-address 192.168.10.100;
}
}
```

- If you wanna peek into more stuff, Check the dhcp-options man page

4. How to Get DHCP Started

```
$ touch /var/lib/dhcpd/dhcpd.leases
```

--> Test whether your config file is OK.

```
$ dhcpd   -t
```

--> Test whether your leases file file is A-OK.

```
$ dhcpd   -T
```

This lease ascii db is vital and documents acquired, renewed or released leases otherwise the DHCP server will not function.

--> Start the dhcp service [/usr/sbin/dhcpd]

```
$ service dhcpd restart
```

--> Check whether dhcpd service is started with the following :

```
$  dhcpd  -f
$  ps ax | grep dhcpd
```

5. Configuration Part: ISC DHCP Client

sends a standardized DHCP broadcast request packet to the DHCP server with a source IP address of 255.255.255.255.

Edit file /etc/sysconfig/network-scripts/ifcfg-eth0 :

change BOOTPROTO=none or static to BOOTPROTO=dhcp

OR

$ netconfig

Check the box with “Use dynamic IP configuration (BOOTP/DHCP)”

This will eventually make change auto to the file above.

Restart network

4 service network restart

Check to see if DHCP server is up

$ dhcpd -f DHCP-server-IP/Hostname

Check your IP :

$ ifconfig

Note that the new IP address assigned to the client is reassigned dynamically by the server from the ranges given in /etc/dhcpd.conf on the server

NOTE:-

DHCP uses the BOOTP protocol for its communication between the client and server.

Make sure there are no firewalls blocking this traffic. DHCP servers expect requests on UDP port 67 and the DHCP clients expect responses on UDP port 68.

The DHCP server writes all current IP address "leases" to the file /var/lib/dhcp3/dhcpd.leases so you should also find the lease there:

$cat /var/lib/dhcp3/dhcpd.leases

lease 192.168.10.229 {

starts 2 2006/09/19 14:01:31;

ends 3 2006/09/20 14:01:31;

binding state active;

next binding state free;

hardware ethernet 00:0c:76:8b:c4:16;

uid "\001\000\014v\213\304\026";

```
client-hostname "trinity";
}
```

What all can a DHCP server provide Clients ?

```
1. IP                   - range
2. netmask              - option subnet-mask
3. BC                   -
4. nameserver           - 'option domain-name-servers'
5. domain               - 'option domain-name'
6. NIS domain           - 'option nis-domain-name'
6. MAC addr-based IP    - 'hardware ethernet' and 'fixed-address'
7. default lease time   - 'default-lease-time'
8. max lease time       - 'max-lease-time'
9  gateway              - 'option routers'
   For netbios/Samba      option netbios-node-type 2
                          option netbios-name-server
```

Advantages of DHCP :

1. Easy configuration if many many clients
2. Saves IPs
2. fixed IP for certain clients
3. Automatic config of the above 9 points

Disadvantages :

Not even one
ddns-update-style interim;
ignore client-updates;

Selects point-to-point node (default is hybrid). Don't change this unless you understand Netbios very well

DHCP Clients Obtaining 169.254.0.0 Addresses

Whenever Microsoft DHCP clients are unable to contact their DHCP server they default to selecting their own IP address from the 169.254.0.0 network until the DHCP server becomes available again. This is frequently referred to as Automatic Private IP Addressing (APIPA). Here are some steps you can go through to resolve the problem:

- Ensure that your DHCP server is configured correctly and use the pgrep command discussed earlier to make sure the DHCP process is running. Pay special attention to your 255.255.255.255 route, especially if your DHCP server has multiple interfaces.

- Give your DHCP client a static IP address from the same range that the DHCP server is supposed to provide. See whether you can ping the DHCP server. If you cannot, double-check your cabling and your NIC cards.

- DHCP uses the BOOTP protocol for its communication between the client and server. Make sure there are no firewalls blocking this traffic. DHCP servers expect requests on UDP port 67 and the DHCP clients expect responses on UDP port 68. Use tcpdump on the server's NIC to verify the correct traffic flows.

Other DHCP Failures

If the DHCP server fails to start then use your regular troubleshooting techniques outlined in Chapter 4, "Simple Network Troubleshooting", to help rectify your problems. Most problems with an initial setup are often due to:

- Incorrect settings in the /etc/dhcpd.conf file such as not defining the networks for which the DHCP server is responsible;

- Firewall rules that block the DHCP bootp protocol on UDP ports 67 and 68;

- Routers failing to forward the bootp packets to the DHCP server when the clients reside on a separate network.

Always check your /var/logs/messages file for dhcpd errors and remember that mandatory keywords in your configuration file may change when you upgrade your operating system. Always read the release notes to be sure.

When a client is to be booted, its boot parameters are determined by consulting that client's *host* declaration (if any), and then consulting any *class* declarations matching the client, followed by the *pool*, *subnet* and *shared-network* declarations for the IP address assigned to the client. Each of these declarations itself appears within a lexical scope, and all declarations at less specific lexical scopes are also consulted for client option declarations. Scopes are never considered twice, and if parameters are declared in more than one scope, the parameter declared in the most specific scope is the one that is used.

When dhcpd tries to find a *host* declaration for a client, it first looks for a *host* declaration which has a *fixed-address* declaration that lists an IP address that is valid for the subnet or shared network on which the client is booting. If it doesn't find any such entry, it tries to find an entry which has no *fixed-address* declaration.

EXAMPLES

A typical dhcpd.conf file will look something like this:

global parameters...

```
subnet 204.254.239.0 netmask 255.255.255.224 {


subnet-specific parameters...
  range 204.254.239.10 204.254.239.30;

}

subnet 204.254.239.32 netmask 255.255.255.224 {

  subnet-specific parameters...
  range 204.254.239.42 204.254.239.62;

}

subnet 204.254.239.64 netmask 255.255.255.224 {

  subnet-specific parameters...
  range 204.254.239.74 204.254.239.94;

}

group {

  group-specific parameters...

  host zappo.test.isc.org {

    host-specific parameters...

  }

  host beppo.test.isc.org {

    host-specific parameters...

  }

  host harpo.test.isc.org {

    host-specific parameters...

  }

}
```

Notice that at the beginning of the file, there's a place for global parameters. These might be things like the organization's domain name, the addresses of the name servers (if they are common to the entire organization), and so on. So, for example:

```
        option domain-name "isc.org";
        option domain-name-servers ns1.isc.org, ns2.isc.org;
```

As you can see in Figure 2, you can specify host addresses in parameters using their domain names rather than their numeric IP addresses. If a given hostname resolves to more than one IP address (for example, if that host has two ethernet interfaces), then where possible, both addresses are supplied to the client.

The most obvious reason for having subnet-specific parameters as shown in Figure 1 is that each subnet, of necessity, has its own router. So for the first subnet, for example, there should be something like:

```
option routers 204.254.239.1;
```

Note that the address here is specified numerically. This is not required - if you have a different domain name for each interface on your router, it's perfectly legitimate to use the domain name for that interface instead of the numeric address. However, in many cases there may be only one domain name for all of a router's IP addresses, and it would not be appropriate to use that name here.

In Figure 1 there is also a *group* statement, which provides common parameters for a set of three hosts - zappo, beppo and harpo. As you can see, these hosts are all in the test.isc.org domain, so it might make sense for a group-specific parameter to override the domain name supplied to these hosts:

```
option domain-name "test.isc.org";
```

Also, given the domain they're in, these are probably test machines. If we wanted to test the DHCP leasing mechanism, we might set the lease timeout somewhat shorter than the default:

```
max-lease-time 120;        default-lease-time 120;
```

TcpWrappers

Almost all BSD / UNIX / Linux like operating systems are compiled with TCP Wrappers support. For e.g. Solaris 9, various Linux / *BSD distributions, and Mac OS X have TCP Wrappers configured to run out-of-the-box. It is a library which provides simple access control and standardized logging for supported applications which accept connections over a network.

TCP Wrapper is a host-based Networking ACL system, used to filter network access to Internet. TCP wrappers was original written to monitor and stop cracking activities on the UNIX workstation in 90s. It was best solution in 90s to protect the UNIX workstations over the Internet. However it has few disadvantages:

1. All UNIX apps must be compiled with the libwrap library.
2. The wrappers do not work with RPC services over TCP.
3. The user name lookup feature of TCP Wrappers uses identd to identify the username of the remote host. By default, this feature is disabled, as identd may appear hung when there are large number of TCP connections.

However, it has one strong advantage over firewall. It works on the application layer. It can filter requests when encryption is used.

Basically, you need to use both host based and network based security. Common services such as pop3, ftp, sshd, telnet, r-services are supported by TCP Wrappers.

TCPD Benefits

1. **Logging** - Connections that are monitored by tcpd are reported through the syslog facility.
2. **Access Control** - tcpd supports a simple form of access control that is based on pattern matching. You can evern hook the execution of shell commands / script when a pattern matches.
3. **Host Name Verification** - tcpd verifies the client host name that is returned by the address->name DNS server by looking at the host name and address that are returned by the name->address DNS server.
4. **Spoofing Protection**

How do I Find Out If Program Is Compiled With TCP Wrappers Or Not?

To determine whether a given executable daemon /path/to/daemon supports TCP Wrapper, check the man page, or ennter:

```
$ ldd /path/to/daemon | grep libwrap.so
```

If this command returns any output, then the daemon probably supports TCP Wrapper. In this example, find out of if sshd supports tcp wrappers on not, enter:

```
$ whereis sshd
```

Sample Output:

```
sshd: /usr/sbin/sshd /usr/share/man/man8/sshd.8.gz
```

```
$ ldd /usr/sbin/sshd | grep libwrap.so
```

Sample Output:

```
        libwrap.so.0 => /lib64/libwrap.so.0 (0x00002b759b381000)
```

ldd is used to see if libwrap.so is a dependency or not. An alternative to TCP Wrapper support is packet filtering using iptables.

Important Files

- **tcpd** - access control facility for internet services.
- **/etc/hosts.allow** - This file describes the names of the hosts which are allowed to use the local INET services, as decided by the /usr/sbin/tcpd server.
- **/etc/hosts.deny** - This file describes the names of the hosts which are NOT allowed to use the local INET services, as decided by the /usr/sbin/tcpd server.
- If the same client / user / ip is listed in both hosts.allow and hosts.deny, then hosts.allow takes precedence and access is permitted. If the client is listed in hosts.allow, then is access permitted. If the client is listed in hosts.deny, then access is denied.
- **tcpdchk** and **tcpdmatch** - test programs for tcpd

Syntax (format) Of Host Access Control Files

Both /etc/hosts.allow and /etc/hosts.deny uses the following format:

daemon_list : client_list **[: shell_command]**

Where,

- **daemon_list** - a list of one or more daemon process names.
- **client_list** - a list of one or more host names, host addresses, patterns or wildcards that will be matched against the client host name or address.

WildCards

The access control language supports explicit wildcards (quoting from the man page):

ALL The universal wildcard, always matches.

LOCAL Matches any host whose name does not contain a dot character.

UNKNOWN
Matches any user whose name is unknown, and matches any host whose name or address are unknown. This pattern should be used with care: host names may be unavailable due to temporary name server problems. A network address will be unavailable when the software cannot figure out what type of network it is talking to.

KNOWN Matches any user whose name is known, and matches any host whose name and address are known. This pattern should be used with care: host names may be unavailable due to temporary name server problems. A network address will be unavailable when the soft-ware cannot figure out what type of network it is talking to.

PARANOID
Matches any host whose name does not match its address. When tcpd is built with -DPARANOID (default mode), it drops requests from such clients even before looking at the access control tables. Build without -DPARANOID when you want more control over such requests.

TCPD Configuration Examples

Set default policy to to deny access. Only explicitly authorized hosts are permitted to access. Update /etc/hosts.deny as follows:

```
 # The default policy (no access) is implemented with a trivial
deny file
ALL: ALL
```

Above will denies all service to all hosts, unless they are permitted access by entries in the allow file. For example, allow access as follows via /etc/hosts.allow:

```
  ALL: LOCAL @devels
  ALL: .nixcraft.net.in EXCEPT boobytrap.nixcraft.net.in
```

Log and deny access (booby traps) - we do not allow connections from crackers.com:

```
ALL : .crackers.com \
    : spawn (/bin/echo %a from %h attempted to access %d >> \
      /var/log/connections.log) \
    : deny
```

A Typical UNIX Example

Allow access to various service inside LAN only via /etc/hosts.allow:

```
popd : 192.168.1.200 192.168.1.104
imapd : 192.168.1.0/255.255.255.0
sendmail : 192.168.1.0/255.255.255.0
sshd : 192.168.1.2 172.16.23.12
```

Deny everything via /etc/hosts.deny:

```
 ALL : ALL
```

Reject All Connections

Restrict all connections to non-public services to localhost only. Suppose sshd and ftpd are the names of service which must be accessed remotely. Edit /etc/hosts.allow. Add the following lines:

```
sshd ,ftpd : ALL
ALL: localhost
```

Save and close the file. Edit /etc/hosts.deny. Add the following line:

```
ALL: ALL
```

Default Log Files

TCP Wrappers will do all its logging via syslog according to your /etc/syslog.conf file. The following table lists the standard

locations where messages from TCP Wrappers will appear:

1. AIX – /var/adm/messages
2. HP-UX – /usr/spool/mqueue/syslog
3. Linux – /var/log/messages
4. FreeBSD / OpenBSD / NetBSD – /var/log/messages
5. Mac OS X – /var/log/system.log
6. Solaris – /var/log/syslog

Use the following command to view logs:

```
# tail -f /path/to/log/file
# grep 'ip' /path/to/log/file
# egrep -i 'ip|hostname' /path/to/log/file
```

How Do I Predicts How The Tcp Wrapper Would Handle a Specific Request For Service?

Use tcpdmatch command. predict how tcpd would handle a sshd request from the local system:

```
tcpdmatch sshd localhost
```

The same request, pretending that hostname lookup failed:

```
tcpdmatch sshd 192.168.1.5
```

To predict what tcpd would do when the client name does not match the client address:

```
tcpdmatch sshd paranoid
```

Replace sshd with in.telnetd, or ftpd and so on. You can use all daemon names specified in inetd.conf or xinetd.conf file.

How do I Examines My TCP Wrapper Config File?

Use tcpdchk command toexamines your tcp wrapper configuration and reports all potential and real problems it can find.

```
tcpdchk
tcpdchk -v
```

A Note About TCP Wrappers and Firewall

- You need to use both (firewall and tcpd) to fight against crackers.
- TCP Wrappers are most commonly employed to match against IP addresses and host level protection.
- Never configure TCP Wrappers on firewall host.
- Put TCP Wrappers on all UNIX / Linux / BSD workstations.
- Do not use NIS (YP) netgroups in TCP Wrappers rules.
- Put TCP Wrappers behind a firewall systems as TCP Wrappers is no substitute for netfilter or pf firewall.
- TCP Wrappers does provide increased security as firewall cannot examine encrypted connections (read as packets).

Xinetd

xinetd, the eXtended InterNET Daemon, is an open-source daemon which runs on many Linux and Unix systems and manages Internet-

based connectivity. It offers a more secure extension to or version of inetd, the Internet daemon.

xinetd performs the same function as inetd: it starts programs that provide Internet services. Instead of having such servers started at system initialization time, and be dormant until a connection request arrives, xinetd is he only daemon process started and it listens on all service ports for the services listed in its configuration file. When a request comes in, xinetd starts the appropriate server. Because of the way it operates, xinetd (as well as inetd) is also referred to as a super-server.

Task: xinetd Configuration files location

Following are important configuration files for xinetd:

- /etc/xinetd.conf - The global xinetd configuration file.
- /etc/xinetd.d/ directory - The directory containing all service-specific files such as ftp

Task: Understanding default configuration file

You can view default configuration file with less or cat command:
less /etc/xinetd.confOR# cat /etc/xinetd.confOutput:

```
# Simple configuration file for xinetd
#
# Some defaults, and include /etc/xinetd.d/

defaults
{
        instances               = 60
        log_type                = SYSLOG authpriv
        log_on_success          = HOST PID
        log_on_failure          = HOST
        cps                     = 25 30
}

includedir /etc/xinetd.d
```

Where,

- **instances = 60 :** Determines the number of servers that can be simultaneously active for a service. So 60 is the maximum number of requests xinetd can handle at once.

- **log_type** = SYSLOG authpriv: Determines where the service log output is sent. You can send it to SYSLOG at the specified facility (authpriv will send log to /var/log/secure file).

- **log_on_success** = HOST PID: Force xinetd to log if the connection is successful. It will log HOST name and Process ID to /var/log/secure file.

- **log_on_failure = HOST**: Force xinetd to log if there is a connection dropped or if the connection is not allowed to /var/log/secure file

- **cps = 25 30**: Limits the rate of incoming connections. Takes two arguments. The first argument is the number of connections per second to handle. If the rate of incoming connections is higher than this, the service will be temporarily disabled. The second argument is the number of seconds to wait efore re-enabling the service after it has been disabled. The default for this setting is 50 incoming connections and the interval is 10 seconds. This is good to avoid DOS attack against your service.

- **includedir /etc/xinetd.d**: Read other service specific configuration file this directory.

Task: How to create my own service called foo

Here is sample config file for service called foo located at /etc/xinetd.d/foo

```
# vi /etc/xinetd.d/foo
```

And append following text:

```
service login
{
socket_type = stream
protocol = tcp
wait = no
user = root
server = /usr/sbin/foo
instances = 20
}
```

Where,

- **socket_type**: Sets the network socket type to stream.
- **protocol**: Sets the protocol type to TCP
- **wait**: You can set the value to yes or no only. It Defines whether the service is single-threaded (if set to **yes**) or multi-threaded (if set to **no**).
- **user**: User who will run foo server

Task: Stop or restart xinetd

To restart xinetd service type the command:

```
# /etc/init.d/xinetd restart
```

To stop xinetd service type the command:

```
# /etc/init.d/xinetd stop
```

To stop xinetd service type the command:

```
# /etc/init.d/xinetd start
```

Task: Verify that xinetd is running

Type the following command to verify xinetd service is running or NOT:

/etc/init.d/xinetd statusOutput:

xinetd (pid 6059) is running...

SAMBA

SAMBA – smb protocol – Server Msg Block aka netbios/tcp-ip

[c] Andrew Tridgell

– CIFS – ADS

History : 1. IBM PC DOS – M$ DOS – called MSDOS – PCBIOS 1985
2. PCBIOS ---> NetBEUI –
3. Soon they discarded NETBEUI bcos it did not support TCP/IP
4. PCBIOS ---> NetBEUI -> NETBIOS/TCP-IP aka NBNT
5. Changed name of NETBIOS/TCP-IP to SMB to CIFS to ADS
[Common Internet File system]

IPX/SPX – Novell – InterNW Pkt Exchange / Sequential Pkt Exchange

sawtimber
samba

Lastest : LongHorn – Whistler 2005 – Vista 2006

Uses of Samba:

1. As a File Server [File Sharing]/HW/SW Resource sharing like NFS but across OS'
2. As a WINS [Windows Internet Name Server] or NBNS server
3. As a PDC [SAM - Security Access Module]
4. As a Print Server using CUPS

Config File : /etc/samba/smb.conf

```
----Program 1-----------------------------------------

[funny]

# share or section or service

path = /opt/funny

# directive

-----------------------------------------------------------

$ service smb restart

$ testparm
```

Windows

```
$ net view

$ net view IP

$ net use * \\<ip>\funny

$ net use * /d

$ nbtstat -a IP

$ rpm -ql samba

     /etc/logrotate.d/samba - Log Rotation File

               --> Log Files: /var/log/samba

     /etc/pam.d/samba
```

Samba relies heavily upon authentication before providing access to the users such as file and printers.Its integrated with Samba Authentication, Accounting as well as session management is handled via the samba entries in pam.d

/etc/samba/smbusers

Provides a mapping between windows users and local linux based users Although samba can integrate in Windows World before providing access to local resources such as files on the file system, users must be authenticated by the local linux system. Default smbusers file contains mapping for local linux administrator which is root and root's equivalent user in windows world is administrator. However some call it as admin. If either of two users try to connect to share they should be equated to the local unix user root providing they know the password. The password should be same as of administrator and root user,otherwise it would prompt for the password, when they connect to the samba share.

If local windows users logs in through guest, pcguest and smbguest they are equated with "nobody" user on linux system

/etc/sysconfig/samba

Which specifies the parameter to run smbd and nmbd as a deamon

/var/spool/samba

If remote users attempt to spool print job to our samba server,they are generally spooled to this directory Samba nicely integrates with CUPS (defaut modular Printing System)

SMB services are provided by the NetBIOS protocol. NetBIOS makes its own name space available, which is completely different from the domain name system.

This name space can be accessed with the Unique Naming Convention (UNC) notation: all services provided by a server are addressed as \\Server\Servicename.

File or print services offered by a server are also called shares.

The server side of Samba consists of 2 parts:

- smbd. SMB/CIFS server

- This daemon provides file and print services for clients in the network.
- authentication and authorization
- File and printer sharing

- nmbd. NetBIOS name server

- This daemon handles all NetBIOS-related tasks.
- resource browsing
- WINS server

To integrate Linux as client in a Windows environment, Samba provides 2 tools:

- winbind. This daemon integrates a Linux system into a Windows authentication system (Active Directory).

- nmblookup. This tool can be used for NetBIOS name resolution and testing.

- smbclient. This tool provides access to SMB file and print services.

Samba version 3.0.22. An important new feature in this version is the Kerberos support in winbind. This allows a Kerberos based integration into Active Directory domains. Novell is an important contributor of the Samba project.

Packages installed in RHEL AS 4:

```
$ rpm -qa | grep samba

  samba-3.0.10-1.4E.2
```

- represents the files that are included into your system as Samba Server

samba-client-3.0.10-1.4E.2

- contains client based component

system-config-samba-1.2.21-1

- tool to configure Samba

samba-common-3.0.10-1.4E.2

contains shared components

```
$ rpm -ql   samba-common

/etc/samba/smb.conf
```

Main Configuration File

/etc/samba/lmhosts

The hostname "localhost" considered to be as netbios hostnamelmhosts is similar to lmhosts of windows world. In the event when you attempt to access windows based system or samba based system by name, translation can occur specially when you are using samba based clients with the aide of lmhosts file One of the options but not only the option Since samba based clients can also rely upon /etc/hosts as well as DNS and WINS (Centralize Name Repository)

/usr/bin/net

It can be used for joining samba system to a remote domain such as NT4 Style domain or windows 2000 style Active directory domain net command allows us to join those domains

/usr/bin/smbpasswd

allows us to equate password for locally stored users, so that when we attempt to authenticate remote users they are able to do so through

$ smbpasswd

$ /usr/bin/testparm

If you change manually to smb.conf file which resides in /etc/samba, testparm which check to insure that the parameters are correct

/var/log/samba

Log files pertaining to Samba Server

The Samba Configuration File

The /etc/samba/smb.conf file is the main configuration file you'll need to edit.

Three ways to approach this file:

- system-config-samba (Redhat's Tool)
- SWAT
- Manually

It is split into five major section

File Format - smb.conf

Section	Description
[global]	General Samba configuration parameters
[printers]	Used for configuring printersUsed for configuring printers
[homes]	Defines treatment of user logins
[netlogon]	A share for storing logon scripts. (Not created by default.)

[profile]	A share for storing domain logon information such as "favorites" and desktop icons. (Not created by default.)

smb.conf Minimum Settings, "Global" Section

Parameter	value	Description
domain logons	Yes	Tells Samba to become the PDC
preferred master	Yes	Makes the PDC act as the central store for the names of all windows clients, servers and printers on the network. Very helpful when you need to "browse" your local network for resources. Also known as a local master browser.
domain master	Yes	Tells Samba to become the master browser across multiple networks all over the domain. The local master browsers register themselves with the domain master to learn about resources on other networks.
os level	65	Sets the priority the Samba server should use when negotiating to become the PDC with other Windows servers. A value of 65 will usually make the Samba server win.
wins support	Yes	Allows the Samba server to provide name services for the network. In other words keeps track of the IP addresses of all the domain's servers and clients.
time server	Yes	Lets the samba server provide time updates for the domain's clients.
workgroup	"homenet"	The name of the

			Windows domain we'll create. The name you select is your choice. I've decided to use "homenet".
security	user		Make domain logins query the Samba password database located on the samba server itself.
smb passwd file	/etc/samba/smbpasswd		It is useful to specify the name and location of the Samba password file. This helps to make Samba version upgrades where the default locations may change.
private dir	/etc/samba		Specifies default directory for some supporting temporary files. As with the password file, it is a good practice to specify this value.

Security Levels

- user (user, server, ADS, domain)
- user

In user mode we can authenticate against local unix /etc/passwd file

- server

We pass authentication off

Server mode simply passes off authentication to the password authentication server such as NT4 or WIN2K domain controller

- Domain

This is used to join NT4 style domain. You need a computer account in NT4 style domain

- ADS

Same as Domain but just different behaviour

IF ADS is in native mode, they you will need to join the domain

And you need to know kerberos Realm, so that we can accept kerberos tickets

- share

This mode where you tie a password to share, and if a user knows the password they granted as readonly-readwrite share. This will be implemented where the everyone knows the password

Creating User

```
$ useradd champu

$ passwd champu

Assigning smb password to user champu

$ smbpasswd -a champu

editing configuration file

$ vi /etc/samba/smb.conf
 [myshare]
   comment = Windoze champu
   path = /home/champu
   valid users = champu
   public = yes
   writable = yes
   printable = no
   create mask = 0765
- public

This parameter is a synonym for guest ok.

$ service smb restart
```

gathering information:

```
$ smbclient -L 192.168.10.66 -U champu

The -L option should be used to determine if the Samba server is
even

running andlistening for network requests.
```

Accessing Samba Server

```
$ smbclient //192.168.10.66/myshare -U champu
smb: \> ls
smb: \> mkdir docs
smb: \> ls
smb: \> quit
```

Acessing Samba Server From Windows

```
Acessing Samba Server From X Windows
Open Nautilus
     in Location section
     smb://192.168.10.66
```

Permanent Mounting with Linux Samba Clients

```
$ mkdir /bill

$ vi /etc/samba/pass

username = champu

password = x

$ vi /etc/fstab
```

```
//192.168.10.60/myshare  /bill  smbfs  credentials=/etc/samba/pass
0  0

$ mount -a

$ df -h

$ cd /bill
```

Permanent Mounting Linux Samba Client Using Auto Mounter

```
$ vi /etc/auto.master

/misc     /etc/auto.smb  --timeout=60

$ vi /etc/auto.smb

samba  -fstype=smbfs,username=champu,password=x
//192.168.10.60/myshare

$ service autofs restart

$ cd /misc/samba

$ ls -l

$ vi /etc/samba/smb.conf
```

```
[myshare]
   hosts deny = 192.168.10.100
   comment = Windoze champu
   path = /home/champu
   valid users = champu
   public = yes
   writable = yes
   printable = no
   create mask = 0765
```

```
$ service smb restart
```

```
[hpcolor]
   comment = The HP 4500N
   path = /usr/spool/lpd/hpcolor
   browseable = yes
   printable = yes
   public = yes
   writable = yes
   create mode = 0700
```

Usage: smbmount

```
$ smbmount //192.168.10.60/myshare /bill -o
username=champu,password=x

$ cd /bill
```

There are basically two ways in which this can happen:

- The LMBs register themselves with a WINS server and thus are able

to determine that other LMBs serve the same workgroup.

• The workgroup is a “domain”: All systems in the domain make use of one Primary Domain Controller (PDC) for authentication. Such a PDC is required also to be the DMB. Since all systems know the IP address of the PDC, they also know which DMB to use.

Lbm hhists : static mapping

WINS server: dynamic mapping

Samba as a NT Domain Member

Samba emulates a NT workstation when becoming part of the domain. So, the first thing you need to do is create a machine account for your Samba machine on the domain controller. In NT you would use the program Server Manager for Domains to create the account. Once the account is created, all you need to add are the following lines to your smb.conf file under the global section.

- Your Workgroup or Domain that you want
- to login to workgroup = FREEOS
- Tell Samba to talk to domain controller
- for authentication

security = domain

- Specify the server to get authenticate from. You can specify the NetBIOS names of the servers or simply put in a “*” here to let Samba find the server through broadcast password server = PS1 PS2

Make sure Samba is using encrypted passwords

encrypt passwords = yes

Now stop the Samba daemons

$ /etc/rc.d/init.d/smb stop

Give the following command to join the NT Domain

$ smbpasswd -j DOMAIN -r DOMAINPDC

Samba Pdc

We will setup 1 domain "mydomain1" on a linux machine with samba.

1. Create a samba config files in /etc/samba/ and copy paste the content in 2nd step.
   a. smb.conf

2. Your smb.conf will look like below:

```
 [global]
 workgroup = mydomain1
            netbios name =server1
            time server = Yes
            domain logons = Yes
            os level = 65
            preferred master = Yes
            domain master = Yes
            encrypt passwords = yes
            smb passwd file = /etc/samba/smbpasswd
            security = user
            mangling method = hash
            add machine script = /usr/sbin/useradd -d /dev/null -g
trust -s /bin/false -M %u
            log file = /var/log/samba/log.%m
            log level = 3 passdb:5 auth:10 winbind:2
            logon path = \\%L\profiles\%U
            logon drive = H:
            logon home = \\%L\%U\.profile
            logon script = logon.cmd
            interfaces = 192.168.2.249/24  ##put your samba server
IP address
            bind interfaces only = yes
            lock directory = /var/lib/samba/locks/server1

    [homes]
            read only = No
            browseable = Yes
            create mask = 0644
            directory mask = 0755


    [netlogon]
            path=/var/lib/samba/netlogon
            guest ok = yes

    [profiles]
            path=/var/lib/samba/profiles
            browseable = yes
            read only = No
            create mask = 0600
            directory mask = 0700
            root preexec = PROFILE=/var/lib/samba/profiles/%u; if [
! -e $PROFILE ]; \
    then mkdir -pm700 $PROFILE; chown %u:%g $PROFILE;fi


    3. Then create below directory:
         /var/lib/samba/locks/server1

    4.  Start samba server:-
         /etc/init.d/smb start


    5. Check smb started or not.
         ps -ef|grep smb

    6. Add trust account (for NT machines only)
```

```
        groupadd trust
        useradd -g trust -d /dev/null -s /bin/false <machine name>\$
        passwd -l <machine name>\$
```

====> NOTE: PLEASE DONT FORGET TO GIVE '\$' IN ABOVE 2 COMMANDS

```
        smbpasswd -ma <machine name>
```

7. Adding administrator account

```
    smbpasswd -a root
    (GIVE Samba Passwd for root)
```

8. FOR WIN XP PROF users NOT for WIN98 ot XP HOME

login to that windows machine (machine name) with administrator.
Right click to "My Computer" and click on "Properties"
Click on "Computer Name" Tab
Click on "Change"
Put Domain - "mydomain1"
Click OK
It will ask for Domain admin username & passwd. Give username: root and smbpasswd of root
If everything is good then it will show you "Welcome to mydomain1"

SAMBA PDC WITH LINUX /WINDOWS CLIENT

SAMBA PDC
make dir: /var/lib/samba/locks/server
smb.conf

```
[global]
            workgroup = MYDOMAIN
            netbios name =server
         #   time server = Yes
            domain logons = Yes
            os level = 33
            preferred master = Yes
         # local master = Yes
            local master = no
            domain master = Yes
            encrypt passwords = yes
            smb passwd file = /etc/samba/smbpasswd
            security = user
            passdb backend = tdbsam
```

#when u use passdb backend = tdbsam ;when samba user is created, it stores the user name and passwd in passwd.tdb file rather than smbpasswd file

```
  mangling method = hash
  add machine script = /usr/sbin/useradd -d /dev/null -g trust -s
/bin/false -M %u
```

```
#  add machine script = /usr/sbin/useradd -n -c "Workstation (%u)"
-M -d /nohome -s /bin/false "%u"

            log file = /var/log/samba/log.%m
            log level = 3 passdb:5 auth:10 winbind:2

#for login windows machines
            logon path = \\%L\profiles\%U
            logon drive = H:
            logon home = \\%L\%U\.profile
            logon script = %m.bat
            logon script = %U.bat

##put your samba server IP address : eth0(optional)
            interfaces = eth0 192.168.1.0/24
            bind interfaces only = yes
            lock directory = /var/lib/samba/locks/server

    [homes]
            read only = No
            browseable = Yes
            create mask = 0644
            directory mask = 0755
            valid users = %S
            valid users = MYDOMAIN\%S

##

    [netlogon]
            path=/var/lib/samba/netlogon
            guest ok = yes

    [profiles]
            path=/var/lib/samba/profiles
            browseable = yes
            read only = No
            create mask = 0600
            directory mask = 0700
            root preexec = PROFILE = /var/lib/samba/profiles/%u; if
[ !-e $PROFILE ]; then mkdir -pm700 $PROFILE; chown %u:%g
$PROFILE;fi
```

Client side configuration

```
1>.smb.conf

[global]
netbios name = station
   workgroup = MYDOMAIN
   security = domain
   password server = 192.168.1.2
#   realm = AVTECH.LOCAL
encrypt passwords = yes
idmap uid = 16777216-33554431
idmap gid = 16777216-33554431
#idmap backend = ad
template homedir = /home/%D/%U
```

