Linux Pocket Reference For System Administrators

(5th Edition)

By

Muhammad Kamran Azeem

Spanish translation by:

Gerardo Diaz

gerardobdiaz@arnet.com.ar

http://gerardodiaz.tripod.com

Linux Pocket Reference for System Administrators

Last revised:

20050831

Published by:

Enabling Technologies (E-Tech)

Peshawar, Pakistan.

Printed by:

Trade link

Mezzanine floor, Wali-Center, Blue Area, Islamabad, Pakistan.

About *Emerging Technologies*

Emerging Technologies is a concept, a vision, a theme, a dream, an objective and a goal, to bring Open-Source technologies to the youth of Pakistan.

Emerging Technologies, formerly POSEP (Pakistan Open Source Education Project), was idealized by some people from both Islamabad and Peshawar, in Pakistan. Later on it materialized in Peshawar, in mid of year 2004.

Emerging Technologies deals in Open-Source software development and training, ranging from programming languages like C/C++, PERL to databases like PostgreSQL and MySQL.

For more information about this project, please check www.etech.com.pk

You can also write to info@etech.com.pk

This page intentionally left blank

Table of Contents

About the Author

Muhammad Kamran Azeem is in computing field since 1993 when he ever first touched a personal computer keyboard. His first passion was batch programming in DOS. He brings with him around twelve years of programming experience (by the time of this writing) in technologies ranging from GWBASIC, Pascal, Fortran, COBOL, FoxPro, C/C++, Oracle Developer and Visual BASIC, to PERL, PHP, PostgreSQL, Apache, Sendmail, Q-Mail and similar open-source server side technologies. He is certified under Oracle's OCP-DBA (Oracle8) and Cisco's CCNA program. Being with SCO-Unix since 1997, he got hooked and locked with Linux in year 2000. Since then Linux takes away major portion of his day.

He has been teaching O and A level students of Cambridge University in Pakistan since 1998. He has also served as an instructor to graduate and post-graduate level in different institutes in Islamabad, such as Preston University and Petroman Training Institute, for the subjects as: Software Engineering, C/C++, Data Structures, Oracle, Visual BASIC and Advance Operating Systems. He is also a well-known trainer for "System Administration" and "ISP setup" courses in major IT training institutes of Islamabad such as NCR, COMSATS, IBMACE and COMWAVE.

He has designed and implemented Oracle based (and non-Oracle based) network systems for many organizations. He configured the ISP of Pakistan Telecommunication Authority and also developed their RADIUS (radius.pta.gov.pk). Recently he has configured ISP of Data4Networks, where is the prime maintainer of the web, mail and DNS servers.

His passion for Linux hinders his passion for trekking. When he is not in the office or not in home, then he may be busy somewhere else , trouble-shooting some linux based network.

His latest project was *POSEP, Pakistan Open Source Education Project*, which was then transformed into *Emerging Systems*, lately, by his friends in Peshawar.

Kamran is with Ora-Tech Systems in Islamabad since Jan 2005, where he is working as an Octopus (really!), dealing with Linux based servers and Oracle Technologies together, along security consultancy and training.

Kamran can be reached at: **mkazeem@isb.paknet.com.pk**

Dedication

To the sufferers of 9/11.

Acknowledgments

What-ever I am today, I could have never been without: blessings of Allah Almighty, my parents prayers and my hard work.

I acknowledge my parents for giving me education and letting me opt for taking computer as a hobby (and buying me my ever first computer in 1993), which later on became my profession. I would also like to thank my brothers and sisters, especially my brother *Imran*, who is a faster and better programmer than I am. He has always come up with ideas and solutions beyond my imagination. He is pro-Microsoft type of person, so I never try to convert him to Linux.

I would like to thank my friend *Umar Ghaffar*, (I have failed to find him since 1998) who indeed brought me into the magnificent field of computers way back in 1993. At that time we used to play (popular at that time) games, like TMNT2 (Teenage Mutant Ninja Turtles 2) and do a lot of batch programs together. I would also like to thank my friend *Shahzad Siddique* who literally hijacked me from the world of Databases and introduced me to the world of Networks. “Welcome to the world of networks”, were his words when I cleared my CCNA certification.

I want to thank my friends and my students as I alone could not have done it. I would specially thank my friend and my *ustaad* (teacher), *Feysal Abdullah*, who believes that Linux will never make up to the desktop of end-user. He remains mysterious and each time I meet him, I am always left wondering on the depth of his knowledge on life, networks and Linux.

I would like to thank Mr. *Abd-ur-Rauf Bhatti*, CEO of COMWAVE Institute Of Technology, Islamabad, who provided me a chance to prove my skills in Linux. The major portion of this book was structured and tested in this institute. By the grace of Almighty Allah, I have the honor of teaching Linux and networking to professionals in this prestigious institute.

I would like to thank Microsoft Corporation for providing us enough tools to start our computing career with, only to realize later that it was all just *gimmik* and that those tools really didn't work as they should have. And also to their attitude / intention of owning everything on the planet (and may be the universe itself) which makes many people angry and furious.

I would specially like to thank the *Emerging Systems* team members: Nayyar, *Inam-ul-Haq, Tauseef, Asif, (Professor) Mahmood* and our programmer / chief cook *Sajjad*, for joining hands, and for providing the environment in which both *POSEP*, now *Emerging Systems*, and 5th edition of this book was possible.

Also my beloved wife, for allowing me to spend time with this book. Allah blessed me with a baby daughter *Fatima*, just after I finished reviewing this book. Now, once this book is being printed commercially, she is one year old. Quite some progress!

Special Thanks to ***Gerardo Diaz*** (`gerardobdiaz@arnet.com.ar`) for translating this book into Spanish.

At last I would also like to thank all the people out there on the Internet who have contributed to the Linux community in what ever possible way.

Muhammad Kamran Azeem

Translations(and translators)

Translation in Spanish (in 2004) by:

Gerardo Diaz

gerardobdiaz@arnet.com.ar

http://gerardodiaz.tripod.com

(Helloooo .... are you still there buddy? 'cause 6th edition is coming soon !)

Revision history

First version / edition of this book came out in December 2002.

Like most things in life, this book also could not remain static and thus you have the fifth edition in front of you. Last revised on 31 Aug 2005.

In this 5th edition the major changes are

1) example domain name is changed,
2) diagrams are introduced,
3) more text is introduced in many chapters
4) New chapters about **Network wiring / cable schemes**, **DHCP** , **Advance permissions** (sudo, umask, setgid, etc) and **FTP** are introduced.

Preface

Why this book was written?, has a simple and straight answer: **There was a dire need for it**. People, which includes professional system administrators and professional Linux trainers, had certainly spend some tough time while recalling those commands and procedures which just sometimes go away from the mind. Similarly, in case of a network trouble shoot or a problem hunt, there lacked a proper sequence to follow, which could identify the problems.

This book is NOT a detailed theory based encyclopedia. For that kind of explanation, please refer to Linux Documentation Project (`www.tldp.org`) or the books from O'Reilly, IDG, Sybex, etc. So in short you should not expect explanation of things in this text. This book is "**NOT"** for newbies. This is a reference book and should be treated as reference only. There might be things which might not work because of your machine or your particular Linux distribution or my incorrect explanation. As far as this book is concerned, all material is related to RedHat distribution.

This book was initially written using OpenOffice.org office suite running on a RedHat Linux 8 powered computer. Fifth edition coming out from a Fedora Core 4 powered computer.

Please feel free to inform me about any errors, suggestions at: **`mkazeem@isb.paknet.com.pk`**

Terms and conditions

For private and personal use you are allowed to reproduce this book in any form as far as the name of the original author (that is myself) is maintained in the new production. You are not liable to give me any money nor you are allowed to charge any money from any one except the cost of the media used. I would appreciate an email once you print this book in any number for any one.

For commercial use, the permission must be taken from me personally via email. This permission will only be granted by settling on a contract or agreement between me (the author) and you (the commercial printer).