```
   template shell = /bin/bash
   winbind use default domain = true
~
~
```

2>.system-config-authentication
only check the winbind setting and specify the domain name n other req parameters
it wil do the req. winbind changes in the following files:
a>/etc/nsswitch
b>/etc/pam.d/system-auth

/etc/pam.d/system-auth
session required pam_mkhomedir.so   skel=/etc/skel umask=0077
insert this line at the bottom of the system-auth file just before
session   required   pam_unix.so
3.>Try to join the domain using foll
net rpc join domain-name -U root

also we can join using
system-config-authentication
gui authentication window opens
 select domain join
enter administrator name ie root and passwd
click join

FIREWALL / IPTABLES

What's a firewall?

Without getting into technical explanations, a firewall is simply a host whose main purpose is to protect your network. A firewall restricts certain types of network traffic from the Internet to your protected network(s) – the reverse is also often true.

Firewalls have always referred to as Layer 3 Security because from many years. Firewalls were only be able to understand information as it passed over the network layer 3. Now the technology has moved forward, Firewalls have become more flexible, more intelligent ultimately earning the name such as Layer 7 firewall or application level firewalls.

So the term firewall is not necessarily as accurate, Today as it is always been because there are various levels of firewalls.

Lets peek into concepts of firewall, how they might apply to security practices in linux environment

First we gonna look at IPTABLES, defacto standard for securing linux at the firewall level. So it pretty much comes bundled with everything across the board and it is pretty easy to configure and offers a lot of flexibility.

What a firewall is not?

- Magic – A firewall cannot make your network absolutely secure.
- A bastion host – In an ideal world, this would be true. However, a firewall is only as secure as the work you put into securing it.
- A bastion host is a special purpose computer on a network specifically designed and configured to withstand attack
- A replacement for host security – Every service you allow through the firewall is a potential risk.

Types of exploits

- Local – There is no security without physical security. If someone has physical access to your box, you've lost. Obviously, a firewall won't help you here.

- Local privilege escalation – The trojan horse attack. The attacker already has a local account on your box (inside the gates) and obtains root by some means (vulnerability or misconfiguration). A firewall cannot protect again this type of attacks.

- Remote – Your host is listening on a port that the attacker is able to connect to remotely over a network and exploit a vulnerability somehow. This is the only type of attack a firewall

can (hopefully) protect you against. There is another important point here that most firewall howtos neglect. In order for someone to exploit your box remotely, it has to be listening on some ports (i.e. providing a way for an attacker to connect). Therefore, if your host isn't listening on any ports, you are safe from remote exploits (unless the attacker manages to attack the network stack itself).

Why do you need a firewall?

* Increase your network security - Some services are inherently insecure and impossible to secure on individual hosts. A firewall can help you segment and contain parts of your network to increase security.

* Network access control - A firewall can help you enforce your network security policies by selectively allowing network services (to all or selected hosts).

* Logging - Because a firewall must examine all inbound/outbound network traffic, it can help you log network activity (that passes through the firewall).

So What's A Packet Filter?

A packet filter is a piece of software which looks at the *header* of packets as they pass through, and decides the fate of the entire packet. It might decide to DROP the packet (i.e., discard the packet as if it had never received it), ACCEPT the packet (i.e., let the packet go through), or something more complicated.

Under Linux, packet filtering is built into the kernel (as a kernel module, or built right in), and there are a few trickier things we can do with packets, but the general principle of looking at the headers and deciding the fate of the packet is still there.

IPTABLES

- replaces older IPchains firewall in linux
- available since 2.4 kernel
- Allows configuration of built-in firewall rules for host-based protection

IPtables can be used for routing, forwarding, filtering

- ipfwadm for linux kernel 2.0
- ipchains for linux kernel 2.2
- iptables for linux kernel 2.4/2.6

What is Netfilter/Iptables?

Netfilter is the framework in Linux kernels that allow for firewalling, NAT, and packet mangling.

Iptables is the userspace tools that works with the Netfilter framework (technically a lie; Iptables is also a part of the Netfilter framework in the kernel). Think of Netfilter as kernel space, and Iptables as userspace.

- Iptables is merely user space tool provides the administrator, means of configuring the core netfilter services that actually part of the kernel

IPtables is Statefull packet filtering firewall

We can monitor the states of communication process and make decisions based on that Defintely a useful feature. In the past, people were able to bypass firewall rules by skipping the beginning part of TCP communication process.

A stateful firewall (any firewall that performs stateful packet inspection (SPI) or stateful inspection) is a firewall that keeps track of the state of network connections (such as TCP streams, UDP communication) travelling across it. The firewall is programmed to distinguish legitimate packets for different types of connections. Only packets matching a known connection state will be allowed by the firewall; others will be rejected.

Before the advent of stateful firewalls, a stateless firewall, a firewall that treats each network frame (or packet) in isolation, was normal. Such a firewall has no way of knowing if any given packet is part of an existing connection, is trying to establish a new connection, or is just a rogue packet. Modern firewalls are connection-aware (or state-aware), affording network administrators finer-grained control of network traffic.

- can filter based upon source

IPAddress, protocol, port and connection state connection state which defines iptables as a statefull packet filtering firewall

- Can filter based upon MAC Addresss

This is obviously used very less. But in some of DMZ environment where you have control set of MACs this might be a little bit easier or makes more sense to use this feature

- Can filter out malformed packets based upon TCP Flags set in

packets

So we know that particular machine will never be seeing a christmas tree packet where pretty much everything is turned on, so we can set out filters to protect against these type of attacks

Packet Processing In iptables

All packets inspected by iptables pass through a sequence of built-in tables (queues) for processing. Each of these queues is dedicated to a particular type of packet activity and is controlled by an associated packet transformation/filtering chain.

There are three tables in total. The first is the mangle table which is responsible for the alteration of quality of service bits in the TCP header. This is hardly used in a home or SOHO environment.

The second table is the filter queue which is responsible for packet filtering. It has three built-in chains in which you can place your firewall policy rules. These are the:

- Forward chain: Filters packets to servers protected by the firewall.

- Input chain: Filters packets destined for the firewall.

- Output chain: Filters packets originating from the firewall.

The third table is the nat queue which is responsible for network address translation. It has two built-in chains; these are:

- Pre-routing chain: NATs packets when the destination address of the packet needs to be changed.

- Post-routing chain: NATs packets when the source address of the packet needs to be changed

Queue Type	**Queue Function**	**Packet Transformation Chain in Queue**	**Chain Function**
Filter	Packet filtering	FORWARD	Filters packets to servers accessible by another NIC on the firewall.
		INPUT	Filters packets destined to the firewall.
		OUTPUT	Filters packets originating from the firewall

Nat	Network Address Translation	PREROUTING	Address translation occurs before routing. Facilitates the transformation of the destination IP address to be compatible with the firewall's routing table. Used with NAT of the destination IP address, also known as **destination NAT** or **DNAT**.
		POSTROUTING	Address translation occurs after routing. This implies that there was no need to modify the destination IP address of the packet as in pre-routing. Used with NAT of the source IP address using either one-to-one or many-to-one NAT. This is known as **source NAT**, or **SNAT**.
		OUTPUT	Network address translation for packets generated by the firewall. (Rarely used in SOHO environments)
Mangle	TCP header modification	PREROUTING POSTROUTING OUTPUT INPUT FORWARD	Modification of the TCP packet quality of service bits before routing occurs. (Rarely used in SOHO environments)

You need to specify the table and the chain for each firewall rule you create. There is an exception: Most rules are related to filtering, so iptables assumes that any chain that's defined without an associated table will be a part of the filter table. The filter table is therefore the default.

To help make this clearer, take a look at the way packets are handled by iptables. In Figure 14.1 a TCP packet from the Internet arrives at the firewall's interface on Network A to create a data connection.

The packet is first examined by your rules in the mangle table's PREROUTING chain, if any. It is then inspected by the rules in the nat table's PREROUTING chain to see whether the packet requires DNAT. It is then routed.

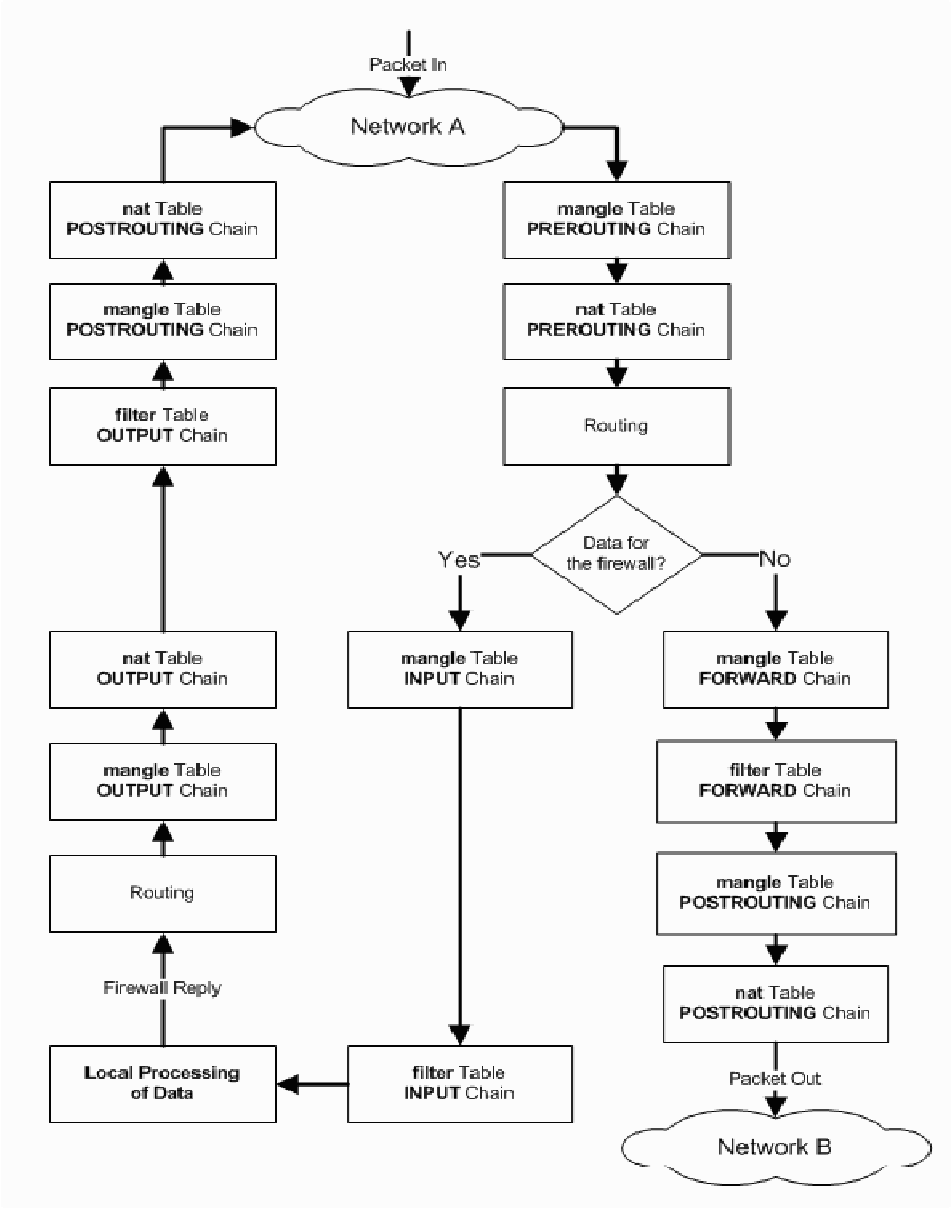
If the packet is destined for a protected network, then it is filtered by the rules in the FORWARD chain of the filter table and, if necessary, the packet undergoes SNAT in the POSTROUTING chain before arriving at Network B. When the destination server decides to reply, the packet undergoes the same sequence of steps. Both the FORWARD and POSTROUTING chains may be configured to implement quality of service (QoS) features in their mangle tables, but this is not usually done in SOHO environments.

If the packet is destined for the firewall itself, then it passes through the mangle table of the INPUT chain, if configured, before being filtered by the rules in the INPUT chain of the filter table before. If it

successfully passes these tests then it is processed by the intended application on the firewall.

At some point, the firewall needs to reply. This reply is routed and inspected by the rules in the OUTPUT chain of the mangle table, if any. Next, the rules in the OUTPUT chain of the nat table determine whether DNAT is required and the rules in the OUTPUT chain of the filter table are then inspected to help restrict unauthorized packets. Finally, before the packet is sent back to the Internet, SNAT and QoS mangling is done by the POSTROUTING chain

Packet Flow As Follows



Targets And Jumps

Each firewall rule inspects each IP packet and then tries to identify it as the target of some sort of operation. Once a target is identified, the packet needs to jump over to it for further processing. Table 14.2 lists the built-in targets that iptables uses.

Table 14-2 Descriptions Of The Most Commonly Used Targets

Target	Desciption	Most Common Options
ACCEPT	• iptables stops further processing. • The packet is handed over to the	N/A

		end application or the operating system for processing	
DROP		• iptables stops further processing. • The packet is blocked	N/A
LOG		• The packet information is sent to the syslog daemon for logging • iptables continues processing with the next rule in the table • As you can't log and drop at the same time, it is common to have two similar rules in sequence. The first will log the packet, the second will drop it.	--log-prefix "string" Tells iptables to prefix all log messages with a user defined string. Frequently used to tell why the logged packet was dropped
REJECT		• Works like the DROP target, but will also return an error message to the host sending the packet that the packet was blocked	--reject-with qualifier The qualifier tells what type of reject message is returned. Qualifiers include: icmp-port-unreachable (default) icmp-net-unreachable icmp-host-unreachable icmp-proto-unreachable icmp-net-prohibited icmp-host-prohibited tcp-reset echo-reply
DNAT		• Used to do **destination network address translation**. ie. rewriting the	--to-destination ipaddress Tells iptables what the

	destination IP address of the packet	destination IP address should be
SNAT	• Used to do **source network address translation** rewriting the source IP address of the packet • The source IP address is user defined	--to-source <address>[-<address>][:<port>-<port>] Specifies the source IP address and ports to be used by SNAT.
MASQUERADE	• Used to do Source Network Address Translation. • By default the source IP address is the same as that used by the firewall's interface	[--to-ports <port>[-<port>]] Specifies the range of source ports to which the original source port can be mapped.

Important Iptables Command Switch Operations

Each line of an iptables script not only has a jump, but they also have a number of command line options that are used to append rules to chains that match your defined packet characteristics, such the source IP address and TCP port. There are also options that can be used to just clear a chain so you can start all over again. Tables 14.2 through 14.6 list the most common options.

Table 14-2 General Iptables Match Criteria

iptables command Switch	**Desciption**
-t <-table->	If you don't specify a table, then the filter table is assumed. As discussed before, the possible built-in tables include: filter, nat, mangle
-j <target>	Jump to the specified target chain when the packet matches the current rule.
-A	Append rule to end of a chain
-F	Flush. Deletes all the rules in the selected table
-p <protocol-type>	Match protocol. Types include, icmp, tcp, udp, and all
-s <ip-address>	Match source IP address
-d <ip-address>	Match destination IP address

-i <interface-name>	Match "input" interface on which the packet enters.
-o <interface-name>	Match "output" interface on which the packet exits

In this command switches example

iptables -A INPUT -s 0/0 -i eth0 -d 192.168.1.1 -p TCP -j ACCEPT

iptables is being configured to allow the firewall to accept TCP packets coming in on interface eth0 from any IP address destined for the firewall's IP address of 192.168.1.1. The 0/0 representation of an IP address means any.

Table 14-4 Common TCP and UDP Match Criteria

Switch	Desciption
-p tcp --sport <port>	TCP source port. Can be a single value or a range in the format: *start-port-number:end-port-number*
-p tcp --dport <port>	TCP destination port. Can be a single value or a range in the format: *starting-port:ending-port*
-p tcp --syn	Used to identify a new TCP connection request. ! --syn means, not a new connection request
-p udp --sport <port>	UDP source port. Can be a single value or a range in the format: *starting-port:ending-port*
-p udp --dport <port>	UDP destination port. Can be a single value or a range in the format: *starting-port:ending-port*

Checking whether Iptables is default installed on our server

$ rpm -q iptables

$ rpm -ql iptables

- /sbin/iptables

this allows us to view the configuration and change it

- /sbin/iptables-restore

this allows us to restore the firewall or running firewall configuration from a saved configuration

- /sbin/iptables-save

this allows the save the running configuration

- Modules are stored in /lib/iptables/

- Iptables itself is a module

$ lsmod | grep -i iptab

Modules for Iptables:

- ip_tables
- iptable_filter
- iptable_nat
- iptable_mangle

$ lsmod | grep -i ipt

- ipt_REJECT - target (fate)
- ipt_state - allows to maintain state information
- ip_conntrack - kernel can keep track of connections

SYNTAX

$ iptables -t table <Action> <Direction/Chains> <Packet Pattern> -j <fate>

iptables Won't Start

The iptables startup script expects to find the /etc/sysconfig/iptables before it starts. If none exists, then symptoms include the firewall status always being stopped and the /etc/init.d/iptables script running without the typical [OK] or [FAILED] messages.

If you have just installed iptables and have never applied a policy, then you will face this problem. Unfortunately, running the service iptables save command before restarting won't help either. You have to create this file.

[root@bigboy tmp]# service iptables start

[root@bigboy tmp]#

[root@bigboy tmp]# touch /etc/sysconfig/iptables

[root@bigboy tmp]# chmod 600 /etc/sysconfig/iptables

[root@bigboy tmp]# service iptables start

Applying iptables firewall rules: [OK]

[root@bigboy tmp]#

Linux Iptables allow or block ICMP ping request

The Internet Control Message Protocol (ICMP) has many messages that are identified by a "type" field. You need to use 0 and 8 ICMP code types.

=> **Zero** (0) is for echo-reply

=> **Eight** (8) is for echo-request.

To enable ICMP ping incoming client request use following iptables rule (you need to add following rules to script).

My default firewall policy is blocking everything.

Task: Enable or allow ICMP ping incoming client request

Rule to enable ICMP ping incoming client request (assuming that default iptables policy is to drop all INPUT and OUTPUT packets)

```
SERVER_IP="202.54.10.20"

iptables -A INPUT -p icmp --icmp-type 8 -s 0/0 -d $SERVER_IP -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT

iptables -A OUTPUT -p icmp --icmp-type 0 -s $SERVER_IP -d 0/0 -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Task: Allow or enable outgoing ping request

To enable ICMP ping outgoing request use following iptables rule:

```
SERVER_IP="202.54.10.20"

iptables -A OUTPUT -p icmp --icmp-type 8 -s $SERVER_IP -d 0/0 -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT

iptables -A INPUT -p icmp --icmp-type 0 -s 0/0 -d $SERVER_IP -m state --state ESTABLISHED,RELATED -j ACCEPT
```

How do I disable outgoing ICMP request?

Use the following rules:

```
iptables -A OUTPUT -p icmp --icmp-type echo-request -j DROP
```

OR

```
iptables -A OUTPUT -p icmp --icmp-type 8 -j DROP
```

ICMP echo-request type will be block by above rule.