Latest version of this book can always be found at:
www.geocities.com/mk_azeem

It is also your moral and ethical duty to print ***all*** pages of this book, from page one till the last page and keep them in one binding. Printing it without the starting pages is strongly discouraged and is like someone without a face. I would not like my hard-work to be censored by your opinions and views about life and all that.

Permission to translate this book in any language is granted to everyone, provided the name of original author (me in this case, Kamran Azeem) is maintained and is clearly visible / readable on the title of the new (translated) book.

Muhammad Kamran Azeem

Disclaimer

The computer technology (rather all technologies) are man made and thus are NOT perfect. So the author is not responsible for any damage to any machine, software , living (or dead) soul, caused by the contents of this book in any possible way.

Chapter 1: **Basics**

It would be better to start with the commonly used commands, which at times just slip out of mind. Just brief description is given below.

Directory and disk usage related commands:

Following command gives a list of files and directories in a long listing (even hidden files will be displayed), human readable sizes, KB, MB. Also sorted by time (-t) and in reverse order (-r) i.e. Newest in the end of listing.

```
[root@mainserver /]# ls -l -h -a -t -r
```

Following command captures lines showing directories only page by page, to check for files use ^f. (This ^ d or ^f is NOT ctrl-d or ctrl-f)

```
[root@mainserver /]# ls -l | grep "^d" | less
```

To check for free space on your partitions:

```
[root@mainserver /]# df -h
```

To check for the used space by a particular directory:

```
[root@mainserver /]# du –c –h /home/kamran
```
or
```
[root@mainserver /]# du –s –h /home/kamran
```

Mounting / un-mounting floppies /cd-rom/ directories:

To check for currently mounted partitions, use mount command without any arguments:

```
[root@mainserver /]# mount
/dev/hda6 on / type ext3 (rw)
none on /proc type proc (rw)
usbdevfs on /proc/bus/usb type usbdevfs (rw)
/dev/hda1 on /boot type ext3 (rw)
```

```
/dev/hda2 on /data type ext3 (rw)
none on /dev/pts type devpts(rw,gid=5,mode=620)
none on /dev/shm type tmpfs (rw)
/dev/hda3 on /mnt/hda3 type vfat (rw)
```

You can also use the mounted files system table ***mtab*** to check for currently mounted partitions.

```
[root@mainserver /]# cat /etc/mtab
/dev/hda6 / ext3 rw 0 0
none /proc proc rw 0 0
usbdevfs /proc/bus/usb usbdevfs rw 0 0
/dev/hda1 /boot ext3 rw 0 0
/dev/hda2 /data ext3 rw 0 0
none /dev/pts devpts rw,gid=5,mode=620 0 0
none /dev/shm tmpfs rw 0 0
/dev/hda3 /mnt/hda3 vfat rw 0 0
```

Following command mounts the floppy drive in mount point */mnt/floppy*

```
[root@mainserver /]# mount /dev/fd0 /mnt/floppy
```

Similarly

```
[root@mainserver /]# mount /dev/cdrom /mnt/cdrom
```

Following command mounts your *VFAT* partition, C: drive, in */mnt/windir*

```
[root@mainserver /]# mount -t vfat /dev/hda2 /mnt/windir
```

If you want to look into an *.iso* image , you can mount it to an empty directory (*/mnt/cdimage*) using *loopback* option and then use it as an ordinary directory*.*

```
[root@mainserver /]# mount -o loop  /downloads/cd1.iso  /mnt/cdimage
```

Following command mounts an *NFS* share */usr/redhat72* from the machine *192.168.1.254* to your local mount point */mnt/cdrom*:

```
[root@mainserver /]# mount –t nfs 192.168.1.254:/usr/redhat72 /mnt/cdrom
```

To unmount any mount point use the following command but the mount point must not be busy:

```
[root@mainserver /]# umount /mnt/cdrom
```

Note that you cannot mount an *NTFS* partition unless it's support is enabled in the kernel. See chapter **Kernel Modification and Compilation**. By this time of writing only **read-only** support for NTFS is available in RedHat linux. Writing is in experimentation/ development stage and will destroy your data.

File copying / moving / renaming / deletion / compression and decompression:

Following command copies all (non-hidden) files, including any subdirectories from */home/kamran* to */mnt/floppy* , overwriting any previously existing files (***--reply=yes***)

```
[root@mainserver /]# cp -vr /home/kamran/* /mnt/floppy/ --reply=yes
```

The above command will change the ownership of the newly copied files on */mnt/floppy* to the *uid* and *gid* of the user issuing the copy (cp) command, in this case the user *root*. To preserve the file permissions while copying to the new location, use the **-p** switch as well. For example:

```
[root@mainserver /]# cp -v -r -p  /home/kamran/*  /mnt/floppy/ --reply=yes
```

Following command deletes all files, including subdirectories, forcefully (without asking yes/no) from the directory */tmp.*

```
[root@mainserver /]# rm -fr /tmp/*
```

Following command creates a tar file in */tmp* as *mytar.tar* from the files in current directory

```
[root@mainserver /]# tar cvf  /tmp/mytar.tar   *
```

Following command untars the tar file *mytar.tar* from */tmp* in current directory

```
[root@mainserver /]# tar xvf /tmp/mytar.tar
```

Following command untars the tar file *mytar.tar* from */tmp* in */backup/restore*

```
[root@mainserver /]# tar xvf /tmp/mytar.tar  -C /backup/restore
```

Following command adds a file (*newfile.pl*) to already existing tar file *existing.tar* .

```
[root@mainserver /]# tar rvf existing.tar  newfile.pl
```

Following command compresses (using maximum compression) the tar file *mytar.tar* to a 'gzip' file *mytar.tar.gz*

```
[root@mainserver /]# gzip -v9 mytar.tar
```

Following command extracts / decompresses the tar file out of the *.gz* file in the current directory

```
[root@mainserver /]# gzip -vd mytar.tar.gz
```

Disk checking and formatting:

List of all partitions on all attached hard drives (both SCSI and IDE)

```
[root@mainserver /]# fdisk -l
```

To resize partitions use:

```
[root@mainserver /]# parted
```

Following command formats a floppy disk in ***ext2*** format, also checks for and marks, bad sectors

```
[root@mainserver /]# mke2fs -c /dev/fd0
```

Following command formats a floppy disk in dos format

```
[root@mainserver /]# mkdosfs /dev/fd0
```

Following command checks for errors on the partition, the partition being checked should be in unmounted state during check.

```
[root@mainserver /]# e2fsck /dev/hda3
or
[root@mainserver /]# fsck -t ext2 /dev/hda3
```

Process management:

Following command shows the list of all running processes with their process ids (*PID*), even owned by other users:

```
[root@mainserver /]# ps aux | less
```

You can also use pstree :

```
[root@mainserver /]# pstree
```

Following command will kill a specific process, say process id *12432*:

```
[root@mainserver /]# kill 12432
```

or

```
[root@mainserver /]# kill –s KILL 12432
```

or

```
[root@mainserver /]# kill –KILL 12432
```

or

```
[root@mainserver /]# kill –9 12432
```

Package management:

Following command lists all the currently installed packages and captures only those lines which have the word *sendmail* in them

```
[root@mainserver /]# rpm -qa | grep sendmail
```

or

```
[root@mainserver /]# rpm -qa sendmail*
```

To check presence of the package 'bind' in the system, use the following command:

```
[root@mainserver /]# rpm -q bind
```

Following command lists all the information about the specific installed package.

```
[root@mainserver /]# rpm -qi sendmail
```

Following command lists all the documentation files and their location of a particular installed package.

```
[root@mainserver /]# rpm -qd sendmail
```

Following command lists file locations of all of the files of a particular installed package.

```
[root@mainserver /]# rpm -ql sendmail
```

Following command lists all information of the un-installed package:

```
[root@mainserver /]# rpm -qpi ymessenger-0.99.19-1.i386.rpm
```

Similarly,you can check the list of files a package contains, before actually installing it:

```
[root@mainserver /]# rpm -qpl ymessenger-0.99.19-1.i386.rpm
```

Also to check, which package a file belongs to by:

```
[root@mainserver /]# rpm -qf /etc/mail/sendmail.cf
sendmail-8.12.8-4
```

If not installed then install it from (mounted) Redhat CD by using the following command:

```
[root@mainserver /]# rpm -ivh /mnt/cdrom/RedHat/RPMS/bind*.rpm
```

If an older version of the package is installed and you want to upgrade it to a new version, then use (U):

```
[root@mainserver /]# rpm -Uvh /mnt/cdrom/RedHat/RPMS/bind*.rpm
```

If some version of a package is installed but some files got corrupted, and you want to re-install it then use –force with it while installing.

```
[root@mainserver /]# rpm -ivh –-force /mnt/cdrom/RedHat/RPMS/bind*.rpm
```

Following command erases (removes / un-installs) a specific package.

```
[root@mainserver /]# rpm -e sendmail
```

Sometimes during installation of some rpm package you get some error about missing libraries. To check which package provides you that library, you can use:

```
[root@mainserver /]# rpm -q –-redhatprovides libc.so.6
glibc-2.3.2-11.9
```

So in the above example you see that if you install ***glibc-2.3.2-11.9*** , you will automatically get the required library (***libc.so.6***) , installed. But remember that for the above command to work correctly, you should have ***rpmdb-redhat-a.b*** package already installed on your system.

```
[root@mainserver tmp]# rpm -q rpmdb-redhat
rpmdb-redhat-9-0.20030313
```

System services configuration / device configuration:

To check the status of the daemons listening on different ports:

```
[root@mainserver /]# netstat -at
```

Following command will generate a hardware list of your computer and save it as tar file.

```
[root@mainserver /]# sysreport
```

To configure boot up services use *setup* or *ntsysv*

```
[root@mainserver /]# setup
```

or

```
[root@mainserver /]# ntsysv
```

Assuming you want to add the ***ypbind*** service to run level 3 only; then,

```
[root@mainserver /]# chkconfig --add ypbind
[root@mainserver /]# chkconfig --level 3 ypbind on
```

To configure various hardware devices (prior to Redhat 8.0) , use :

```
[root@mainserver /]# kbdconfig          (for keyboard layout)
[root@mainserver /]# modemtool          (for modem)
[root@mainserver /]# mouseconfig        (for mouse)
[root@mainserver /]# printconf          (for printer configuration, needs X interface)
[root@mainserver /]# timeconfig         (for time zone configuration)
[root@mainserver /]# Xconfigurator (for VGA card)
[root@mainserver /]# xvidtune           (for Monitor display adjustment)
```

To configure various hardware devices (Redhat 8.0 and 9.0) , use :

```
[root@mainserver /]# redhat-config-kbdconfig        (for keyboard layout)
```

```
[root@mainserver /]# redhat-config-modemtool        (for modem)
[root@mainserver /]# redhat-config-mouseconfig      (for mouse)
[root@mainserver /]# redhat-config-printconf        (for printer configuration, needs X interface)
[root@mainserver /]# redhat-config-timeconfig       (for time zone configuration)
[root@mainserver /]# redhat-config-xfree86          (for VGA card)
[root@mainserver /]# xvidtune                       (for Monitor display adjustment)
```

To run a shell terminal from *X*windows, run the command :

```
[root@mainserver /]# xterm
```

or

```
[root@mainserver /]# konsole
```

User Management and file permissions / ownerships:

To add a new group called *mailusers* use:

```
[root@mainserver /]# groupadd mailusers
```

To add new user *kamran* and make him a member of the group *mailusers*, use:

```
[root@mainserver /]# useradd -s /bin/false -g mailusers kamran
```

To change the password if some user, say *kamran*, run the following command as root:

```
[root@mainserver /]# passwd kamran
```

To **lock** the password of user *kamran*, run the following command as root:

```
[root@mainserver /]# passwd -l kamran
```

To **un-lock** the password of user *kamran*, run the following command as root:

```
[root@mainserver /]# passwd -u kamran
```

To delete a user say *kamran*, **including** his home directory, use:

```
[root@mainserver /]# userdel -r kamran
```

To change permissions of an object (file or directory), use chmod:

```
[root@mainserver /]# chmod 770 myfile.txt
```
(changes the permissions of the file to rwx for both owner and group)

or

```
[root@mainserver /]# chmod ug+rwx,o-rwx myfile.txt
```

To change ownership of a directory */project* to *kamran* and set the group permissions to the group *mailusers*, use chown (-R will change ownership of all files and subdirectories *under* the directory *project*:

```
[root@mainserver /]# chown –R kamran:mailusers /project
```

For more topics like sudo, umask and setgid, etc, see the chapter, ***Advance permissions***

Modules management:

To insert a module, eg. you want to add the support for your 3com Ethernet card 3c509 ,then use :

```
[root@mainserver /]# modprobe 3c509 io=0x300 irq=5
```

or

```
[root@mainserver /]# insmod 3c509 io=0x300 irq=5
```

then

```
[root@mainserver /]# ifconfig eth0 192.168.1.5 netmask 255.255.255.0 up
```

To list the currently loaded modules use :

```
[root@mainserver /]# lsmod
```

To remove a module use :

```
[root@mainserver /]# rmmod 3c509
```

Routing table and naming:

To check the routing table :

```
[root@mainserver /]# route
```

or

```
[root@mainserver /]# route -n
```

Note: If the simple `route` command (as shown above) sort of hangs / gives a lot of delay before showing the routing table, and the `route -n` command shows the routing table in numeric form instantly, then this indicates a problem with name resolution.

To add a default route or gateway:

```
[root@mainserver /]# route add default gw  mainserver.mydomain.mydomain.com
```

or

```
[root@mainserver /]# route add default gw  192.168.1.254
```

To delete a default gateway:

```
[root@mainserver /]# route del default
```

To add a route to a network

```
[root@mainserver /]# route add –net 192.168.1.0 netmask 255.255.255.0 dev eth0
```

To delete a network from the routing table

```
[root@mainserver /]# route del –net 192.168.1.0 netmask 255.255.255.0 dev eth0
```

To add a host route:

```
[root@mainserver /]# route add 192.168.1.1 eth0
```

To delete a host route

```
[root@mainserver /]# route del 192.168.1.1
```

To trace the path / number of hops to a particular host on the network:

```
[root@mainserver /]# traceroute wks1.mydomain.com
```

or

```
[root@mainserver /]# traceroute www.linux.org
```

To do tracing of the route and ping at the same time, use *mtr*:

```
[root@mainserver /]# mtr wks1.mydomain.com
```

To find a host and/or it's IP on the network use the following commands. These will query your default DNS server of your network:

```
[root@mainserver /]# host wks1.mydomain.com
```

or

```
[root@mainserver /]# nslookup wks1.mydomain.com
```

or

```
[root@mainserver /]# dig wks1.mydomain.com
```

You can use *hostname* to get the FQDN of the a specific host. But remember that the FQDN hostname must be set in either */etc/sysconfig/network* file or must be set manually using the *hostname* command. It is a good practice to write FQDN of each computer in it's own */etc/sysconfig/network* file. The check

is that the command *hostname --fqdn* must return the FQDN of your machine. See below.

```
[root@mainserver /]# cat /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=mainserver.mydomain.com


[root@mainserver /]# hostname
mainserver.mydomain.com


[root@mainserver /]# hostname --fqdn
mainserver.mydomain.com
```

In order to find out the name of your DNS domain, you should not use *domainname* command without arguments, as it will return you the name of the NIS domain, which in this case is not yet set. You can use ***-d*** switch with *domainname* command or use *dnsdomainname* command. The *domainname -y* also returns you the name of NIS domain (equal to *ypdomainname* command).

```
[root@mainserver /]# domainname
(none)


[root@mainserver /]# domainname -d
mydomain.com


[root@mainserver /]# dnsdomainname
mydomain.com


[root@mainserver /]# domainname -y
(none)


[root@mainserver /]# ypdomainname
(none)
```