See ICMP TYPE NUMBERS (type fields). You can also get list of ICMP types, just type following command at shell prompt:

```
$ /sbin/iptables -p icmp -h

$ iptables -t filter -P INPUT DROP

$ iptables -t filter -P OUTPUT ACCEPT

$ iptables -t filter -P FORWARD ACCEPT


$ allow local loopback connections

$ iptables -t filter -A INPUT -i lo -j ACCEPT
```

Drop INVALID connections

```
$ iptables -t filter -A INPUT -m state --state INVALID -j DROP

$ iptables -t filter -A OUTPUT -m state --state INVALID -j DROP

$ iptables -t filter -A FORWARD -m state --state INVALID -j DROP
```

Allow all established and related

```
$ iptables -t filter -A INPUT -m state --state ESTABLISHED,RELATED
-j ACCEPT

$ iptables -t filter -A OUTPUT -m state --state ESTABLISHED,RELATED
-j ACCEPT

$ iptables -t filter -A FORWARD -m state --state
ESTABLISHED,RELATED -j ACCEPT
```

Allow connections to my ISP's DNS servers

```
iptables -t filter -A INPUT -s 213.73.255.52 -p tcp -m tcp ! --tcp-
flags SYN,RST,ACK SYN -j ACCEPT

iptables -t filter -A INPUT -s 213.73.255.52 -p udp -j ACCEPT

iptables -t filter -A INPUT -s 213.132.189.250 -p tcp -m tcp ! --
tcp-flags SYN,RST,ACK SYN -j ACCEPT

$ iptables -t filter -A INPUT -s 213.132.189.250 -p udp -j ACCEPT

$ iptables -t filter -A INPUT -s 213.73.255.53 -p tcp -m tcp ! --
tcp-flags SYN,RST,ACK SYN -j ACCEPT

$ iptables -t filter -A INPUT -s 213.73.255.53 -p udp -j ACCEPT
```

open ports 4662,4672 = amule, 5900,5901 = vnc, 22 = ssh

```
$ iptables -t filter -A INPUT -p tcp -m tcp --dport 4662 -j ACCEPT

$ iptables -t filter -A INPUT -p udp -m udp --dport 4672 -j ACCEPT

$ iptables -t filter -A INPUT -p tcp -m tcp --dport 5900 -j ACCEPT

$ iptables -t filter -A INPUT -p tcp -m tcp --dport 5901 -j ACCEPT

$ iptables -t filter -A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
```

bittorrent :

```
$ iptables -t filter -A INPUT -p tcp -m tcp --dport 6881:6889 -j
ACCEPT
```

samba (only connections from lan are accepted)

```
$ iptables -t filter -A INPUT -o eth0 -s 192.168.0.0/255.255.255.0
```

```
-p tcp -m tcp --dport 137:139 -j ACCEPT

$ iptables -t filter -A INPUT -o eth0 -s 192.168.0.0/255.255.255.0
-p udp -m udp --dport 137:139 -j ACCEPT

log all other attempted in going connections

$ iptables -t filter -A INPUT -o eth0 -j LOG
```

NAT

set up IP forwarding and nat

```
$ iptables -t nat -P POSTROUTING ACCEPT

$ iptables -t nat -P PREROUTING ACCEPT

# 6891:6900 = msn filetransfers

# 192.168.0.1 = gateway

# 192.168.0.216 = client in network

$ iptables -t nat -A PREROUTING -i eth1 -p tcp -m tcp --dport
6891:6900 -j DNAT --to-destination 192.168.0.216:6891-6900

$ iptables -t nat -A PREROUTING -i eth1 -p udp -m udp --dport
6891:6900 -j DNAT --to-destination 192.168.0.216:6891-6900

$ iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

Policy

```
$ IPTABLES -P INPUT ACCEPT

$ IPTABLES -P OUTPUT ACCEPT

$ IPTABLES -P FORWARD ACCEPT
```

Tables

```
$ IPTABLES -N tcp_packets

$ IPTABLES -N icmp_packets

$ IPTABLES -N udpincoming_packets
```

IP Masquerade

```
$ IPTABLES -t nat -A POSTROUTING -o $INET_IFACE -j MASQUERADE
```

Squid transparent proxy

```
$ IPTABLES -A PREROUTING -t nat -i eth0 -p tcp --dport 80 -j
REDIRECT --to-port 3128

$ IPTABLES -A PREROUTING -t nat -i eth2 -p tcp --dport 80 -j
REDIRECT --to-port 3128
```

```
$ IPTABLES -A PREROUTING -t nat -i ppp0 -p tcp --dport 80 -j
REDIRECT --to-port 3128
```

smtp

```
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 25 -j ACCEPT
```

www

```
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 80 -j ACCEPT
```

https

```
$IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 443 -j ACCEPT
```

mail

```
$ IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 465 -j ACCEPT

$ IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 993 -j ACCEPT

$ IPTABLES -A tcp_packets -p TCP -s 0/0 --dport 995 -j ACCEPT
```

wlan vpn

```
$ IPTABLES -A wlan_packets -p UDP -s 0/0 --dport 5000 -j ACCEPT

$ IPTABLES -A wlan_packets -p ALL -j DROP
```

iptables blocking with mac address
Drop all connection coming from mac address 00:0F:EA:91:04:08 (add command to your firewall script)

```
$ iptables -A INPUT -m mac --mac-source 00:0F:EA:91:04:08 -j DROP
```

iptables allowing with mac address
Allow port 22 for mac address 00:0F:EA:91:04:07

```
$ iptables -A INPUT -p tcp --destination-port 22 -m mac --mac-
source
00:0F:EA:91:04:07 -j ACCEPT
```

General syntax:

```
$ iptables RULE -m time --timestart TIME --timestop TIME --days
DAYS -j ACTION
```

Where,

- --timestart TIME : Time start value . Format is 00:00-23:59 (24 hours format)
- --timestop TIME : Time stop value.
- --days DAYS : Match only if today is one of the given days. (format: Mon,Tue,Wed,Thu,Fri,Sat,Sun ; default everyday)

An example
Suppose you would like to allow incoming ssh access only available from Monday to Friday between 9 AM to 6. Then you need to use iptables as follows:
Input rule:

```
$ iptables -A INPUT -p tcp -s 0/0 --sport 513:65535 -d 202.54.1.20
--dport 22 -m state --state NEW,ESTABLISHED -m time --timestart
09:00 --timestop 18:00 --days Mon,Tue,Wed,Thu,Fri -j ACCEPT
```

Output rule:

```
$ iptables -A OUTPUT -p tcp -s 202.54.1.20 --sport 22 -d 0/0 --
dport 513:655 -m state --state ESTABLISHED -m time --timestart
09:00 --timestop 18:00 --days Mon,Tue,Wed,Thu,Fri -j ACCEPT
```

Force SYN packets check

Make sure NEW incoming tcp connections are SYN packets; otherwise we need to drop them:

```
$ iptables -A INPUT -p tcp ! --syn -m state --state NEW -j DROP
```

Force Fragments packets check

Packets with incoming fragments drop them. This attack result into Linux server panic such data loss.

```
$ iptables -A INPUT -f -j DROP
```

XMAS packets

Incoming malformed XMAS packets drop them:

```
$ iptables -A INPUT -p tcp --tcp-flags ALL ALL -j DROP
```

Drop all NULL packets

Incoming malformed NULL packets:

```
$ iptables -A INPIT -p tcp --tcp-flags ALL NONE -j DROP
```

Protect against SYN floods by rate limiting the number of new connections from any host to 60 per second. This does *not* do rate limiting overall, because then someone could easily shut us down by saturating the limit.

```
$ iptables -A INPUT -m state --state NEW -p tcp -m tcp --syn \
    -m recent --name synflood --set
$ iptables -A INPUT -m state --state NEW -p tcp -m tcp --syn \
    -m recent --name synflood --update --seconds 1 --hitcount 60 -j
DROP
```

The same can be achieved in ipfw using the dummynet shaper:

Direct SYN

```
ipfw pipe 500 config bw 64Kbit/s queue 5

ipfw add 500 pipe 500 tcp from any to any in setup
```

Port scanning

A lot of hosts try to port scan my server these days, looking for open services they can try to exploit. Since I run very few services on my server, what I like to do is look for port connections to a commonly scanned port (port 139, for Windows File Sharing), and then block the hosts who attempt the connection from talking to my server for an entire day. The rule is quite simple using the iptables recent module:

Anyone who tried to portscan us is locked out for an entire day.

```
$ iptables -A INPUT   -m recent --name portscan --rcheck --seconds
```

```
86400 -j DROP

$ iptables -A FORWARD -m recent --name portscan --rcheck --seconds
86400 -j DROP
```

Once the day has passed, remove them from the portscan list

```
$ iptables -A INPUT   -m recent --name portscan --remove

$ iptables -A FORWARD -m recent --name portscan --remove
```

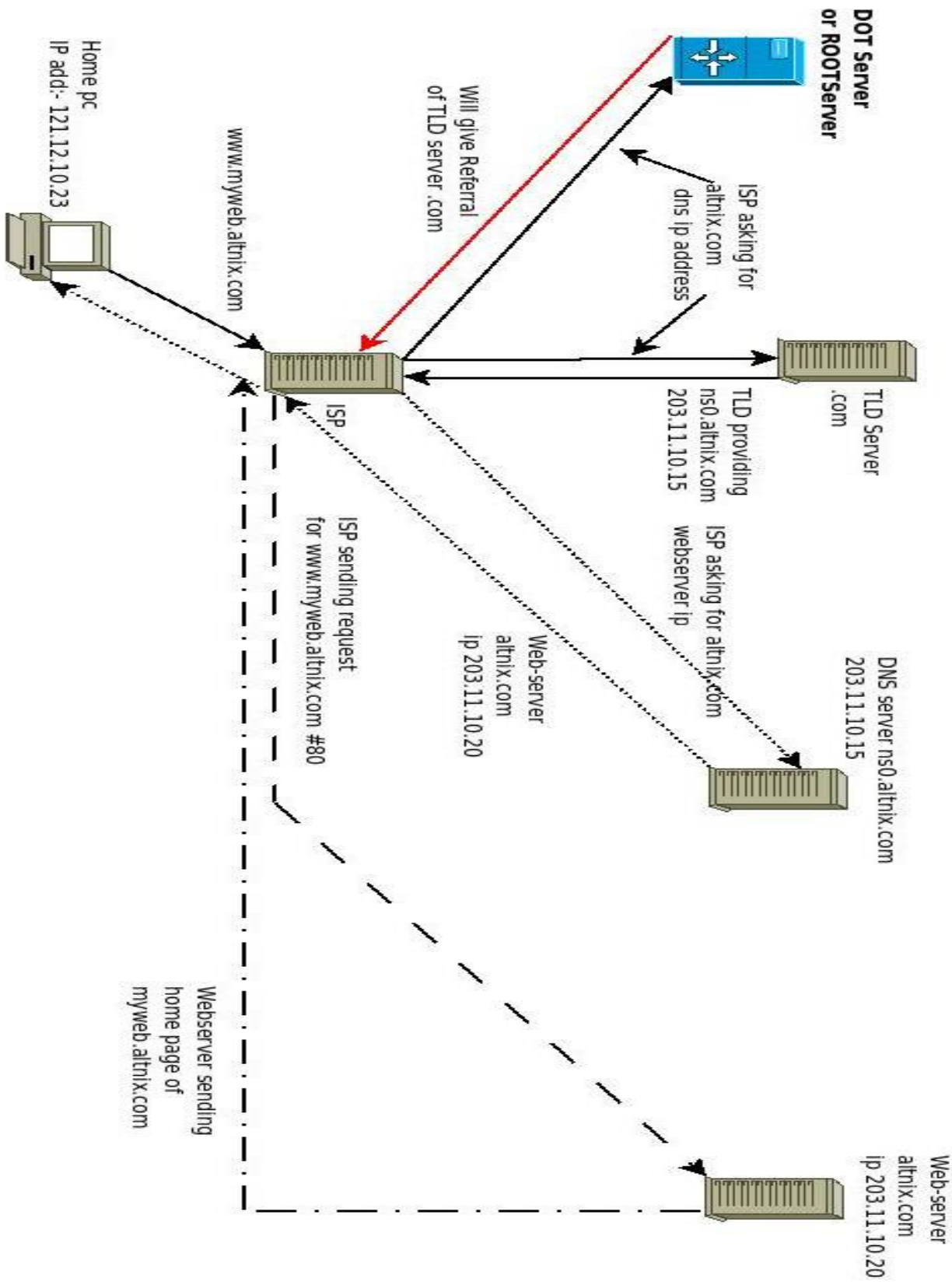
CLOSE INCOMING TCP

```
$ IPTABLES -A tcp_packets -m state --state ESTABLISHED,RELATED -j
ACCEPT

$ IPTABLES -A tcp_packets -p TCP -s 0/0 -j DROP
```

CLOSE INCOMING UDP

```
$    IPTABLES    -A    udpincoming_packets    -m    state    --state
ESTABLISHED,RELATED -j ACCEPT

$ IPTABLES -A udpincoming_packets -p UDP -j DROP
```

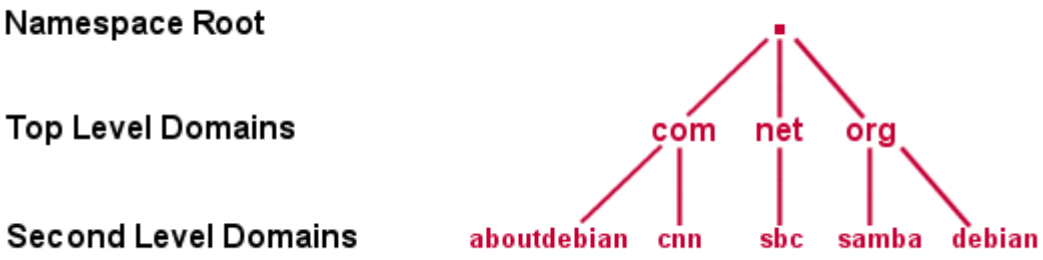
DNS



How DNS works

DNS Basics

Finding a single server out of all of the servers on the Internet is like trying to find a single file on drive with thousands of files. In both cases it helps to have some hierarchy built into the directory to logically group things. The DNS "namespace" is hierarchical in the same type of upside-down tree structure seen with file systems. Just as you have the root of a partition or drive, the DNS namespace has a root which is signified by a period.



When specifying the absolute path to a file in a file system you start at the root and go to the file:

/etc/bind/named.conf

When specifying the absolute path to a server in the DNS namespace you start at the server and go to the root:

www.aboutdebian.com.

Note that period after the 'com' as it's important. It's how you specify the root of the namespace. An absolute path in the DNS namespace is called a FQDN (Fully Qualified Domain Name). The use of FQDNs are prevalent in DNS configuration files and it's important that you always use that trailing period.

Internet resources are usually specified by a domain name *and* a server hostname. The www part of a URL is often the hostname of the Web server (or it could be an alias to a server with a different host name). DNS is basically just a database with records for these hostnames. The directory for the entire telephone system is not stored in one huge phone book. Rather, it is broken up into many pieces with each city having, and maintaining, its piece of the entire directory in its phone book. By the same token, pieces of the DNS directory database (the "zones") are stored, and maintained, on many different DNS servers located around the Internet. If you want to find the telephone number for a person in Poughkeepsie, you'd have to look in the Poughkeepsie telephone book. If you want to find the IP address of the www server in the some-domain.com domain, you'd have to query the DNS server that stores the DNS records for that domain.

The entries in the database map a host/domain name to an IP address. Here is a simplistic logical view of the type of information that is stored (we'll get to the A, CNAME, and MX designations in a bit).

A	**www.their-domain.com**	**172.29.183.103**
MX	**mail.their-domain.com**	**172.29.183.217**
A	**debian.your-domain.com**	**10.177.8.3**
CNAME	**www.your-domain.com**	**10.177.8.3**
MX	**debian.your-domain.com**	**10.177.8.3**

This is why a real Internet server needs a **static** (unchanging) IP address. The IP address of the server's NIC connected to the Internet has to match whatever address is in the DNS database. Dynamic DNS does provide a way around this for home servers however, which we'll see later.

When you want to browse to www.their-domain.com your DNS server (the one you specify in the TCP/IP configuration on your desktop computer) most likely won't have a DNS record for the their-domain.com domain so it has to contact the DNS server that does. When your DNS server contacts the DNS server that has the DNS records (referred to as "resource records" or "zone records") for their-domain.com your DNS server gets the IP address of the www server and relays that address back to your desktop computer. So which DNS server has the DNS records for a particular domain?

When you register a domain name with someone like Network Solutions, one of the things they ask you for are the server names and addresses of two or three "name servers" (DNS servers). These are the servers where the DNS records for your domain will be stored (and queried by the DNS servers of those browsing to your site). So where do you get the "name servers" information for your domain? Typically, when you host your Web site using a Web hosting service they not only provide a Web server for your domain's Web site files but they will also provide a DNS server to store your domain's DNS records. In other words, you'll want to know who your Web hosting provider is going to be before you register a domain name (so you can enter the provider's DNS server information in the name servers section of the domain name registration application).

> You'll see the term "zone" used in DNS references. Most of the time a zone just equates to a domain. The only times this wouldn't be true is if you set up subdomains *and* set up separate DNS servers to handle just those subdomains. For example, a company would set up the subdomains us.their-domain.com and europe.their-domain.com and would "delegate" a separate DNS server to each one of them. In the case of these two DNS servers their zone would be just the subdomains. The zone of the DNS server for the parent their-domain.com (which would contain the servers www.their-domain.com and mail.their-domain.com) would only contain records for those few machines in the parent domain.
>
> Note that in the above example "us" and "europe" are subdomains while "www" and "mail" are host names of servers in the parent domain.

Once you've got your Web site up and running on your Web hosting

provider's servers and someone surf's to your site, the DNS server they specified in their local TCP/IP configuration will query your hosting provider's DNS servers to get the IP address for your Web site. The DNS servers that host the DNS records for your domain, i.e. the DNS servers you specify in your domain name registration application, are the **authoritative** DNS servers for your domain. The surfer's DNS server queries one of your site's authoritative DNS servers to get an address and gets an authoritative response. When the surfer's DNS server relays the address information back to the surfer's local PC it is a "non-authoritaive" response because the surfer's DNS server is not an authoritative DNS server for your domain.

Domains and Delegation

The Domain Name System uses a tree (or hierarchical) name structure. At the top of the tree is the root node followed by the Top-Level Domains (TLDs), then the Second-Level Domains (SLD) and any number of lower levels, each separated with a dot.
s Note The root of the tree is represented most of the time as a silent dot (.), but there are times when it
is VERY important.

TLDs are split into two types:
1. Generic Top-Level Domains (gTLD): For example, .com, .edu, .net, .org, .mil, etc.
2. Country Code Top-Level Domains (ccTLD): For example, .us, .ca, .tv, .uk, etc. Country Code TLDs use a standard two-letter sequence defined by ISO 3166.2 Figure 1-1

DNS name resolution is nothing but resolving host names, such as www.nixcraft.com, to their corresponding IP addresses. DNS works as the "phone book" for the Internet by translating hostname into IP address or vise versa. Most DNS server stores following information:
a) **Hostname** and their **IP address**
b) List of **mail server and their IP address** for given domain name
c) **Anti spam configuration** and much more.

Without DNS name resolution, nothing will work on the Internet. Nobody likes to remember IP address, so DNS is foundation of many Internet services such as web, proxy, email and so on.

Resolving DNS names to IP addresses

When you type www.yahoo.com into a web browser, the application has to find out IP address associated with www.yahoo.com. Each part of network has DNS server or name servers. Each application send a request called dns lookup to DNS server. Each DNS server has limited information about host names and ip address. Almost all DNS server constantly query each other to get information using root servers.

Each computer is configured to query specific name server. Usually home computers are configure to query ISP name servers or free dns name servers. Here is a typical UNIX / Linux /etc/resolv.conf file with nameserver IP address:

```
$ cat /etc/resolv.conf
```

Sample output:
```
nameserver 208.67.222.222
nameserver 208.67.220.220
```

Each application can find www.yahoo.com IP address by sending a request to 208.67.222.222 or 208.67.220.220 IP address. This procedure is called **hostname resolution** and the algorithm that performs this operation is called the **resolver**. Let us see how to find out IP address for freebsd.nixcraft.in hostname:

1. The web browser will check local cache database to find out answer. If it can get an answer directly from these, it proceeds no further.
2. Otherwise request will be sent to nameserver IP 208.67.222.222 to find IP address for freebsd.nixcraft.in host.
3. 208.67.222.222 server will decide if that IP has been recently looked up before. If it has, there is no need to ask further, since the result would be stored in a local cache.
4. 208.67.222.222 will see if the domain is local. I.e. if it is a computer that it has direct information about. In this case this would only be true if the 208.67.222.222 were Obsidian's very own name server.
5. 208.67.222.222 will strip out the TLD (Top Level Domain) .in It will query a root name server, asking what name server is responsible for .IN. Depend upon the answer 208.67.222.222 will query authoritative server for IP address.
6. 208.67.222.222 will return the result to the application.
7. 208.67.222.222 will store each of these results in a local cache with an expiry date. To avoid having to look them up a second time.

Caching nameserver

To run a caching-only name server, the following files are required and must be created or copied to the appropriate directories on your server.

- Copy the named.conf file to the /etc/ directory.
- Copy the db.127.0.0 file to the /var/named/ directory.
- Copy the db.cache file to the /var/named/ directory.
- Copy the named script file to the /etc/rc.d/init.d/ directory.

To run a master name server, the following files are required and must be created or copied to the appropriate directories on your server.

Copy the named.conf file to the /etc/ directory.
Copy the db.127.0.0 file to the /var/named/ directory.
Copy the db.cache file to the /var/named/ directory.
Copy the db.208.164.186 file to the /var/named/ directory.
Copy the db.altnix file to the /var/named/ directory.
Copy the named script file to the /etc/rc.d/init.d/ directory.

To run a slave name server, the following files are required and must be created or copied to the appropriate directories on your

server.

- Copy the named.conf file to the /etc/ directory.
- Copy the db.127.0.0 file to the /var/named/ directory.
- Copy the db.cache file to the /var/named/ directory.
- Copy the named script file to the /etc/rc.d/init.d/ directory.

Linux DNS and BIND Server

Caching-only name Server

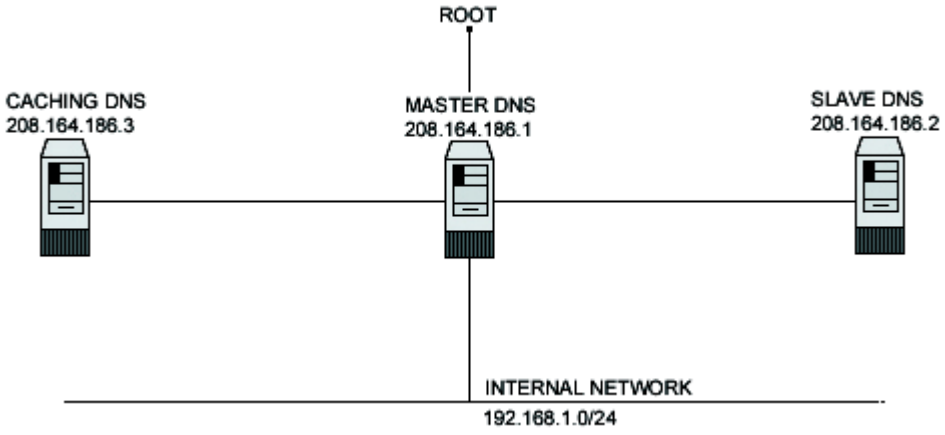
Setting up a caching server for client local machines will reduce the load on the site's primary server. A caching only name server will find the answer to name queries and remember the answer the next time we need it. This will shorten the waiting time the next time significantly. For security reasons, it is very important that DNS doesn't exist between hosts on the corporate network and external hosts; it is far safer to simply use IP addresses to connect to external machines from the corporate network and vice-versa.

In our configuration and installation we'll run BIND/DNS as non root-user and in a chrooted environment. We also provide you three different configurations;

- one for a simple caching name server only *client*
- one for a slave *secondary server*
- one for a master name server *primary server*.

The simple caching name server configuration will be used for your servers that don't act as a master or slave name server, and the slave and master configurations will be used for your servers that act as a master name server and slave name server. Usually one of your servers acts as master, another one acts as slave and the rest act as simple caching client name server.

This is a graphical representation of the DNS configuration we use in this book. We try to show you different settings



Caching Only DNS

Master DNS

Slave DNS

on different servers. A lot of possibilities exist, an

Caching-only name servers are servers not authoritative for any

domains except 0.0.127.in-addr.arpa, the localhost. A caching-only name server can look up names inside and outside your zone, as can primary and slave name servers. The difference is that when a caching-only name server initially looks up a name within your zone, it ends up asking one of the primary or slave names servers for your zone for the answer.

The necessary files to setup a simple caching name server are:

- named.conf
- db.127.0.0
- db.cache
- named script

To configure the /etc/named.conf file for a simple caching name server, use this for all servers that dont act as a master or slave name server. Setting up a simple caching server for local client machines will reduce the load on the network's primary server. Many users on dialup connections may use this configuration along with bind for such a purpose. Create the named.conf file, **touch** /etc/named.conf and add the following lines to the file:

```
options {
directory "/var/named";
forwarders { 208.164.186.1; 208.164.186.2; };❶
forward only;
};
//
// a caching only nameserver config
zone "." in {
type hint;
file "db.cache";
};

zone "0.0.127.in-addr.arpa" in {
type master;
file "db.127.0.0";
};
```

❶ In the forwarders line, 208.164.186.1 and 208.164.186.2 are the IP addresses of your Primary *Master* and Secondary *Slave* DNS server. They can also be the IP addresses of your ISPs DNS server and another DNS server, respectively.

TIP: To improve the security of your BIND/DNS server you can stop it from even trying to contact an off-site server if their forwarder is down or doesn't respond. With the forward only option set in your named.conf file, the name server doesn't try to contact

other servers to find out information if the forwarder doesn't give it an answer.

To configure the /var/named/db.127.0.0 file for a simple caching name server,you can use this configuration for all machines on your network that don't act as a master or slave name server. The db.127.0.0 file covers the loopback network. Create the following files in /var/named/, **touch** /var/named/db.127.0.0 and add the following lines in the file:

```
$TTL 345600

@       IN      SOA     localhost.
root.localhost.  (

00      ; Serial

86400   ; Refresh

7200    ; Retry

2592000 ; Expire

345600 ); Minimum

IN      NS      localhost.

1        IN      PTR     localhost.
```

Configure the /var/named/db.cache file for a simple caching name server before starting your DNS server. You must take a copy of db.cache file and copy this file to the /var/named/ directory. The db.cache tells your server where the servers for the root zone are.

Use the following commands on another Unix computer in your organization to query a new db.cache file for your DNS Server or pick one from your Red Hat Linux CD-ROM source distribution:

```
[root@deep]# dig @.aroot-servers.net . ns >
db.cache
```

Primary master name Server

A primary master name server for a zone reads the data for the zone from a file on it's host and are authoritative for that zone.The necessary files to setup a primary master name server are:

named.conf
db.127.0.0
db.208.164.186
db.altnix
db.cache
named script

To configure the /etc/named.conf file for a master name server, use this configuration for the server on your network that acts as a master name server. After compiling DNS, you need to set up a primary domain name for your server. We'll use altnix.com as an example domain, and assume you are using IP network address of 208.164.186.0. To do this, add the following lines to your

/etc/named.conf. Create the named.conf file **touch** /etc/named.conf and add:

```
options {
directory "/var/named";
fetch-glue no;                                    ❶
recursion no;                                     ❷
allow-query { 208.164.186/24; 127.0.0/8; };       ❸
allow-transfer { 208.164.186.2; };                ❹
transfer-format many-answers;
};

// These files are not specific to any zone
zone "." in {
type hint;
file "db.cache";
};
zone "0.0.127.in-addr.arpa" in {
type master;
file "db.127.0.0";
};

// These are our primary zone files
zone "altnix.com" in {
type master;
file "db.altnix ";
};

zone "186.164.208.in-addr.arpa" in {
type master;
file "db.208.164.186";
};
```

❶❷ The fetch-glue no option can be used in conjunction with the option recursion no to prevent the server's cache from growing or becoming corrupted. Also, disabling recursion puts your name servers into a passive mode, telling it never to send queries on behalf of other name servers or resolvers. A non-recursive name server is very difficult to spoof, since it doesn't send queries, and hence doesn't cache any data.

❸ In the allow-query line, 208.164.186/24 and 127.0.0/8 are the IP addresses allowed to ask ordinary questions to the server.

❹ In the allow-transfer line, 208.164.186.2 is the IP address allowed to receive zone transfers from the server. You must ensure that only your real slave name servers can transfer zones from your name serve, as the information provided is often used by spammers and IP spoofers.

NOTE: The options recursion no, allow-query, and allow-transfer in the named.conf file above are security features.

To configure the /var/named/db.127.0.0 file for a master and slave name server, you can use this configuration file by both a master name server and a slave name server. The db.127.0.0 file covers the loopback network. Create the following files in /var/named/.

Create the db.127.0.0 file, **touch** /var/named/db.127.0.0 and add:

```
 ; Revision History: April 22, 1999 -
admin@mail.altnix.com

 ; Start of Authority (SOA) records.

 $TTL 345600

 @  IN  SOA  deep.altnix.com.
admin.mail.altnix.com.  (

 00      ; Serial

 86400   ; Refresh

 7200    ; Retry

 2592000 ; Expire

 345600 ); Minimum


 ; Name Server (NS) records.
```

```
        NS   deep.altnix.com.
        NS   mail.altnix.com.

        ; only One PTR record.
        1      PTR  localhost.
```

To configure the /var/named/db.208.164.186 file for a master name server, Use this configuration for the server on your network that acts as a master name server. The file db.208.164.186 maps host names to addresses. Create the following files in /var/named/.

Create the db.208.164.186 file, **touch** /var/named/db.208.164.186 and add:

```
         ; Revision History: April 22, 1999 -
        admin@mail.altnix.com
         ; Start of Authority (SOA) records.
         $TTL 345600
         @  IN  SOA  deep.altnix.com.
        admin.mail.altnix.com.  (
         00      ; Serial
         86400   ; Refresh
         7200    ; Retry
         2592000 ; Expire
```

```
345600 ); Minimum

; Name Server (NS) records.

NS   deep.altnix.com.

NS   mail.altnix.com.

; Addresses Point to Canonical Names (PTR) for
Reverse lookups

1  PTR      deep.altnix.com.

2  PTR      mail.altnix.com.

3  PTR      www.altnix.com.
```

To configure of the /var/named/db.altnix file for a master name server, use this configuration for the server on your network that acts as a master name server. The file db.altnix maps addresses to host names. Create the following file in /var/named/.

Create the db.altnix file **touch** /var/named/db.altnix and add:

```
; Revision History: April 22, 1999 -
admin@mail.altnix.com

; Start of Authority (SOA) records.

$TTL 345600

@  IN  SOA  deep.altnix.com.
admin.mail.altnix.com.  (

00      ; Serial

86400   ; Refresh

7200    ; Retry

2592000 ; Expire

345600 ); Minimum

; Name Server (NS) records.

NS   deep.altnix.com.

NS   mail.altnix.com.

; Mail Exchange (MX) records.

MX   0  mail.altnix.com.

; Address (A) records.

localhost     A       127.0.0.1

deep    A       208.164.186.1
```

```
mail    A       208.164.186.2

www     A       208.164.186.3

; Aliases in Canonical Name (CNAME) records.

;www                             CNAME
deep.altnix.com.
```

To configure the /var/named/db.cache file for a master and slave name servers Before starting your DNS server you must take a copy of the db.cache file and copy it into the /var/named/ directory. The db.cache tells your server where the servers for the root zone are.

Use the following command on another Unix computer in your organization to query a new db.cache file for your DNS Server or pick one from your Red Hat Linux CD-ROM source distribution:

```
[root@deep] /# dig @.aroot-servers.net . ns >
db.cache
```

Don't forget to copy the db.cache file to the /var/named/ directory on your server where you're installing DNS server after retrieving it over the Internet.

Dig

http://www.madboa.com/geek/dig/

NS Trace
http://www.dollardns.net/cgi-bin/nstrace/index.pl?

Dns Crawler
http://www.dollardns.net/cgi-bin/dnscrawler/index.pl?

who is
http://whois.dollardns.net/domain.pl?

http://www.dollardns.net/index.html?http://www.dollardns.net/compare.html

Apache

What is Apache?

- Apache is an open-source web server (HTTP server)

- Apache servers up web pages (HTML) and most any other content that can be accessed via a web browser.

- It runs on Linux and Windows

- Apache is the number one web server on the internet today and has been since 1996.

- Infact, 70% of all Internet websites run on Apache webserver

Lets talk about the history of apache web server

It was originally developed at National Center for Supercomputing Applications at the university of illinois by Robert McCool in around 1991. But it quickly became the public domain.

Robert McCool left NCSA in 1995 and httpd web server was not maintained officially, so loosely organised group of developes around the world came together to exchange patches, updates and fixes.

- Apache was created around 1995 by a group of webmasters who got together to create patches for the old httpd server in Unix.

- Today, that group is called the Apache Software Foundation.

- The homepage for Apache is http://httpd.apache.org

- The source and compiled versions of Apache are all free

- Why Apache

- high Performance
- Open Source
- Free
- Unrestrictive license
- can be modified and redistribute with another name
i.e Covalent Apache, IBM http server
- Runs on linux/unix/windows

- Apache HTTP Server

version 1.3

+ single thread process model

Version 2.0

+ multi-thread support (SMP)
+ support for non-unix platforms using MPM modules
  - MPM --> Multi-Processing Modules
+ supports IPV6
+ new API

- meant easier for writing 3rd party modules

Differnce between iis & Apache

Apache -First, Apache doesn't install a lot of extra programs. A default Apache build doesn't install any Apache modules (extensions) at all -- just a basic webserver

IIS- By default, Windows 2000 and IIS install seven external Dynamic Link Library (DLL) files plus FrontPage server extensions

Apache- Apache components, if their installed, run as a nonprivileged user, so if a buffer overflow occurs, damage is minimal. Conversely, Microsoft IIS allows system-level access, thereby potentially granting root (superuser) permission. Any user, even a remote one, who has root permission can access, change, and delete any file anywhere on the system.

IIS - If the Internet Information Server (IIS) process dies on a Windows Web server, no further requests are served until the process is restarted

Apache - If a single Apache process dies, only the request being served by that process is affected.

This approach has an obvious advantage over Web servers that use a single process to respond to all requests: If the Internet Information Server (IIS) process dies on a Windows Web server, no further requests are served until the process is restarted. If a single Apache process dies, only the request being served by that process is affected

Installing Apache

- RedHat Linux or Fedora Linux installs Apache web server, by default. You can ofcourse, choose not to install it at installation time if you wish.

- Apache is, however, disabled by default. In other words, it isnt running.

- RHEL or fedora does not, however, install the 31 optional related applications that go along with Apache. You can install these from the Package Manager application.

Administering Apache

- The Apache web server has its own configuration directory, /etc/httpd/

- Inside this directory, there are subdirectories and soft links to other directories.

- The actual configuration file for Apache is at /etc/httpd/conf/ and it is called httpd.conf

- All log files go to /var/log/httpd

- The httpd.conf can be edited manually.

- Apache can be started manually by typing "httpd" at the command line but the recommended way to start it is using the Services program.

- The "DocumentRoot" directory (/var/www/htm/) is where the default website is located.

- Apache can also deliver user's webpages. This is defined with the "UserDir" directive and users directories can be acccessed, through

  Apache like this:

http://www.altnix.com/~sadhiq

- Apache can also support Virtual hosting with the "VirtualHost" directive

- web server servers many different websites

- Apache cannot be run from a Super Server like XinetD.

- To administer apache, GUI tool is available: system-config-httpd

SPECS

Package: httpd

Version: 2.2

Conf file: /etc/httpd/conf/httpd.conf

DocumentRoot for storing web pages: /var/www/html

Apache Modules: /etc/httpd/modules/

/usr/lib/httpd/modules/

Commands:

1. Syntax Check
   $ apachectl configtest
   OR
   $ httpd -t
2. check compiled in modules
   $ httpd -l

Service:

$ service httpd start
OR
$ apachectl start

Four main directories:

- /etc/httpd/conf/

- /etc/httpd/conf.d/

- /etc/httpd/logs/

  --> sym link to /var/log/httpd/

- /etc/httpd/modules

Packages included in RHEL:

```
$ rpm -qa | grep httpd
        httpd-manual-2.0.52-19.ent
        httpd-2.0.52-19.ent
        system-config-httpd-1.3.1-1
        httpd-suexec-2.0.52-19.ent
```

Configuration Files

/etc/httpd/conf/httpd.conf - Main Apache Configuration File
/
/etc/logrotate.d/httpd -- Logrotation File
/
/etc/httpd/conf.d
- Contains items that have included for different type of **application**
- Files under /etc/httpd/conf.d
  perl.conf
  php.conf (CGI scripting language)
  python.conf (CGI scripting language)
  ssl.conf (how ssl to be implemented)
  webalizer.conf (analysis software)
  welcome.conf (in the event attempt to access the URL which has no default document)

Quickway: webserver configuration

```
$ cd /var/www/html
$ vi index.html
----------------------------
       <html>
       <head>
       <title>
       Quickway: webserver configuration
       </title>
       <body>
       Quickway: webserver configuration: IP
        </body>
        </html>
 ----------------------------

$ service httpd restart

$ links http://<IP_of_the_webserver>
```

Dedicated webserver configuration: altnix.com

```
$ cp /etc/httpd/conf/httpd.conf /etc/httpd/conf/httpd.conf_ORIG

$ vi /etc/httpd/conf/httpd.conf

  ---------------------------------------------

  ServerRoot "/etc/httpd" #default

  Listen 80 #default

  ServerName www.altnix.com:80

  ServerAdmin john@altnix.com
```

```
  User apache #default
  Group apache #default
  # set folder for the webpages
        DocumentRoot "/var/www/html"
  # set the name of the file that is first read
        DirectoryIndex index.html
           apachectl configtest
       OR
       $ httpd -t
```

Start the http service

```
       $ service httpd restart
       OR
       $ apachectl restart
```

Confirm the httpd daemon running on port 80

```
       $ netstat -antp | grep :80
```

Test from the client machine

```
              linux> links http://www.altnix.com
```

make httpd start on bootup