To check the status of a wintendo machine on your network, use:

```
[root@mainserver /]# nmblookup -A wks1.mydomain.com
```

or

```
[root@mainserver /]# nmblookup -A 192.168.1.1
```

To check for the open ports on a machine on the network (may be the localhost itself):

```
[root@mainserver /]# nmap wks1.mydomain.com
```

Miscellaneous:

To create a boot floppy out of the images provided by the installation CROM:

```
[root@mainserver /]# dd   if=/mnt/cdrom/images/boot.img of=/dev/fd0
```

To find help on some topic :

```
[root@mainserver /]# man snmp
```

or

```
[root@mainserver /]# man 5 snmp
```

(This is normally referred as snmp(5) in docs)

or

```
[root@mainserver /]# info snmp
```

Also you can seek for a particual keyword (say: ***umask***) in the man pages by:

```
[root@mainserver /]# man -K umask
```

Talking of *info* command, here are some important shortcut keys while navigating in *info* pages.

Key	*Function*
N	Takes you to the next page in sequence after the present one.
P	Takes you to the previous page in sequence after the present one.
U	Takes you up one level in the page hierarchy.
Enter	Jumps to the link under the cursor.
M	Follow link from menu and asks you which entry to use.
F	Follows standard cross reference, which usually has “Note:” written close

Key	*Function*
	to it.
L	Displays the last page shown. Equal to pressing back button on a web browser window.
Space bar	Scroll forward one page.
Backspace	Scroll backward one page.

To find a file in the */usr* directory, use:

```
[root@mainserver /]# find /usr -name “*.doc”
```

To find files and directories owned by the group ***students*** in the */usr* directory, use:

```
[root@mainserver /]# find /usr -group students
```

Assuming a user left the organization and the files owned by his account name must be found and deleted for security reasons, use :

```
[root@mainserver /]# find / -user kamran -exec rm  '{}'  ';'
```

Want to perform some quick mathematical calculations ? okay okay, use:

```
[root@mainserver /]# bc
```

To set up date and time as 20 May 2003 17:35 you should use:

```
[root@mainserver /]# date 0520173503
```

To synchronize it with the hardware clock, use:

```
[root@mainserver /]# hwclock –-systohc
```

To check for memory and swap space usage, you can use:

```
[root@mainserver /]# free
```

or

```
[root@mainserver /]# top
```

To check the system load status averages and the uptime:

```
[root@mainserver /]# uptime
```

To check when the system was booted last time:

```
[root@mainserver /]# who -b
```

To check the status of a particular service:

```
[root@mainserver /]# service sendmail status
```

or

```
[root@mainserver /]# /etc/rc.d/init.d/sendmail status
```

To check the status of all services:

```
[root@mainserver /]# service --status-all
```

To create a boot disk for your system, use the *mkbootdisk* command, but first check the kernel version of your system by *uname -r* command or by looking into the */lib/modules* direcotry:

```
[root@mainserver /]# uname -r
2.4.20-8
```

or

```
[root@mainserver /]# ls /lib/modules
2.4.20-8
```

```
[root@mainserver /]# mkbootdisk --device /dev/fd0 2.4.20-8
```

Sometimes symbolic links to files are broken, eg assuming link file is deleted or some file is over written on it. To create a symbolic link for the */boot/grub/menu.lst* file in the */etc* directory as `grub.conf` use:

```
[root@mainserver /]# ln -s /boot/grub/menu.lst /etc/grub.conf
```

Note: If you ommit *-s* in the *ln* command, a hard link will be created, which will NOT show up in the *ls -l* command output as a link file, rather the newly created hard link will be shown as a regular file. Still it will function as a link file.

If you want to know the RedHat version, read the file
`/etc/redhat-release or /etc/issue`

```
[root@mainserver /]# cat /etc/redhat-release
Red Hat Linux release 9 (Shrike)
```

Similarly

```
[root@mainserver /]# cat /etc/issue
Red Hat Linux release 9 (Shrike)
Kernel \r on an \m
```

Also *uname* can be very helpful to determine your processor model, motherboard class/model and machine architecture. Note: for Intel users the processor may be *i386, i486, i586(P-1)* or *i686(P-2 to P-4)*. Mine is AMD Athlon, so it shows *athlon*. Similarly for Intel based machines or AMD based machines, or any machine known as IBM-PC/AT compatible will show *i386* under *uname -i* command.

```
[root@mainserver /]# uname -p
athlon

[root@mainserver /]# uname -m
i686

[root@mainserver /]# uname -i
i386
```

Important Vi editor commands:

Undo: **u**
Redo: ^r
Repeat: **.**
Visual mode / highlighter mode/ selection mode (used with arrow keys) : **v**
Cut: **"*c**

Copy: **"*y**
Paste: **"*p**
Select all: **ggVG**
Show the name of current file being edited: **Ctrl+g**
Execute shell command: **:!**
Search forward: **/linux**
Search backwards: **?linux**
Search and replace a text in the entire file: **:%s/linux/LINUX/g**

One last thing, tends to be very useful in situations. Assume you changed path using ***cd*** command to some very long/deep level directory, then you want to come back to the previous directory you came from and then from there again you want to come back to the same deep level directory. This creates a lot of typing frustration. Solution is **cd –** command.

```
[root@mainserver network-scripts]# pwd
/etc/sysconfig/network-scripts

[root@mainserver network-scripts]# cd /lib/modules/

[root@mainserver modules]# cd -
/etc/sysconfig/network-scripts


[root@mainserver network-scripts]# cd -
/lib/modules
```

To mount flash (USB) disk use:

```
[root@mainserver modules]# mount /dev/sda1 /mnt/floppy
```

Chapter 2: **Using fdisk**

fdisk is a very powerful tool. And as a system administrator you should know how to use this tool effectively and efficiently without causing harm to data. This tool will save you a lot of time during installation and during normal system operation, both.

To check the partition list, use *fdisk -l*:

```
[root@mainserver /]# fdisk -l

Disk /dev/hda: 255 heads, 63 sectors, 4867 cylinders
Units = cylinders of 16065 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1             1         7     56196   83  Linux
/dev/hda2   *         8      1282  10241437+   c  Win95 FAT32 (LBA)
/dev/hda3          1283      3577  18434587+  83  Linux
/dev/hda4          3578      4867  10361925    5  Extended
/dev/hda5          3578      3705   1028128+  83  Linux
/dev/hda6          3706      3738    265041   83  Linux
/dev/hda7          3739      3771    265041   82  Linux swap
/dev/hda8          3772      4867   8803588+  83  Linux
```

The ***** in B**oot** column shows the active or boot partition, in this case it is `/dev/hda2` or 2nd primary partition on FAT32. The **Id** column contains the partition file system type in hexadecimal format. Also the command will list all the hard disks attached to the system and will display all the partitions in each hard disk.

While creating partitions, it should be noted that at one time there can be a maximum of four primary (with no extended) partitions on one hard drive, numbered 1 to 4. Also there can be only one extended (with max three primary) partition on one hard disk. You can create as many partitions in extended as you want, numbered 5 and above. So in the above example output of the command, `fdisk -l`, you can see three primary partitions, numbered `/dev/hda1` to `/dev/hda3` and an extended partition (considered a special type of primary partition) `/dev/hda4`, defined in hard disk `/dev/hda`, which is the primary master hard disk on this system. This extended partition contains four logical partitions(as many windows users call / identify them), numbered `/dev/hda5` to `/dev/hda8`.

The IDE devices are numbered using the following scheme :

Primary master `/dev/hda`
Primary slave `/dev/hdb`
Secondary master`/dev/hdc`
Secondary slave `/dev/hdd`

To modify partitions use `fdisk` in interactive mode. You have to type in the hard disk device id as a parameter. Type the following command and press ***m*** to get a list of available commands.

```
[root@mainserver /]# fdisk /dev/hdc

The number of cylinders for this disk is set to 4867.
There is nothing wrong with that, but this is larger than 1024, and could in
certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): m
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)

Command (m for help):
```

Scenario:

Assume we want to create some partitions on a new hard disk (`/dev/hdc`), attached to the system as secondary master, of size 2GB. This hard disk also contains some bad sectors somewhere in the middle and at the end. i.e from cylinder 559 till cylinder 579 and from cylinder 611 till cylinder 1023. (This is told to us by someone who has also been using linux with this hard-disk and has given it to us for experimentation. We in turn want to use this on a old 486 computer lying in the corner of our lab.) We want01 to create a primary partition of 100MB (primary 1) which will serve as '**/boot**' later on. This over- comes many boot constraints while installing Linux boot loader: LILO. Then we want to create partitions for win95 and linux. There will be one win95 partition of 100MB (primary 2) and then we will create an extended partition of all leftover space. In this extended partition we will create one partition of 900MB, which will serve as '/' later on; and a 64 MB partition which will serve as a **<swap>** .

Configuration / procedure:

First you check the current partition table by pressing **p** to print the partition table on the screen.

```
Command (m for help):p

Disk /dev/hdc: 64 heads, 63 sectors, 1023 cylinders
Units = cylinders of 4032 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
```

This shows that there are no partitions defined on the current hard disk. Now to add a new partition use ***n*** and fdisk will ask you for the type of partition, **primary** or **extended**. Type ***p*** for primary and give ***1*** as partition number and ***+100M*** as size.

```
Command (m for help):n

Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-1023, default 1): 1
Last cylinder or +sizeM or +sizeK (1-1023, default 1023): +100M
```

Now we can check the partition table by pressing p again at the fdisk's command prompt.

```
Command (m for help): p

Disk /dev/hdc: 64 heads, 63 sectors, 1023 cylinders
Units = cylinders of 4032 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hdc1             1        51    102815+  83  Linux
```

Now we want to add a win95 primary partition (FAT32). Again add a new primary partition by pressing ***p*** and press ***2*** for partition number. Press enter for default starting address and specify ***+100M*** for the size.

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (52-1023, default 52):(press Enter)
Using default value 52
Last cylinder or +sizeM or +sizeK (52-1023, default 1023): +100M

Command (m for help): p

Disk /dev/hdc: 64 heads, 63 sectors, 1023 cylinders
Units = cylinders of 4032 * 512 bytes
```

```
   Device Boot    Start       End    Blocks   Id  System
/dev/hdc1             1        51    102815+  83  Linux
/dev/hdc2            52       102    102816   83  Linux
```

If you notice here we have created another primary partition of size 100MB and have also confirmed the effects by printing the partition table. But there is something confusing here, under the **Id** column fdisk is showing the partition as type 83, which is of file system type **Linux**. But we wanted to make it of type FAT32 ! No problem, we will change the system id of this partition by pressing ***t*** on the fdisk command prompt and specify that we want to change the system id of partition number ***2***, to ***win95 FAT32***, which is of type ***c***.

```
Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes): L

 0  Empty           1c  Hidden Win95 FA 70  DiskSecure Mult bb  Boot Wizard hid
 1  FAT12           1e  Hidden Win95 FA 75  PC/IX           be  Solaris boot
 2  XENIX root      24  NEC DOS         80  Old Minix       c1  DRDOS/sec (FAT-
 3  XENIX usr       39  Plan 9          81  Minix / old Lin c4  DRDOS/sec (FAT-
 4  FAT16 <32M      3c  PartitionMagic  82  Linux swap      c6  DRDOS/sec (FAT-
 5  Extended        40  Venix 80286     83  Linux           c7  Syrinx
 6  FAT16           41  PPC PReP Boot   84  OS/2 hidden C:  da  Non-FS data
 7  HPFS/NTFS       42  SFS             85  Linux extended  db  CP/M / CTOS / .
 8  AIX             4d  QNX4.x          86  NTFS volume set de  Dell Utility
 9  AIX bootable    4e  QNX4.x 2nd part 87  NTFS volume set df  BootIt
 a  OS/2 Boot Manag 4f  QNX4.x 3rd part 8e  Linux LVM       e1  DOS access
 b  Win95 FAT32     50  OnTrack DM      93  Amoeba          e3  DOS R/O
 c  Win95 FAT32 (LB 51  OnTrack DM6 Aux 94  Amoeba BBT      e4  SpeedStor
 e  Win95 FAT16 (LB 52  CP/M            9f  BSD/OS          eb  BeOS fs
 f  Win95 Ext'd (LB 53  OnTrack DM6 Aux a0  IBM Thinkpad hi ee  EFI GPT
10  OPUS            54  OnTrackDM6      a5  FreeBSD         ef  EFI (FAT-12/16/
11  Hidden FAT12    55  EZ-Drive        a6  OpenBSD         f0  Linux/PA-RISC b
12  Compaq diagnost 56  Golden Bow      a7  NeXTSTEP        f1  SpeedStor
14  Hidden FAT16 <3 5c  Priam Edisk     a8  Darwin UFS      f4  SpeedStor
16  Hidden FAT16    61  SpeedStor       a9  NetBSD          f2  DOS secondary
17  Hidden HPFS/NTF 63  GNU HURD or Sys ab  Darwin boot     fd  Linux raid auto
18  AST SmartSleep  64  Novell Netware  b7  BSDI fs         fe  LANstep
1b  Hidden Win95 FA 65  Novell Netware  b8  BSDI swap       ff  BBT

Hex code (type L to list codes):c
Changed system type of partition 2 to c (Win95 FAT32 (LBA))

Command (m for help): p

Disk /dev/hdc: 64 heads, 63 sectors, 1023 cylinders
Units = cylinders of 4032 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hdc1             1        51    102815+  83  Linux
/dev/hdc2            52       102    102816    c  Win95 FAT32 (LBA)

Command (m for help):
```

As you can see, we have verified that the partition id of partition 2 has been successfully changed to Win95 FAT32. Now we also want that our partition 2 should be the boot partition. So we will use a to toggle the boot flag of partition number 2.

```
Command (m for help): a
Partition number (1-4): 2

Command (m for help): p

Disk /dev/hdc: 64 heads, 63 sectors, 1023 cylinders
Units = cylinders of 4032 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hdc1             1        51    102815+  83  Linux
/dev/hdc2   *        52       102    102816    c  Win95 FAT32 (LBA)
```

Now we want to create an extended partition of all available leftover space, and within that we will be creating the partitions which will be used by linux later on during installation. We can also leave it till here as installation program also contains tools like ***Disk Druid*** and ***fdisk***, but lets go on with it for now.

```
Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
e
Partition number (1-4): 3
First cylinder (103-1023, default 103):(press Enter)
Using default value 103
Last cylinder or +sizeM or +sizeK (103-1023, default 1023): (press Enter)
Using default value 1023


Command (m for help): p

Disk /dev/hdc: 64 heads, 63 sectors, 1023 cylinders
Units = cylinders of 4032 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hdc1             1        51    102815+  83  Linux
/dev/hdc2   *        52       102    102816    c  Win95 FAT32 (LBA)
/dev/hdc3           103      1023   1856736    5  Extended


Command (m for help): n
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
l
First cylinder (103-1023, default 103): (Press Enter)
Using default value 103
Last cylinder or +sizeM or +sizeK (103-1023, default 1023): 559
```

Notice that this time after pressing ***n*** for new partition , the options menu is changed. There is ***logical*** in place of ***extended***. This is because there can only be one extended partition in one hard disk, and once that is created, it's option is no more shown. We are allowed to create logical partitions in the extended partition. In the above example we have created a cylinder based partition, from cylinder 103 till cylinder

559. We will create the next partition from cylinder 579 till cylinder 611, which will serve as our swap partition, after-wards.