```
       $ chkconfig --level 35 httpd on
```

What is Virtual Hosting

Virtual Hosting is the ability host multiple separate web sites with one Apache Server

Each site is separate from each other, with different DocumentRoot, log files, permissions, etc

Two types of Virtual Hosting

1. IP based virtual Hosts
   - Each IP corresponds to its own individual website
   - IP based VH is where each virtual host has its own IP address
    + single server
          --> One Apache Daemon, handling multiple websites
     + multiple server
        --> Two or more independent Apache daemons, each one handling a specific website
2. Name based virtual hosts
   - Name based virtual hosts is used for hosting multiple websites on the same webserver IP address.

Name Based Virtual Hosting

Scenario:

+ champu.local
+ funny.local
  - both the websites running on the same ip address
  - in my case: 192.168.10.111

Note: Make sure the DNS 'A' record of champu.local and funny.local should resolve to 192.168.10.111

```
$ mkdir /var/www/html/champu.local

$ cd /var/www/html/champu.local

$ vi index.html
 ---------------------------------------------------------------
        <html>
        <head>
        <title>
            MY FIRST HTML PAGE: CHAMPU: NAME BASED
        </title>
        <body>
 MY FIRST HTML PAGE: CHAMPU: NAME BASED
         </body>
        </html>
$ mkdir /var/www/html/funny.local
$ cd /var/www/html/funny.local
$ vi index.html
---------------------------------------------------------------
        <html>
        <head>
        <title>
             MY FIRST HTML PAGE: FUNNY: NAME BASED
        </title>
        <body>
              MY FIRST HTML PAGE: FUNNY: NAME BASED
        <
        </body>
        </html>
-----------------------------------------------

$ vi /etc/httpd/conf/httpd.conf

-----------------------------------------------

  NameVirtualHost 192.168.10.111:80

  <VirtualHost 192.168.10.111:80>

        ServerAdmin webmaster@champu.local

        DocumentRoot /var/www/html/champu.local/

        ServerName champu.local

        ErrorLog logs/champu.local_error_log

        CustomLog logs/champu.local_access_log common

  </VirtualHost>

  <VirtualHost 192.168.10.111:80>
```

```
        ServerAdmin webmaster@funny.local
        DocumentRoot /var/www/html/funny.local
        ServerName funny.local
        ErrorLog logs/funny.local_error_log
        CustomLog logs/funny.local_access_log common
  </VirtualHost>
  ------------------------------------------------



$ httpd -t
$ service httpd restart
IP Based Virtual Hosting: Single Server Configuration
Note: Make sure the DNS 'A' record of champu.local should resolve
to 192.168.10.11
        and funny.local to 192.168.10.222
Scenario:
        + champu.local
                Resolves to 192.168.10.111
        + funny.local
                R
                Resolves to 192.168.10.222


1. Create IP alias
        # cd /etc/sysconfig/network-scripts/
        # cp ifcfg-eth0 ifcfg-eth0:0
        # vi ifcfg-eth0:0
                DEVICE=eth0:0
                BOOTPROTO=static
                BROADCAST=192.168.10.255
                IPADDR=192.168.10.222
                NETMASK=255.255.255.0
                NETWORK=192.168.10.0
                O
```

```
ONBOOT=yes
```

2. Make the Alias IP up

```
# ifup eth0:0
```

3. Edit httpd.conf file

```
# vi /etc/httpd/conf/httpd.conf
-----------------------------------------------
NameVirtualHost 192.168.10.111:80
<VirtualHost 192.168.10.111:80>
      ServerAdmin webmaster@champu.local
      DocumentRoot /var/www/html/champu.local/
      ServerName champu.local
      ErrorLog logs/champu.local_error_log
      CustomLog logs/champu.local_access_log common
</VirtualHost>
NameVirtualHost 192.168.10.222:80
<VirtualHost 192.168.10.222:80>
      ServerAdmin webmaster@funny.local
      DocumentRoot /var/www/html/funny.local
      ServerName funny.local
      ErrorLog logs/funny.local_error_log
      CustomLog logs/funny.local_access_log common
</VirtualHost>

$ cd /var/www/html/champu.local
$ vi index.html
-------------------------------------------------------------
      <html>
      <head>
      <title>
```

```
            MY FIRST HTML PAGE: CHAMPU: IP BASED: 192.168.10.111
        </title>
          <body>
MY FIRST HTML PAGE: CHAMPU: IP BASED: 192.168.10.111


        </body>
        </html>
$ cd /var/www/html/funny.local
$ vi index.html
 ------------------------------------------------------------
        <html>
        <head>
        <title>
 MY FIRST HTML PAGE: FUNNY: IP BASED: 192.168.10.222
        </title>
        <body>
 MY FIRST HTML PAGE: FUNNY: IP BASED: 192.168.10.222
        <
        </body>
        </html>
$ service httpd restart
================================================
```

IP Based Virtual Hosting: Multiple Server Configuration Scenario

- champu.local
  - Resolves to 192.168.10.111 on port 80
- funny.local
  - Resolves to 192.168.10.222 on port 8080

==========

1. vi /etc/httpd/conf/httpd.conf

```
  ---------------------------------------------
  Listen 192.168.10.111:80
  Listen 192.168.10.222:8080
  NameVirtualHost 192.168.10.111:80
  <VirtualHost 192.168.10.111:80>
       ServerAdmin webmaster@champu.local
       DocumentRoot /var/www/html/champu.local/
www.alnix.com
      ServerName champu.local
      ErrorLog logs/champu.local_error_log
      CustomLog logs/champu.local_access_log common
  </VirtualHost>
 NameVirtualHost 192.168.10.222:8080
  <VirtualHost 192.168.10.222:8080>
      ServerAdmin webmaster@funny.local
      DocumentRoot /var/www/html/funny.local
      ServerName funny.local
      ErrorLog logs/funny.local_error_log
      CustomLog logs/funny.local_access_log common
  </VirtualHost>
$ cd /var/www/html/champu.local
$ vi index.html
        ------------------------------------------------------
----
        <html>
        <head>
        <title>

MY FIRST HTML PAGE: CHAMPU: IP BASED: 192.168.10.111, PORT: 80
        </title>
```

```
        <body>
           MY FIRST HTML PAGE: CHAMPU: IP BASED: 192.168.10.111,
PORT: 80
         </body>
        </html>
```

4. # cd /var/www/html/funny.local

5. # vi index.html

```
        ------------------------------------------------------------
----
        <html>
        <head>
        <title>
           MY FIRST HTML PAGE: FUNNY: IP BASED: 192.168.10.222,
PORT: 8080
        </title>
        <body>
           MY FIRST HTML PAGE: FUNNY: IP BASED: 192.168.10.222,
PORT: 8080
         </body>
         </html>
```

Test from the client machine

```
       linux$> links http://www.champu.local
       linux$> links http://www.funny.local:8080
```

1st WAY: Setup password protection – inside httpd.conf

www.alnix.com

$ mkdir /var/www/html/champu.local/noaccess

$ vi index.html

```
------------------------------------------------------------------
        <html>
        <head>
        <title>
```

```
            CHAMPU: RESTRICTED ACCESS PAGE
        </title>
        <body>
            CHAMPU: RESTRICTED ACCESS PAGE
        </body>
        </html>
$ vi /etc/httpd/conf/httpd.conf
-------------------------------------------------
   <VirtualHost 192.168.10.111:80>
       ServerAdmin webmaster@champu.local
       DocumentRoot /var/www/html/champu.local/
       ServerName champu.local
        <Directory /var/www/html/champu.local/noaccess>
                AuthName "Restricted Site"
                AuthType Basic
                AuthUserFile /var/www/html/champu.local/.passwords
                require valid-user
        </Directory>
       ErrorLog logs/champu.local_error_log
       CustomLog logs/champu.local_access_log common
   </VirtualHost>
  ------------------------------------------------
Notes:
        Basic - Standard username/password combination.
        Digest - MD5 encrypted username/password combinations.
$ htpasswd -c /var/www/html/champu.local/.passwords champu
--> Give access to user john also
--> adds the user "john" to the password file
                /var/www/html/champu.local/.passwords
$ htpasswd -m /var/www/html/champu.local/.passwords john
Test
        $ links http://www.champu.local/noaccess
```

--> Prompts for username/passwd

Troubleshooting Apache

Checking the Logs

If there is something wrong with your Apache, but you have no idea how to figure out what's wrong,

your first clues will be in the log files.

There are a few log files around. All of them are located inside /var/log/httpd/

access_log

67.185.0.236 - - [18/Jun/2005:12:05:50 -0700] "GET / HTTP/1.0" 200 721

10.0.1.80 - - [18/Jun/2005:12:11:07 -0700] "GET /~jaspenelle/__journal1.jpg HTTP/1.1" 200 19079

66.239.233.163 - - [18/Jun/2005:12:15:06 -0700] "GET /~jaspenelle/avy14.gif HTTP/1.0" 200 1661

67.185.60.155 - - [18/Jun/2005:12:18:48 -0700] "GET / HTTP/1.0" 200 721

67.185.0.236 - - [18/Jun/2005:12:25:39 -0700] "GET / HTTP/1.0" 200 721

10.0.1.80 - - [18/Jun/2005:12:28:04 -0700] "GET /~jaspenelle/avy14.gif HTTP/1.1" 200 1661

10.0.1.80 - - [18/Jun/2005:12:28:46 -0700] "GET /~jaspenelle/avy7.png HTTP/1.1" 200 13066

This file is simply a listing of every file requested from your server. Unless you have changed the default configuration, it will be in Common Log Format:

Common Log Format syntax

remotehost rfc931 authuser [date] "request" status bytes

www.altnix.com

remotehost	Remote host name or IP address
rfc931	The remote log name of the user.
authuser	The user name as which the user has authenticated himself.
[date]	Date and time of the request.
"request"	The request line exactly as it came from the client.
status	The HTTP status code returned to the client.

bytes The content-length of the document transferred.

error_log

[Mon Feb 07 23:33:18 2005] [notice] suEXEC mechanism enabled (wrapper: /usr/sbin/suexec2)

[Mon Feb 07 23:33:18 2005] [notice] Digest: generating secret for digest authentication ...

[Mon Feb 07 23:33:18 2005] [notice] Digest: done

[Mon Feb 07 23:33:18 2005] [notice] Apache/2.0.52 (Gentoo/Linux) PHP/4.3.10 configured -- resuming normal

operations

[Sat Jun 18 13:01:54 2005] [error] [client 10.0.1.80] File does not

exist: /var/www/localhost/htdocs/favicon.ico

[Sat Jun 18 13:02:14 2005] [error] [client 10.0.1.80] File does not

exist: /var/www/localhost/htdocs/favicon.ico

[Sat Jun 18 13:02:18 2005] [error] [client 10.0.1.80] File does not

exist: /var/www/localhost/htdocs/favicon.ico

[Sat Jun 18 13:02:21 2005] [error] [client 10.0.1.80] File does not

exist: /var/www/localhost/htdocs/favicon.ico

[Sat Jun 18 13:02:24 2005] [error] [client 10.0.1.80] File does not

exist: /var/www/localhost/htdocs/favicon.ico

As you can see, this file can contain a lot of stuff, depending on the ErrorLevel directive in your

httpd.conf file. It tells you if apache started up correctly, what errors it has run into, ... In general it

will tell you what went wrong. If something isn't working right, this should be the first file you check

for more information.

Tips and Tricks of my trade

Restart Apache Server without affecting existing connections

Sometimes you want to restart your Apache server after changing some configuration in your virutual hosts,sites etc, but you have few hundred clients currently downloading files from your server and you don’t want to disconnect them.

You need to use the following command

```
$ service httpd graceful
```

This will gracefully restart your Apache with new configuration without affecting your client’s connections.

Performance tunning

The Apache HTTP Server is a modular program where the administrator can choose the functions to be included in the server by selecting a set of modules [2]. The modules can be compiled either statically as part of the 'httpd' binary, or as Dynamic Shared Objects (DSOs). DSO modules can either be compiled when the server is built, or added later via the apxs utility, which allows compilation at a later date. The mod_so module must be statically compiled into the Apache core to enable DSO support.

Run Apache with only the required modules. This reduces the memory footprint, which improves the server performance. Statically compiling modules will save RAM that's used for supporting dynamically loaded modules, but you would have to recompile Apache to add or remove a module. This is where the DSO mechanism comes handy. Once the mod_so module is statically compiled, any other module can be added or dropped using the 'LoadModule' command in the 'httpd.conf' file. Of course, you will have to compile the modules using 'apxs' if they weren't compiled when the server was built.

Choose appropriate MPM:

The Apache server ships with a selection of Multi-Processing Modules (MPMs) which are responsible for binding to network ports on the machine, accepting requests, and dispatching children to handle the requests [3]. Only one MPM can be loaded into the server at any time.

Choosing an MPM depends on various factors, such as whether the OS supports threads, how much memory is available, scalability versus stability, whether non-thread-safe third-party modules are used, etc.

Linux systems can choose to use a threaded MPM like worker or a non-threaded MPM like prefork:

The *worker* MPM uses multiple child processes. It's multi-threaded within each child, and each thread handles a single connection. Worker is fast and highly scalable and the memory footprint is comparatively low. It's well suited for multiple processors. On the other hand, worker is less tolerant of faulty modules, and a faulty thread can affect all the threads in a child process.

The *prefork* MPM uses multiple child processes, each child handles one connection at a time. Prefork is well suited for single or double CPU systems, speed is comparable to that of worker, and it's highly tolerant of faulty modules and crashing children - but the memory usage is high, and more traffic leads to greater memory usage.

Multi Thread
a thread is a process-within-a-process. Multiple threads reside within a single process. Threading has several advantages:

- Resources (memory, etc.) can be shared between threads.

- Multiple threads can execute simultaneously.

Apache 1.3's case, the lack of multiple threads means that a separate process must be used to respond to each incoming request. This approach has an obvious advantage over Web servers that use a single process to respond to all requests: If the Internet Information Server (IIS) process dies on a Windows Web server, no further requests are served until the process is restarted. If a single Apache process dies, only the request being served by that process is affected

The administrator must ensure that enough processes are available to handle incoming requests without forking new ones, but not so many that the system hits resource limits. Several directives in the Apache configuration file accomplish this:

- The **MaxClients** setting limits the number of Apache processes that will be created. Typically, memory is the limitation on this setting. If your Apache process takes up 20 MB of memory, and you have 1000MB of free RAM, you could have up to 50 Apache processes (1000MB/20MB = 50).

- The **MinSpareServers** and **MaxSpareServers** settings keep a number of processes waiting around, to avoid the delay imposed by forking a new process. New processes are forked continually to keep the number of available servers between these thresholds, but incoming HTTP requests do not have to wait for processes to be forked because spares are available.

To account for differences between platforms, while retaining the reliability of multiple processes, Apache 2.0 provides several different models for controlling Apache processes and threads in the form of Multi-Processing Modules (MPMs):

- The **prefork MPM** replicates the single-threaded behavior of Apache 1.3. This is the default MPM for UNIX systems.

- The **worker MPM** "implements a hybrid multithreaded multi-process Web server." Several processes are started, each with a fixed number of threads. Processes are started or stopped as necessary to regulate the total number of threads.

- The **perchild MPM** regulates the total number of threads by varying the number of threads in each process. This MPM also allows Apache processes to operate as multiple user IDs, which can be useful for managing several virtual hosts.

http://httpd.apache.org/docs/2.0/mpm.html

http://httpd.apache.org/docs/2.0/mod/worker.html

http://httpd.apache.org/docs/2.0/misc/perf-tuning.html

http://httpd.apache.org/docs/2.2/mod/prefork.html

http://books.google.co.in/books?id=cnDuw7GV4uYC&pg=PA180&lpg=PA180&dq=Difference+between++worker+MPM+%26+prefork+MPM&source=web&ots=4hKq5VQwf&sig=HocOBWL7lUwRrWjup1cp7sbf4eI&hl=en&sa=X&oi=book_result&resnum=5&ct=result#PPA186,M1

http://tldp.org/LDP/LGNET/123/vishnu.html#MPM

http://www.howtoforge.com/configuring_apache_for_maximum_performance

keepalive and keepalivetimeout :-

KeepAlive:-

This directive is taking "on"/"off" as parameter. In simple term - whether you want to use the feature or not. For example, once you visit a site (www.someting.com), there would be a number of connection from your machine to the remote machine (on port 80). Once the browse finished fetcing pages, the socket will be closed (if KeepAlive off). If you click on a link on that page, another connection will be initiated. Remember that opening/closing socket will require some overhead from OS, and Apache itself (same thing with closing the sockets).

KeepAliveTimeout:-

KeepAliveTimeout will determin howlog a persistent connection will be kept open.

The number of seconds Apache will wait for a subsequent request before closing the connection. Once a request has been received, the timeout value specified by the Timeout directive applies.Setting KeepAliveTimeout to a high value may cause performance problems in heavily loaded servers. The higher the timeout, the more server processes will be kept occupied waiting on connections with idle clients.

Apache monitoring

wtop search for it

is a tool for benchmarking your Apache HTTP server.

apachectl

is a front end to the Apache HTTP server which is designed to help the administrator control the functioning of the Apache httpd

daemon.

apxs

is a tool for building and installing extension modules for the Apache HTTP server.

dbmanage

is used to create and update the DBM format files used to store usernames and passwords for basic authentication of HTTP users.

htdigest

is used to create and update the flat-files used to store usernames, realms and passwords for digest authentication of HTTP users.

htpasswd

is used to create and update the flat-files used to store usernames and passwords for basic authentication of HTTP users.

httpd

is the Apache HTTP server program.

instdso.sh

is a script which installs Apache DSO modules.

logresolve

is a post-processing program to resolve IP-addresses in Apache's access log files.

rotatelogs

is a simple program for use in conjunction with Apache's piped log file feature.

SendMail

Sendmail queries the database version of the file such as berkeley DB

- /etc/mail/acesss.db

Berkeley DB should be installed to support sendmail to read the DB files

file /etc/mail/access*

rpm -qa | grep -i db

Package Version/Name: db4-4.2.

- /etc/mail/helpfile

SMTP commands

e.g. telnet localhost 25

HELP

- /etc/mail/local-host-names

To know how to handle the domains which are considered to be local

So it handles routing for

- localhost
- localhost.domain
- FQDN, in my case - postfix.altnix.com
- 192.168.10.30

- 127.0.0.1

Default MTA accepts messages for all the above

- /var/spool/mail/

- contains mailbox per user

e.g. /var/spool/mail/~username

- Traditional Unix Mbox

Sendmail uses macrotize language called m4, As sendmail configuration is so complex its abstracted to macro utility using m4

- /etc/mail/sendmail.mc

You can make changes and configure in sendmail.mc

Sendmail.mc is much easier to understand

- /etc/mail/sendmail.cf

Sendmails main configuration file

- /etc/mail/sendmail.mc

dnl --> way of commenting

Sendmail is separated into two daemons:

$ ps -ax | grep -i sendmail

- Once accepts connection

e.g. sendmail: accepting connections on port 25

- Other runs the queue

e.g. sendmail: Queue runner@01:00:00 for /var/spool/clientmqueue

01:00 --> 1 minute

mails get stored in /var/spool/clientmqueue, queue runner daemon wakes up every 1 minute

/var/spool/clientmqueue own by smmsp

smsp --> sendmail mail submission program

So users in our localsystem invokes the mail in local queu

/var/spool/clientmqueue, which gets scanned every 1 minute by

the queue runner

- /etc/mail/trusted-users

users that can send mail as others without a warning Able to rewrite from section without sendmail complaining

- /etc/mail/virtusertable

Allows us to setup virtual domains

e.g.

champu@postfix.altnix.com champu (local account)

So we have given you a brief introduction to the default implementation of sendmail

within Redhat framework

generate a message and sendmail deliver the mail

to the user champu