```
Command (m for help): n
Command action
   l   logical (5 or over)
   p   primary partition (1-4)
l
First cylinder (560-1023, default 560): 579
Last cylinder or +sizeM or +sizeK (579-1023, default 1023): 611


Command (m for help): p

Disk /dev/hdc: 64 heads, 63 sectors, 1023 cylinders
Units = cylinders of 4032 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hdc1             1        51    102815+  83  Linux
/dev/hdc2   *        52       102    102816    c  Win95 FAT32 (LBA)
/dev/hdc3           103      1023   1856736    5  Extended
/dev/hdc5           103       559    921311+  83  Linux
/dev/hdc6           579       611     66528   83  Linux
```

Since this partition has to be our swap partition, we need to change it's system id to **82**, which is **Linux swap**.

```
Command (m for help): t
Partition number (1-6): 6
Hex code (type L to list codes):82
Changed system type of partition 6 to 82 (Linux swap)

Command (m for help): p

Disk /dev/hdc: 64 heads, 63 sectors, 1023 cylinders
Units = cylinders of 4032 * 512 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hdc1             1        51    102815+  83  Linux
/dev/hdc2   *        52       102    102816    c  Win95 FAT32 (LBA)
/dev/hdc3           103      1023   1856736    5  Extended
/dev/hdc5           103       559    921311+  83  Linux
/dev/hdc6           579       611     66528   82  Linux swap
```

All the partitions created, now let's save our efforts, i.e write the partition table by using ***w*** on the fdisk prompt. If ***fdisk*** was called from a command prompt, then after pressing ***w,*** we will be taken back to the shell command prompt. And if we called ***fdisk*** during partitioning option of Linux installation, then we will be taken back to the previous screen where you are to select hard disk to run ***fdisk*** on. Go more back from there and select Disk druid this time to assign mount points and format the partitions during installation.

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

```
[root@mainserver /]#
```

Oh, and by the way, if at any time you think that you have messed up things then just come out of ***fdisk*** by pressing ***q*** instead of ***w***. This will quit ***fdisk*** without saving / writing the partition table. You can then re-run it. This normally happens when you accidentally delete a partition or assign incorrect system id or incorrect size, etc. Also deleting a partition is very simple, just press ***d*** on the ***fdisk*** command prompt, you will be asked for the partition number from ***1*** to maximum number of partitions you have on the drive. Save your changes by pressing ***w*** on the ***fdisk*** command prompt.

Partition size calculation:

Now lets calculate the sizes of these cylinder based partitions on the basis of the information available from the print partition table command, ***p***.

The statement **`Units = cylinders of 4032 * 512 bytes ,`** in the output generated by print (***p***) command of ***fdisk***, means that each cylinder consists of 4032 * 512 bytes, which is equal to 2064384 bytes. We have (51-1)+1 = 51 cylinders in `/dev/hdc1` partition. So 51 * 2064384 bytes = 105283584 bytes. Divide this number by 1024 which gives us 102816 KB (kilo bytes). Further divide this new number by 1024 again which gives us 100.40625 MB (mega bytes). This was in fact our size which we specified while creating our `/dev/hdc1` partition. Now since this method for finding out the size of partition is confirmed, lets find out the size of the partitions `/dev/hdc3`, `/dev/hdc5` and `/dev/hdc6`, which were created using cylinder numbers:

1)/dev/hdc3 (extended partition)

```
(1023-103)+1 = 921 cylinders
921 * 4032 * 512 = 1901297664 bytes
1901297664 / 1024 = 1856736 KB
1856736 /1024 = 1813.21875 MB
1813.21875 / 1024 = 1.770721436 GB
```
, which is nearly equal to 1.8 GB

2)/dev/hdc5

```
(559-103)+1 = 457 cylinders
```

```
457 * 4032 * 512 = 941359104 bytes
941359104 / 1024 = 919296 KB
919296 / 1024 = 897.75 MB, which is nearly equal to  900 MB.
```

3)/dev/hdc6

```
(611-579) + 1 = 33 cylinders
33 * 4032 * 512 = 68124672 bytes
68124672 / 1024 = 66528 KB
66528 / 1204 = 64.96875 MB, which is nearly equal to 65 MB
```

Now we know that the total number of cylinders on this hard disk were 1023, out of which we have used up to 611. so how much space is still left on this hard disk, (this space in fact contains a lot of bad sectors).

```
(1023 - 611) +1 = 413 cylinders
413 * 4032 * 512 = 852590592 bytes
852590592 / 1024 = 832608 KB
832608 / 1024 = 813.09375 MB, which is nearly equal to 813 MB
```

Now you have created the partitions, you can use the command ***mke2fs*** to format the new Linux partition. You have to use ***mkswap*** to format the swap partition and ***mkdosfs*** to format the FAT32 partition.

```
[root@mainserver /]# mke2fs -c /dev/hdc1

mke2fs 1.27 (8-Mar-2002)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
25792 inodes, 102815 blocks
5140 blocks (5.00%) reserved for the super user
First data block=1
13 block groups
8192 blocks per group, 8192 fragments per group
1984 inodes per group
Superblock backups stored on blocks:
        8193, 24577, 40961, 57345, 73729

Checking for bad blocks (read-only test): done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 38 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

```
[root@mainserver /]# mkdosfs -c -F 32 /dev/hdc2
mkdosfs 2.8 (28 Feb 2001)

[root@mainserver /]# mkswap /dev/hdc6
Setting up swapspace version 1, size = 133052 KiB
```

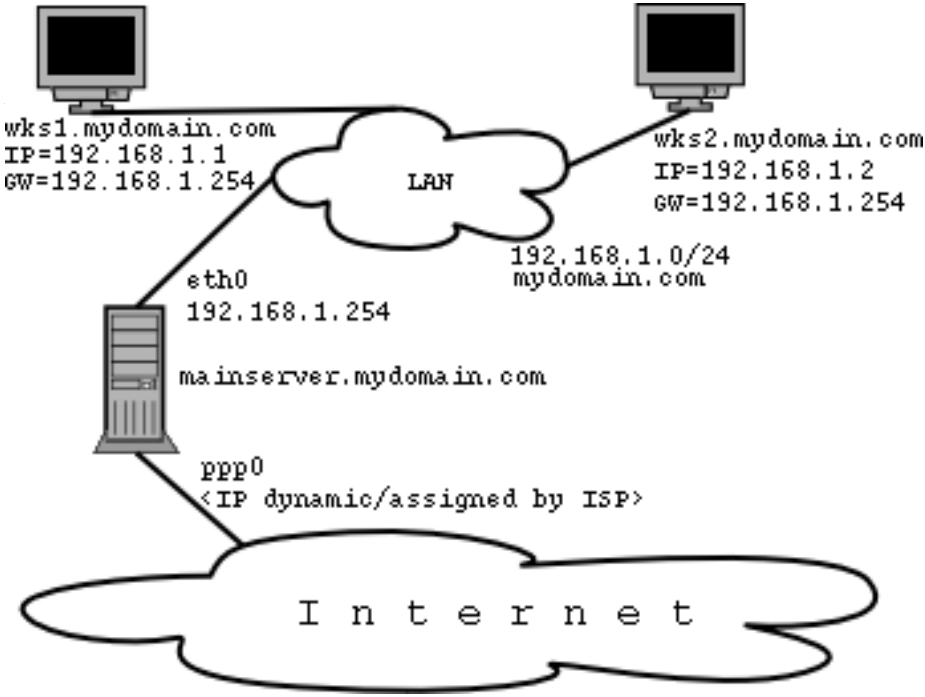
You further need to turn on swap space usage by using ***swapon*** command.

```
[root@mainserver /]# swapon -a
```

Chapter 3: **Basic Network trouble-shooting**

Example network:

Our example network is a class C private IP scheme `192.168.1.0` network, which is of-course non-route able. The network has a domain named **mydomain.com** which is not registered on the Internet and serves as the part of fully qualified domain names of the machines of this network only. There is a modem attached with the server to connect to Internet, so this server also acts as a gateway for this network. This network will be used in all examples of this book, until/unless specified otherwise. *See Fig 1*.