```
# which mutt
```

Mutt is a great client and default it reads mbox format but also have the ability to interact with Maildirs which is newer and more robust way of storing mail messages.

However there is any environment variable set which mutt relies upon

```
$ set | grep -i mutt
MAIL=/var/spool/mail/root
```

This environment variable should point to proper users mail box

As root you can read anyone's mailbox

```
$ which sendmail
$ ls -l /usr/sbin/sendmail
$ ls -l /etc/alternatives/mta
```

SENDMAIL is monolithic. It handles all messaging binding to the MTA port

as well as local delivery

```
$ ps -ef | grep sendmail
```

You could see various instances (process) but all are tied with same binary

By default sendmail accepts mails from local user and deliver it to local and remote user

Configure mail server to accept internet email

1. Allowing Sendmail to accept mails from network

```
$ vi  /etc/mail/sendmail.mc
```

search for 127.0, put dnl at the front of the line

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

2. Convert Macro based file

```
$ cd /etc/mail
$ make
```

OR

```
$ cd /etc/mail
$ m4 sendmail.mc > sendmail.cf
```

3. Restart Sendmail Service

```
$ service sendmail restart
```

4.

```
$ su - champu
$ mail tree
Subject:
     <------ data here: mail content --->
.    <-- to end the mail -->
telnet localhost 25
helo altnix.com
mail from: <champu@altnix.com>
rcpt to: <tree@lwqmail.altnix.com>
data
```

Hey TEST MAIL FROM TELNET

```
.
quit
[root@lwqmail mail]# vi /etc/aliases
                tree: champu
           tree: tree,champu
$ newaliases
```

ACL

Allow for altnix.com

Allow for 192.168.10.0/24

Deny for dummy.org

```
$ vi /etc/mail/access
@altnix.com                    RELAY
192.168.10.                    RELAY
@dummy.org                     REJECT
$ postmap /etc/mail/access
$ service sendmail restart
```

Q: Mail alias

A: modify /etc/aliases, run newaliases

Q: Receive mail for altnix.com

A: modify sendmail mc as above, and add domain to /etc/mail/local-host-names

```
$ vi /etc/mail/local-host-names
               altnix.com
```

Debugging:

mail -v root

mailq, mailq -Ac

sendmail -q

tail -f /var/log/maillog

Configure for pop3 (or imap)

A: 1) install dovecot

2) vi /etc/dovcot.conf

protocols = pop3

3) service dovecot restart

4) chkconfig dovecot on

Testing:

note: root is not permitted to login

echo "pop" | mail –s test student

telnet localhost 110

user student

pass student

stat

list

retr 1

quit

Setup a SMTP server

- john's mails should be spooled to /var/spool/mail/john
- Your server should accept mails from remote networks [internet]

1. $ cd /etc/mail/
2. $ cp sendmail.mc sendmail.mc.org
3. $ cp sendmail.cf sendmail.cf.org
4. $ vi /etc/mail/sendmail.mc

Find this line :

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

add the word dnl to the beginning so it looks like this :

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

```
$ m4 /etc/mail/sendmail.mc > /etc/mail/sendmail.cf
$ chkconfig --level 35 sendmail on
$ service sendmail restart
```

john's mails should be spooled to /var/spool/mail/john

Nothing to do. This is done by default by sendmail*

```
$ netstat -antp | grep :25
127.0.0.1:25
```

---- > previous output before you run m4 command

0.0.0.1:25

---- > after u r m4 command output will be look like this

Your local domain is altnix.com. Configure the send mail server for your local LAN by following these conditions

- Relay the mail from 192.168.10.0/24 network
- If any mail coming from dummy.com domain block all mails
- user5's mail should be get by user2 and himself

1. Edit /etc/mail/local-host-names

   altnix.com

2. $ vi /etc/mail/sendmail.mc

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

```
$ m4 /etc/mail/sendmail.mc > /etc/mail/sendmail.cf
$ vi /etc/mail/access
      192.168.10 RELAY
      @dummy.com REJECT
$ chkconfig --level 35 sendmail on
$ service sendmail restart
$ Edit /etc/dovecot.conf
```

```
        protocols = imap imaps pop3 pop3s
$ chkconfig --level 35 dovecot on
$ service dovecot restart
$ vi /etc/aliases
  user5: user2
$ newaliases
```

All mails to altnix.com should get by dhoni user

```
$ vi /etc/mail/virtusertable
      @altnix.com dhoni
$ service sendmail restart
$ chkconfig --level 35 sendmail on
```

Create a encapsulated SSL imap server \{IMAPS\}.

- Create an IMAP certificate for your hostname
- In [CN], put champu.altnix.com
- Only eric should be allowed from .altnix.com
  and all from .dummy.com should be denied

OR

- User eric should be able to access mail using IMAP over SSL
- Create an IMAP certificate for your hostname
- Only eric should be allowed from .altnix.com
  and all from .dummy.com should be denied

```
$ vi /etc/dovecot.conf
 protocols pop3 pop3s imap imaps
$ cd /usr/share/ssl/certs
$ mv dovecot.pem dovecot.pem_OLD
$ make dovecot.pem
     Country Name (2 letter code) [GB]:IN
```

```
State or Province Name (full name) [Berkshire]:Maharashtra
Locality Name (eg, city) [Newbury]:Mumbai
Organization Name (eg, company) [My Company Ltd]:Altnix
Organizational Unit Name (eg, section) []:Admin
Common Name: champu.altnix.com
```

```
$ cp dovecot.pem /usr/share/ssl/private
$ service dovecot restart
$ netstat -antp | grep :143
$ netstat -antp | grep :110
$ netstat -antp | grep :993
$ netstat -antp | grep :995
$ chkconfig --level 35 dovecot on
```

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 you can use your 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.

You 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 you can add to your /etc/sendmail.mc file; they should all be on one line.

- RFC-Ignorant: A valid IP address checker.

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

- Easynet: An open proxy list.

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

- Spamcop: A spammer blacklist.

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

- Spamhaus: A spammer blacklist.

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

Be sure to visit the URLs listed to learn more about the individual services.

Spamassassin

Once sendmail receives an e-mail message, it hands the message over to procmail, which is the application that actually places the e-mail in user mailboxes on the mail server. You can make procmail temporarily hand over control to another program, such as a spam filter. The most commonly used filter is spamassassin.

spamassassin doesn't delete spam, it merely adds the word "spam" to the beginning of the subject line of suspected spam e-mails. You can then configure the e-mail filter rules in Outlook Express or any other mail client to either delete the suspect message or store it in a special Spam folder.

Downloading And Installing Spamassassin

Most RedHat and Fedora Linux software products are available in the RPM format. When searching for the RPMs, remember that the filename usually starts with the software package name and is followed by a version number, as in spamassassin-2.60-2.i386.rpm. (For help downloading, see Chapter 6, "Installing RPM Software").

Starting Spamassassin

You can use the chkconfig command to get spamassassin configured to start at boot:

```
[root@bigboy tmp]# chkconfig --level 35 spamassassin on
```

To start, stop, and restart spamassassin after booting:

```
[root@bigboy tmp]# service spamassassin start
[root@bigboy tmp]# service spamassassin stop
[root@bigboy tmp]# service spamassassin restart
```

Configuring procmail for spamassassin

The /etc/procmailrc file is used by procmail to determine the procmail helper programs that should be used to filter mail. This file isn't created by default.

spamassassin has a template you can use called /etc/mail/spamassassin/spamassassin-spamc.rc. Copy the template to the /etc directory.

```
[root@bigboy tmp]# cp /etc/mail/spamassassin/spamassassin-spamc.rc
/etc/procmailrc
```

Configuring Spamassassin

The spamassassin configuration file is named /etc/mail/spamassassin/local.cf. A full listing of all the options available in the local.cf file can be found in the Linux man pages using the following command:

```
[root@bigboy tmp]# man Mail::SpamAssassin::Conf
```

You can customize this fully commented sample configuration file to meet your needs.

```
###########################################################################
# See 'perldoc Mail::SpamAssassin::Conf' for
# details of what can be adjusted.
###########################################################################
```

```
#
# These values can be overridden by editing
# ~/.spamassassin/user_prefs.cf (see spamassassin(1) for details)
#

# How many hits before a message is considered spam. The lower the
# number the more sensitive it is.

required_hits           5.0


# Whether to change the subject of suspected spam (1=Yes, 0=No)
rewrite_subject         1


# Text to prepend to subject if rewrite_subject is used
subject_tag             *****SPAM*****



 # Encapsulate spam in an attachment (1=Yes, 0=No)
report_safe             1


# Use terse version of the spam report (1=Yes, 0=No)
use_terse_report        0


# Enable the Bayes system (1=Yes, 0=No)
use_bayes               1


# Enable Bayes auto-learning (1=Yes, 0=No)
auto_learn              1


# Enable or disable network checks (1=Yes, 0=No)
skip_rbl_checks         0
use_razor2              1
use_dcc                 1
use_pyzor               1


# Mail using languages used in these country codes will not be
marked
# as being possibly spam in a foreign language.
# - english

ok_languages            en
```

```
# Mail using locales used in these country codes will not be marked
# as being possibly spam in a foreign language.

ok_locales              en
```

Be sure to restart spamassassin for your changes to take effect.

Testing spamassassin

You can test the validity of your local.cf file by using the spamassassin command with the --lint option. This will list any syntax problems that may exist. In this example two errors were found and corrected before the command was run again.

```
[root@bigboy tmp]# spamassassin -d --lint
Created user preferences file: /root/.spamassassin/user_prefs
config: SpamAssassin failed to parse line, skipping:
use_terse_report        0
config: SpamAssassin failed to parse line, skipping: auto_learn
1
lint: 2 issues detected.  please rerun with debug enabled for more
information.
[root@bigboy tmp]# vi /etc/mail/spamassassin/local.cf
...
...
...
[root@bigboy tmp]# spamassassin -d --lint
[root@bigboy tmp]
```

Startup spamassassin

The final steps are to configure spamassassin to start on booting and then to start it.

```
[root@bigboy tmp]# chkconfig spamassassin on
[root@bigboy tmp]# service spamassassin start
```

SQUID

Squid is...

Offshoot of the Harvest Project

a full-featured Web/FTP Proxy and Cache server

designed to run on Unix systems

free, open-source software

the result of many contributions by unpaid (and paid) volunteers

With Squid you can...

Use less bandwidth on your connection when surfing the Web

Reduce latency [time taken to load a web page]

Protect hosts on your intranet by proxying their web traffic

Collect statistics about web traffic on your NW

Prevent uses from visiting illegal sites

Ensure that valid users only surf the net

Enhance user's privacy by filtering sensitive info from web requests

Reduce the load on your web servers

Convert encrypted [HTTPS] requests on one side, to unencrypted [HTTP]

requests on the other

With Squid you CANNOT...

Proxy email [SMTP], instant messaging, or IRC

Squid supports...

proxying and caching of HTTP, FTP, and other

proxying for SSL encrypted traffic

cache hierarchies

ICP, HTCP, CARP, Cache Digests

transparent caching

WCCP (Squid v2.3 and above)

extensive sophisticated access controls

HTTP server acceleration aka surrogate mode

HTTP Interception Caching

SNMP

caching of DNS lookups

URL redirection

traffic shaping

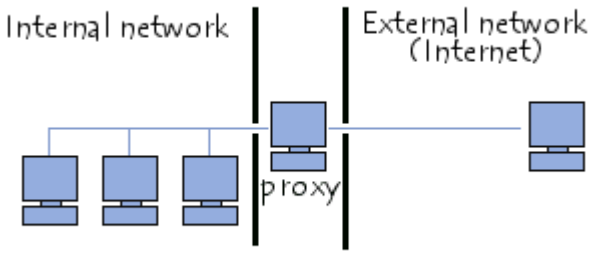
numerous external auth modules

advanced disk storage options

1. Proxy servers

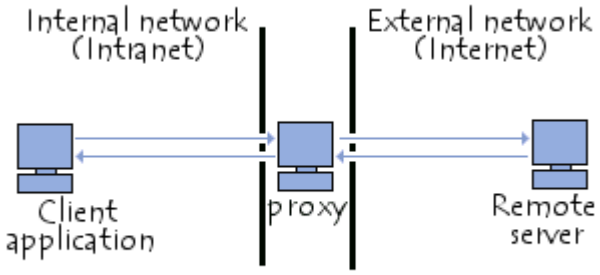
A **proxy** server is a machine which acts as an intermediary between the computers of a local area network (sometimes using protocols other than TCP/IP) and the Internet

Most of the time the proxy server is used for the web, and when it is, it's an HTTP proxy. However, there can be proxy servers for every application protocol (FTP, etc.).



2. The operating principle of a proxy server

The basic operating principle of a proxy server is quite simple: It is server which acts as a "proxy" for an application by making a request on the Internet in its stead. This way, whenever a user connects to the Internet using a client application configured to use a proxy server, the application will first connect to the proxy server and give it its request. The proxy server then connects to the server which the client application wants to connect to and sends that server the request. Next, the server gives its reply to the proxy, which then finally sends it to the application client



3. Features of a proxy server

Nowadays, by using TCP/IP within local area networks, the relaying role that the proxy server plays is handled directly by gateways and routers. However, proxy servers are still being used, as they have some other features.

4. Caching

Most proxies have a **cache**, the ability to keep pages commonly visited by users in memory (or "in cache"), so they can provide them as quickly as possible. Indeed, the term "cache" is used often in computer science to refer to a temporary data storage space (also sometimes called a "buffer.")

A proxy server with the ability to cache information is generally called a "**proxy-cache** server".

The feature, implemented on some proxy servers, is used both to reduce Internet bandwidth use and to reduce document loading time for users.

Nevertheless, to achieve this, the proxy must compare the data it stores in cached memory with the remote data on a regular basis, in order to ensure that the cached data is still valid.

5. Filtering

What's more, by using a proxy server, connections can be tracked by creating *logs* for systematically recording user queries when they request connections to the Internet

Because of this, Internet connections can be filtered, by analysing both client requests and server replies. When filtering is done by comparing a client's request to a list of authorised requests, this is called *whitelisting*, and when it's done with a list of forbidden sites, it's called *blacklisting*. Finally, analysing server replies that comply with a list of criteria (such as keywords) is called *content filtering*.

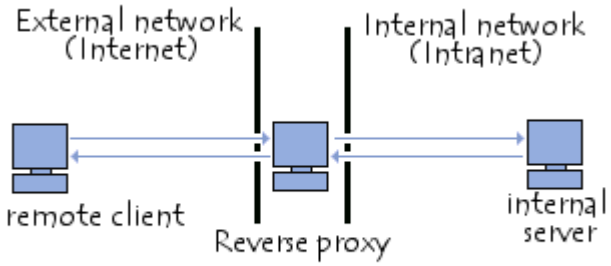
6. Authentication

As a proxy is an indispensable intermediary tool for internal network users who want to access external resources, it can sometimes be used to authenticate users, meaning to ask them to identify themselves, such as with a username and password. It is also easy to grant access to external resources only to individuals authorised to do so, and to record each use of external resources in log files.

This type of mechanism, when implemented, obviously raises many issues related to individual liberties and personal rights.

7. Reverse-proxy servers

A *reverse-proxy* is a "backwards" proxy-cache server; it's a proxy server that, rather than allowing internal users to access the Internet, lets Internet users indirectly access certain internal servers.



The reverse-proxy server is used as an intermediary by Internet users who want to access an internal website, by sending it requests indirectly. With a reverse-proxy, the web server is protected from direct outside attacks, which increases the internal network's strength. What's more, a reverse-proxy's cache function can lower the workload if the server it is assigned to, and for this reason is sometimes called a *server accelerator*.

Finally, with perfected algorithms, the reverse-proxy can distribute the workload by redirecting requests to other, similar servers; this process is called load balancing.