Network: `192.168.1.0`
Network Mask: `255.255.255.0`

Server is called **mainserver** with fully qualified domain name (FQDN) as **mainserver.mydomain.com** It's IP is **`192.168.1.254`** and is the default gateway for the client computers on this network.

Client machines are called **wks1, wks2** and onwards , with FQDNs as **wks1.mydomain.com** and **wks2.mydomain.com** respectively, having IPs **192.168.1.1** and **192.168.1.2** and onwards, respectively.

Basic network configuration and troubleshooting

Check the network settings in the following order:-

Note: All commands are to be given as user ***`root`***.

1)Check the hostname of the computer you are sitting on, and set it accordingly of not correct. For run-time you can set the hostname by using the hostname command again as :

```
[root@mainserver /]# hostname
mainserver

[root@mainserver /]# hostname mainserver.mydomain.com
```

2) Check name resolution

`[root@mainserver /]# vi /etc/hosts` (*client, server both*)

In case DNS is being used for name resolution, edit this file and remove any IP and names of computers which are part of this domain / network. For the machine `mainserver`, you can leave only the following line in it:

```
127.0.0.1        localhost.localdomain     localhost
192.168.1.254    mainserver.mydomain.com   mainserver
```

(interpret this file as ***IP*** then *machine name* then ***fully qualified domain name*** then ***alias* or *nickname***)

In case DNS is ***not*** being used for name resolution, edit this file and put in IP and names of computers which are part of this domain / network. Remember!, In such a case this file needs to be replicated on all the *nix based machines on this network.

(Quite a lot, isn't it !). For the machine **mainserver** and **wks1**, you can have the following lines in this file:

```
127.0.0.1         localhost.localdomain     localhost
192.168.1.254     mainserver.mydomain.com   mainserver
192.168.1.1       wks1.mydomain.com         wks1
```

3) Check for name resolution order

[root@mainserver /]# vi /etc/host.conf *(client, server both)*

In case DNS is being used primarily for name resolution, edit this file and it should have at least the following line:

```
order bind, hosts
```

In case DNS is ***not*** being used primarily for name resolution, rather local `/etc/hosts` file is the primary name resolver, then edit this file and it should have at least the following line:

```
order hosts, bind
```

Sometimes you will see a ***multi on*** line in this file. This means that any particular host in question can have more than one IP addresses in the DNS file. This is the case for example in multi-homed computers, which have more then one network interfaces, like firewall machines or routers, etc.

4) Who is the name server for each machine?

[root@mainserver /]# vi /etc/resolv.conf *(client and server both)*

In case you are on the server machine, and the same machine (i.e. the server) is the DNS server as well, then edit this file and should contain the following:

```
nameserver       127.0.0.1
```

In case you are on the client machine, and some other machine (server) is the DNS server, then edit this file and should contain the following:

```
nameserver       192.168.1.254
```

Note: you can have multiple *nameserver* lines in a /`etc/resolv.conf` file. Like :

```
nameserver       127.0.0.1
nameserver       192.168.1.254
nameserver       192.168.1.121
```

5) Check for *boot time* host name and gateway settings :

[root@mainserver /]# vi /etc/sysconfig/network *(client and server both)*

This file contains your hostname so in case of server it should contain (the file on the server should look like this):

```
NETWORKING=yes
HOSTNAME=mainserver.mydomain.com
```

And the file on the clients, must have a *GATEWAY* mentioned in it:

```
NETWORKING=yes
HOSTNAME=wks1.mydomain.com
GATEWAY=192.168.1.254
```

You can also configure the hostname temporarily by using the command:

[root@mainserver /]# hostname mainserver.mydomain.com

6) Check IP address:

[root@mainserver /]# vi /etc/sysconfig/network-scripts/ifcfg-eth0 *(client and server both)*

This file contains your IP, which will be configured on machine boot up time. So it should contain a minimum of your IP address , network mask and network address, broadcast address, etc.

```
DEVICE=eth0
BOOTPROTO=static
IPADDR=192.168.1.254
NETMASK=255.255.255.0
ONBOOT=yes
```

7) Now restart the network if required, by following commands or just restart your computer:

[root@mainserver /]# /etc/rc.d/init.d/network restart *(client and server both)*

or

[root@mainserver /]# service network restart *(client and server both)*

Check the output of *ifconfig* command after restarting the network. If it doesn't show the new IP which you just gave in the configuration files then this means that your computer is picking up IP from somewhere else. But from where ? The answer is "most probably" from the `/etc/sysconfig/networking/devices/ifcfg-eth0` file. Actually on some Linux installations or in some situations, the `/etc/sysconfig/network-scripts/ifcfg-eth0` file is a **symbolic link** to `/etc/sysconfig/networking/devices/ifcfg-eth0` file. If this link breaks, then weird things happen. Like, you edit the `network-scripts/ifcfg-eth0` file but your system picks up the old IP from no where (actually from `networking/devices/ifcfg-eth0` file). What to do ? Heal the broken hearts. How ? Well, change directory to `/etc/sysconfig/network-scripts`. Copy the `ifcfg-eth0` file, which you think is the correct one, to the `/etc/sysconfig/networking/devices` directory. Or the contrary if you think that the file in `networking/devices/ifcfg-eth0` file is the one that should be loaded then delete any `ifcfg-eth0` file from the `/etc/sysconfig/network-scripts` directory. Then create the symbolic link by :

```
[root@mainserver /]# cd /etc/sysconfig/network-scripts/

[root@mainserver network-scripts]# ln -s ../networking/devices/ifcfg-eth0
```

or

```
[root@mainserver /]# ln -s /etc/sysconfig/networking/devices/ifcfg-eth0
        ➥ /etc/sysconfig/network-scripts/ifcfg-eth0
```

Now restart the network again and check the results by *`ifconfig`* command. This time it should be ok.

Then issue the *`hostname`*, *`ifconfig`* and *`route`* commands to check the changes:

```
[root@mainserver /]# ifconfig

[root@mainserver /]# route

[root@mainserver /]# hostname
```

If *ifconfig* doesn't show the desired IP, then try assign the IP manually for the time being by the command:

```
[root@wks1 /]# ifconfig eth0 192.168.1.254 netmask 255.255.255.0 up
```

If *ifconfig* command gives any errors like ***FILE EXISTS*** , etc, then stop the network first, configure IP by *ifconfig* command and then again start the network:

```
[root@wks1 /]# service network stop
[root@wks1 /]# ifconfig eth0 192.168.1.1 netmask 255.255.255.0 up
```

This time the *ifconfig* command (without any other parameters), should show you the new IP. The route command should show a default gateway in the routing table only on client computers. It should ***not*** show a default gateway for the server, which is not yet connected to any other network.

Remember that your ***loopback*** interface should also be "***up***" for your computer to work correctly. You can check this by ifconfig command without any parameters.

```
[root@wks1 /]# ifconfig
```

And you can turn it up by :

```
[root@wks1 /]# ifup lo
```

or

```
[root@wks1 /]# ifconfig lo 127.0.0.1 netmask 255.0.0.0 up
```

At this point you should be able to ping your own computer:

```
[root@wks1 /]# ping 127.0.0.1
```

```
[root@wks1 /]# ping 192.168.1.1
```

If this is correct and the above two commands give you echo reply, then you should also be able to *ping* gateway IP.

```
[root@wks1 /]# ping 192.168.1.254
```

If you can *ping* gateway IP , then it means you should be able to ping any other computer in the network.