```
$ squid -k

==========>  parse            Check if squid.conf is OK and
syntax-free

==========>  check            Check if SQUID is running

==========>  reconfigure      Re-Read squid.conf w/o stopping
[refresh]

                             aka  service squid reload

             rotate          rotate the log files


              shutdown        Shutdown SQUID gracefully

                              aka  service squid stop

             interrupt       Kill SQUID w/o waiting for trns to
finish

             kill            Kill SQUID mercilessly

             debug           Puts SQUID in debugging mode
```

$ squid -N Is SQUID running ?

$ squid -Nd1 Is DNS working ?

$ squid -zX Init the cache / swap dirs. Used when running SQUID for the first time.

Done by SQUID initscript anyway [start()]

X --> To watch the progress of cache creating

SQUID should not be running when this is done!

$ squid -F Make SQUID refuse all requests until it rebuilds the storage metadata

Done by SQUID initscript anyway [start()]

$ squid -D Disables/prevents initial DNS tests

Done by SQUID initscript anyway [start()]

Already specified in /etc/sysconfig/squid which is read by squid's initscript

$ squid -N -d1 -D Run SQUID with logging to stderr in the fg not bg with level 1 debugging

-N -> Keep SQUID in the fg Does not read /etc/sysconfig/squid
-d1 -> Display level 1 debugging to stderr
-D -> Don't bother with DNS and die since Squid tries to do DNS lookups for a few common domains, and dies with an error if it is not able to resolve them thru /etc/resolv.conf See Directive #11 dns_testnames

$ squid -N Prevent SQUID from becoming a bg process

$ squid -s Enable logging to the syslogd daemon SQUID uses

LOCAL4 syslog facility Level 0 debug msgs are logged with priority

LOG_WARNING Level 1 debug msgs are logged with priority

LOG_NOTICE

Access Controls are the most imp part of your SQUID config file. You will use them to grant access to authorized users and keep out the bad guys.

You can use them to restrict, or prevent access to, certain material; to control request rewriting; to route requests thru a hierarchy; and to support different qualities of service.

What is Access Control ?

Access Control

1. Define a no of ACL Elements or ACL ELEMENTS or ACLE [These refer to specific aspects of client requests such as IP addrs, URL hostnames, request methods and origin server port nos]

2. Access Control Rules or RULES or ACRs [These rules appy to particular services or ops within SQUID eg http_access rules are applied to incoming HTTP requests]

i.e Access Control = ACL Elements + AC Rules
= ACLEs + ACRs

==========

ACCESS CONTROL ELEMENTS - ACLEs

ACL elements are the building blocks of SQUID's access control implementations.

Each ACL has a name, which you use when writing the rules

Eg. acl Workstations src 192.168.0.0/24

You can list multiple values for one ACL element

Eg. acl Http_ports 80 8000 8080

is the same as

acl Http_ports 80

acl Http_ports 8000

acl Http_ports 8080

Squid knows about the following types of ACLEs :

1 src : Source (client) IP addresses

2 srcdomain : Source (client) domain name

3 srcdom_regex : Source (client) domain with RE pattern matching

4 dst : Destination (server) IP addresses [Origin Server]

5 dstdomain : Destination (server) domain name

6 dstdom_regex : Destination (server) with RE pattern matching

7 url_regex : Match any part of a requested URL

8 urlpath_regex: Match any part of a requested URL. Omit protocol/hostname

9 time : Current Time of day, and day of week

10 port : Destination server port no [Dont confuse with #18 myport]

11 proto : Transfer protocol (HTTP,FTP,SSL)

12 method : HTTP request method (GET,PUT,HEAD,POST - FTP, SSL)

13 browser : Allowing / Disallowing Browsers

14 maxconn : Limit on max no of connections from a single client IP

15 arp : Ethernet (MAC) address matching. ARP-based ACLEs

16 proxy_auth : Username/Password authentication Using PAM

17 proxy_auth_regex: user authentication via external processes

18 myport : Local port no [cache] that Client connects to

19 myip : The local IP address of a client's connection

20 src_as : Source (client) Autonomous System number

21 dst_as : Destination (server) Autonomous System number

22 ident : String matching on the user's name

23 ident_regex : RE pattern matching on the user's name

24 referer_regex:

25 req_mime_type:

26 rep_mime_type:

27 snmp_community: SNMP community string matching

28 req_mime_type: RE pattern matching on the request content-type header

29 rep_mime_type: RE pattern matching on the reply (downloaded content) content-type header. This is only usable in the http_reply_access directive, not http_access.

30 external: lookup via external acl helper defined by external_acl_type

Base Types	Used by
==========	=======
IP addresses:	1. src
	2. dst
	3. myip
	4. src_as
	5. dst_as
Domain Names:	srcdomain
	dstdomain
	cache_host_domain directive
Usernames :	ident
	ident_regex
	proxy_auth
	proxy_auth_regex
REs :	srcdom_regex
	dstdom_regex
	url_regex
	urlpath_regex
	browser
	referer_regex
	ident_regex
	proxyauth_regex
	req_mime_type
	rep_mime_type
TCP Port Nos:	port
	myport

Restrict sites

Search for `Access Controls' and append following two lines:
acl blocksites dstdomain .gmail.com
http_access deny blocksites

Save and close the file. Restart Squid:
$ /etc/init.d/squid restart

Restrict word

Let us say you would like to deny access for anyone who browses to a URL with the word "bar" in it. Append following ACL:
acl blockregexurl url_regex -i porn
http_access deny blockregexurl

Restricting Web Access By Time

You can create access control lists with time parameters. For example, you can allow only business hour access from the home network, while always restricting access to host 192.168.1.23.

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

acl home_network src 192.168.1.0/24

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

acl RestrictedHost src 192.168.1.23

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

http_access deny RestrictedHost

http_access allow home_network business_hours

Or, you 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

Restrict .exe .mp3 .avi with customized error page

Now add following lines to your squid ACL section:

```
acl blockfiles urlpath_regex "/etc/squid/blocks.files.acl"
```

You want display custom error message when a file is blocked:

```
# Deny all blocked extension
deny_info ERR_BLOCKED_FILES blockfiles
http_access deny blockfiles
```

Now create /etc/squid/blocks.files.acl file:

```
# vi /etc/squid/blocks.files.acl
```

Append following text:

```
\.[Ee][Xx][Ee]$
\.[Aa][Vv][Ii]$
\.[Mm][Pp][Gg]$
\.[Mm][Pp][Ee][Gg]$
\.[Mm][Pp]3$
```

Create custom error message HTML file called ERR_BLOCKED_FILES in /etc/squid/error/ directory or /usr/share/squid/errors/English directory.

```
# vi ERR_BLOCKED_FILES
```

Append following content:

```
<HTML>

<HEAD>

<TITLE>ERROR: Blocked file content</TITLE>

</HEAD>

<BODY>

<H1>File is blocked due to new IT policy</H1>

<p>Please contact helpdesk for more information:</p>

Phone: 555-12435 (ext 44)<br>

Email: helpdesk@yourcorp.com<br>
```

Caution: Do not include HTML close tags *</HTML> </BODY>* as it will be closed by squid.

Save and close the file. Restart Squid:

```
# /etc/init.d/squid restart
```

Restrict Port

Locate your ACL section and add configuration directive as follows:

```
acl block_port port 1234
http_access deny block_port
http_access allow all
```

If you just want to skip a particular IP (192.168.1.5) try as follows:

```
acl block_port port 1234
acl no_block_port_ip src 192.168.1.5
http_access deny block_port !no_block_port_ip
http_access allow all
```

Close and save the file.
Restart squid proxy server:

```
$ /etc/init.d/squid restart
```

And finally deny all other access to this proxy

```
http_access allow localhost
```

Squid Bandwidth Limiting (Bandwidth Throtling)

How to Write Delay Pool Script in Squid
In this scenario we are going to creat 3 pools

128kbp
256kbps
Unlimited for Admin

```
Acl 128kbps 192.168.0.200/32
Acl 256kbps 192.168.0.201/32
Acl admin 192.168.0.205/32
```

You just creat the ACL Named net, net2, net3

```
delay_pools 3
delay_class 1 2
delay_access 1 deny admin
delay_access 1 deny 256kbps
delay_access 1 allow 128kbps
delay_parameters 1 -1/-1 16000/16000

delay_class 2 2
delay_access 2 deny admin
delay_access 2 allow 256kbps
delay_parameters 2 -1/-1 32000/32000

delay_class 3 2
delay_access 3 allow admin
delay_parameters 3 -1/-1 -1/-1
```

Vi/Vim Examples

what is VI-editor ?

While in vi you can run AIX commands without exiting the editing session. The! creates a shell to execute the command that follows.

1. :!ls will create a shell

2. All files in the current directory are listed. Press return to exit the shell and return to the vi session or...

3. While still in command mode, issue the :r snacks command

4. The contents of snacks, in this case, are read into the vi file. By default, it will appear after the current line.

If you need to run a series of commands without returning to vi after the first command is executed, enter :sh. When you have run all the commands, press to exit the shell and return to vi.

Cursor movement

- h – move left
- j – move down
- k – move up
- l – move right
- w – jump by start of words (punctuation considered words)
- W – jump by words (spaces separate words)
- e – jump to end of words (punctuation considered words)
- E – jump to end of words (no punctuation)
- b – jump backward by words (punctuation considered words)
- B – jump backward by words (no punctuation)
- 0 – (zero) start of line
- ^ – first non-blank character of line
- $ – end of line
- G – Go To command (prefix with number – 5G goes to line 5)

Note: Prefix a cursor movement command with a number to repeat it. For example, 4j moves down 4 lines.

Insert Mode – Inserting/Appending text

13. i – start insert mode at cursor
14. I – insert at the beginning of the line
15. a – append after the cursor
16. A – append at the end of the line
17. o – open (append) blank line below current line (no need to press return)
18. O – open blank line above current line
19. ea – append at end of word
20. Esc – exit insert mode

Editing

- r - replace a single character (does not use insert mode)
- J - join line below to the current one
- cc - change (replace) an entire line
- cw - change (replace) to the end of word
- c$ - change (replace) to the end of line
- s - delete character at cursor and subsitute text
- S - delete line at cursor and substitute text (same as cc)
- xp - transpose two letters (delete and paste, technically)
- u - undo
- . - repeat last command

Marking text (visual mode)

v - start visual mode, mark lines, then do command (such as y-yank)
V - start Linewise visual mode
o - move to other end of marked area
Ctrl+v - start visual block mode
O - move to Other corner of block
aw - mark a word
ab - a () block (with braces)
aB - a {} block (with brackets)
ib - inner () block
iB - inner {} block
Esc - exit visual mode

Visual commands

3. > - shift right
4. < - shift left
5. y - yank (copy) marked text
6. d - delete marked text
7. ~ - switch case

Cut and Paste

- yy - yank (copy) a line
- 2yy - yank 2 lines
- yw - yank word
- y$ - yank to end of line
- p - put (paste) the clipboard after cursor
- P - put (paste) before cursor
- dd - delete (cut) a line
- dw - delete (cut) the current word
- x - delete (cut) current character

Exiting

- :w - write (save) the file, but don't exit
- :wq - write (save) and quit
- :q - quit (fails if anything has changed)
- :q! - quit and throw away changes

Search/Replace

- */pattern* – search for pattern
- ?*pattern* – search backward for pattern
- n – repeat search in same direction
- N – repeat search in opposite direction
- :%s/*old*/*new*/g – replace all *old* with *new* throughout file
- :%s/*old*/*new*/gc – replace all *old* with *new* throughout file with confirmations

Working with multiple files

- :e *filename* – Edit a file in a new buffer
- :bnext (or :bn) – go to next buffer
- :bprev (of :bp) – go to previous buffer
- :bd – delete a buffer (close a file)
- :sp *filename* – Open a file in a new buffer and split window
- ctrl+ws – Split windows
- ctrl+ww – switch between windows
- ctrl+wq – Quit a window
- ctrl+wv – Split windows vertically

- **VI Options:-**

vi has many modes of operation. Some of these will affect the way text is presented, while others will make editing easier for novice users.

```
:set all display all settings
:set display settings different than the default
:set ai sets autoindent on
:set noai turns autoindent mode off
:set nu enables line numbers
:set nonu turns line numbers off
:set list displays non-printable characters
:set nolist hides non-printable characters
:set showmode shows the current mode of operation
:set noshowmode hides mode of operation
:set ts=4 sets tabs to 4-character jumps
:set ic ignores case sensitivity
:set noic case sensitive
```

- **Search**

```
/word     Search “word” from top to bottom
?word     Search “word” from bottom to top
/jo[ha]n     Search “john” or “joan”
```

```
/\< the     Search “the”, “theatre” or “then”
/the\>     Search “the” or “breathe”
/\     Search “the”
/\     Search all words of 4 letters
/\/     Search “fred” but not “alfred” or “frederick”
/fred\|joe     Search “fred” or “joe”
/\     Search exactly 4 digits
/^\n\{3}     Find 3 empty lines
:bufdo /searchstr/     Search in all open files
```

- **Replace**

```
:%s/old/new/g     Replace all occurences of “old” by “new” in
file
:%s/old/new/gw     Replace all occurences with confirmation
:2,35s/old/new/g     Replace all occurences between lines 2 and
35
:5,$s/old/new/g     Replace all occurences from line 5 to EOF
:%s/^/hello/g     Replace the begining of each line by “hello”
:%s/$/Harry/g     Replace the end of each line by “Harry”
:%s/onward/forward/gi     Replace “onward” by “forward” , case
unsensitive
:%s/ *$//g     Delete all white spaces
:g/string/d     Delete all lines containing “string”
:v/string/d     Delete all lines containing which didn't
contain “string”
:s/Bill/Steve/     Replace the first occurence of “Bill” by
“Steve” in current line
:s/Bill/Steve/g     Replace “Bill” by “Steve” in current line
:%s/Bill/Steve/g     Replace “Bill” by “Steve” in all the file
:%s/\r//g     Delete DOS carriage returns (^M)
:%s/\r/\r/g     Transform DOS carriage returns in returns
:%s#]\+>##g     Delete HTML tags but keeps text
:%s/^\(.*\)\n\1$/\1/     Delete lines which appears twice
Ctrl+a     Increment number under the cursor
Ctrl+x     Decrement number under cursor
ggVGg?     Change text to Rot13
```

4. Case

```
Vu     Lowercase line
VU     Uppercase line
g~~     Invert case
vEU     Switch word to uppercase
vE~     Modify word case
ggguG     Set all text to lowercase
:set ignorecase     Ignore case in searches
:set smartcase     Ignore case in searches excepted if an
uppercase letter is used
:%s/\<./\u&/g     Sets first letter of each word to uppercase
:%s/\<./\l&/g     Sets first letter of each word to lowercase
:%s/.*/\u&     Sets first letter of each line to uppercase
:%s/.*/\l&     Sets first letter of each line to lowercase
```

- **Read/Write files**

```
:1,10 w outfile     Saves lines 1 to 10 in outfile
:1,10 w >> outfile     Appends lines 1 to 10 to outfile
:r infile     Insert the content of infile
```

```
:23r infile     Insert the content of infile under line 23
```

- **File explorer**

```
:e .     Open integrated file explorer
:Sex     Split window and open integrated file explorer
:browse e     Graphical file explorer
:ls     List buffers
:cd ..     Move to parent directory
:args     List files
:args *.php     Open file list
:grep expression *.php     Returns a list of .php files
contening expression
gf     Open file name under cursor
```

- **Interact with Unix**

```
:!pwd     Execute the “pwd” unix command, then returns to Vi
!!pwd     Execute the “pwd” unix command and insert output in
file
:sh     Temporary returns to Unix
$exit     Retourns to Vi
```

- **Alignment**

```
:%!fmt     Align all lines
!}fmt     Align all lines at the current position
5!!fmt     Align the next 5 lines
```

- **Tabs**

```
:tabnew     Creates a new tab
gt     Show next tab
:tabfirst     Show first tab
:tablast     Show last tab
:tabm n(position)     Rearrange tabs
:tabdo %s/foo/bar/g     Execute a command in all tabs
:tab ball     Puts all open files in tabs
```

- **Window spliting**

```
:e filename     Edit filename in current window
:split filename     Split the window and open filename
ctrl-w up arrow     Puts cursor in top window
ctrl-w ctrl-w     Puts cursor in next window
ctrl-w_     Maximise current window
ctrl-w=     Gives the same size to all windows
10 ctrl-w+     Add 10 lines to current window
:vsplit file     Split window vertically
:sview file     Same as :split in readonly mode
:hide     Close current window
:only     Close all windows, excepted current
```

:b 2 Open #2 in this window

- **Auto-completion**

Ctrl+n Ctrl+p (in insert mode) Complete word
Ctrl+x Ctrl+l Complete line
:set dictionary=dict Define dict as a dictionnary
Ctrl+x Ctrl+k Complete with dictionary

8. Marks

mk Marks current position as k
'k Moves cursor to mark k
d'k Delete all until mark k
Abbreviations
:ab pr printf("This is a Demo Ver \n"); Define pr as abbreviation of printf("This is a Demo Ver \n");

- **Text indent**

:set autoindent Turn on auto-indent
:set smartindent Turn on intelligent auto-indent
:set shiftwidth=4 Defines 4 spaces as indent size
ctrl-t, ctrl-d Indent/un-indent in insert mode
>> Indent
<< Un-indent

- **Syntax highlighting**

:syntax on Turn on syntax highlighting
:syntax off Turn off syntax highlighting
:set syntax=perl Force syntax highlighting

- **How to Exit**

:q[uit] Quit Vim. This fails when changes have been made.
:q[uit]! Quit without writing.
:cq[uit] Quit always, without writing.
:wq Write the current file and exit.
:wq! Write the current file and exit always.
:wq {file} Write to {file}. Exit if not editing the last
:wq! {file} Write to {file} and exit always.
:[range]wq[!] [file] Same as above, but only write the lines in [range].
ZZ Write current file, if modified, and exit.
ZQ Quit current file and exit (same as ":q!").

5. Editing a File

:e[dit] Edit the current file. This is useful to re-edit the current file, when it has been changed outside of Vim.
:e[dit]! Edit the current file always. Discard any changes to the current buffer. This is useful if you want to start all over again.
:e[dit] {file} Edit {file}.

:e[dit]! {file} Edit {file} always. Discard any changes to the current buffer.
gf Edit the file whose name is under or after the cursor. Mnemonic: "goto file".

- **Inserting Text**

a Append text after the cursor [count] times.
A Append text at the end of the line [count] times.
i Insert text before the cursor [count] times.
I Insert text before the first non-blank in the line [count] times.
gI Insert text in column 1 [count] times.
o Begin a new line below the cursor and insert text, repeat [count] times.
O Begin a new line above the cursor and insert text, repeat [count] times.

- **Inserting a file**

:r[ead] [name] Insert the file [name] below the cursor.
:r[ead] !{cmd} Execute {cmd} and insert its standard output below the cursor.

Ref -

http://www.catswhocode.com/blog/100-vim-commands-every-programmer-should-know
http://www.fortunecity.com/skyscraper/terminus/435/
http://www.thegeekstuff.com/2010/04/vim-editor-tutorial/

****

Best OF Luck

****