If other computers are also configured then you should be able to ping them with their respective IPs. Ping on the name e.g. *mainserver.mydomain.com* or *wks1.mydomain.com* will work only in that case when name resolution is working properly, either through *hosts* files or DNS is configured on the network server and is running.

```
[root@mainserver /]# ping mainserver.mydomain.com
[root@mainserver /]# ping wks1.mydomain.com
```

Note: If ping or any other service like *telnet* or *ssh* complaints that "***Destination Host Unreachable"*** then check the cable connection and IP addresses on both client and server. Also check for firewall blocking ICMP packets. If it complaints with "***Unknown host"*** then there is problem with the name resolution. Check hosts file if DNS is not running or not configured. Check */etc/resolv.conf* and other files if DNS is configured. If ping or any other service complaints with "***Network Unreachable"*** then your computer is not on the same network as the target computer.

Further checking during network trouble shooting:

1)Is the network wiring / cable connections / cable configurations correct ?
You have to be sure that the cabling schemes / wiring is correct and according to some standard. For this you can consult common wiring layouts and cable configurations attached at the end of this book as ***Appendix A***.

2)Is the software package(s) installed ?
It may happen that during a custom or workstation install you forgot to install the particular packages required by the service. You can use **rpm** commands to check for the existence of a package. For example you are trying to establish a telnet or secure shell session with another machine and you get a "*connection refused*" or "*destination host unreachable*" type of errors. The first error means that the telnet service is not running on the target computer. It is either not installed at all or is stopped or is being blocked by some firewall. The second error shows problem with your name resolution or physical connection problem, means

2) Does the Firewall permit access to the service?
Normally during install of Redhat 7.2 , 7.3, 8 and 9, you are asked for a firewall. You might have selected "*High*" or "*Medium*" firewall security option. In this case most of the services will be blocked. The details are:

In case of ***High security*** the following will happen:
- No new connections will be allowed targeted to your machines
- Only DNS and DHCP replies will be allowed by this machine if DNS server (*named*) and DHCP (*dhcpd*) are running on this machine
- Active mode FTP to this machine is not allowed
- IRC DCC file transfers are not allowed
- Real Audio and Remote X windows client sessions are not allowed
- NIS & LDAP replies are not allowed

In case of ***Medium security*** the following will happen:
- Access to ports below 1023 will not be allowed
- NFS and X Font Server and Remote X windows client sessions are not allowed

Note: On a RedHat system, the firewall script generated by *firewall-config* or *redhat-config-securitylevel* or *Gnome Lokkit*, is stored as /etc/sysconfig/iptables .

To temporarily disable a (default) firewall on a redhat 8 and above systems, use :

```
[root@mainserver /]# iptables -F
[root@mainserver /]# iptables -t nat -F
```

On a RedHat 7.2 system, in which *ipchains* is the default software for firewall implementation, use :

```
[root@mainserver /]# ipchains -F
```

Also if you want to use *iptables* instead of *ipchains* on a RedHat 7.2 system, you have to remove / unload the *ipchains* module:

```
[root@mainserver /]# rmmod ipchains
```

3) Is the startup script set up correctly to automatically launch the service?
Normally you can check this by simply using the ***setup*** or ***ntsysv*** or ***chkconfig*** command or the ***chkconfig*** command:

```
[root@mainserver /]# ntsysv
```

or

```
[root@mainserver /]# setup
```

or

```
[root@mainserver /]# chkconfig --list | less
```

And if a service (say ***sendmail***) is not configured to start automatically for a run-level (say run-level 3), you can do so by:

```
[root@mainserver /]# chkconfig --level 3 sendmail on
```

4) Is the configuration file created for the service?
Service might be running but it's related configuration files may not be configured properly. eg. For *sendmail*, the main configuration file is */etc/mail/sendmail.cf* .

5) Does the configuration file permit proper access to the service?
This is sometimes called service level access control / firewall. In this case some computers / addresses might be restricted not to access the service. For example for ***sendmail***, the access file (normally) is `/etc/mail/access` , `/etc/mail/relay-domains`.

6) Are there any other restrictions to the service being shared?
Some times standard Linux security features may block access to a particular service which might otherwise be configured and running. e.g. NFS is running in properly configured state but the local file permissions are not allowing users to access the files made available through NFS. Also name resolution, specifically missing reverse lookup PTR entries in the reverse zone file to the corresponding A type entries in the forward zone file of DNS sometimes results in denial of access.

Chapter 4: DNS

BIND (Berkley Internet Naming Daemon) in *nix is a tool to implement DNS (Domain Name Service). The following procedure should be adapted for implementation and troubleshooting of BIND.

Packages:

For BIND the following packages need to be installed on the server:

caching-nameserver-a.b
bind-a.b
bind-utils-a.b

Where ***a*** and ***b*** represent major and minor version numbers respectively. No special packages on the client is required to be a part of DNS.

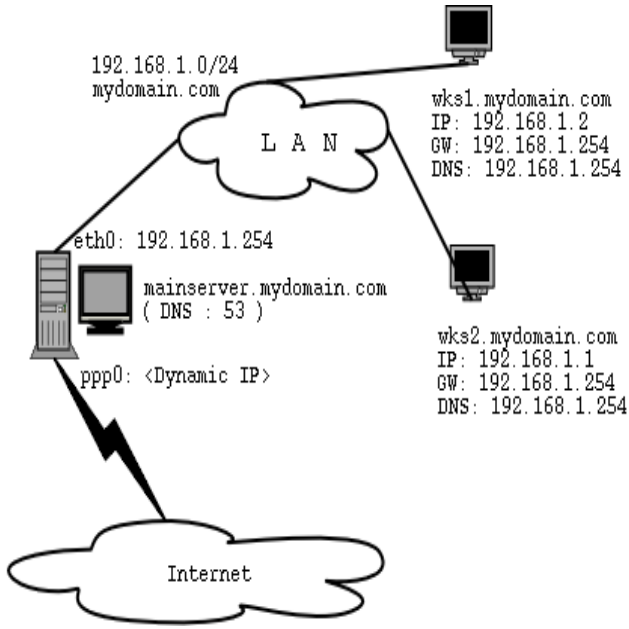


Fig 1

Scenario:

In the example network above, we need to implement DNS. The name of the domain will be ***`mydomain.com`***. See figure.

Configuration:

The following files need to be present only on the server, before starting name daemon:

`/etc/named.conf`
`/var/named/localhost.zone` *(optional)*
`/var/named/0.0.127.in-addr.arpa.zone`
`/var/named/mydomain.com.zone`
`/var/named/1.168.192.in-addr.arpa.zone`
`/var/named/named.ca` (*this file should exist even if zero byte in size*)

And amongst them, the following will need configuration:

`/etc/named.conf`
`/var/named/mydomain.com.zone`
`/var/named/1.168.192.in-addr.arpa.zone`

1) First edit the `/etc/named.conf` file.

```
[root@mainserver /]# vi /etc/named.conf

## named.conf - configuration for bind
# # Generated automatically by bindconf, alchemist et al.
controls {
inet 127.0.0.1 allow { localhost; }
keys { rndckey; };
};

include "/etc/rndc.key";

options {
        directory "/var/named/";
};


zone "." {
        type hint;
        file "named.ca";
};


zone "0.0.127.in-addr.arpa" {
type master;
file "0.0.127.in-addr.arpa.zone";
```

```
allow-update { none; };
};

zone "1.168.192.in-addr.arpa" {
type master;
file "1.168.192.in-addr.arpa.zone";
allow-update { none; };
};


zone "localhost" {
type master;
file "localhost.zone";
allow-update { none; };
};

zone "mydomain.com" {
type master;
file "mydomain.com.zone";
allow-update { none; };
};
```

Note: Either in the options section (which has global effect), or in the zone section you can further tighten or loosen your security. Like you can use ***allow-update*** *{ 192.168.1.0/24;};* line to allow DNS table updates by the clients of your example network. Helpful to DHCP servers and windows 2000 clients. Similarly you can use ***allow-query*** *{ 192.168.1.0/24; };* to restrict the DNS queries to be asked by the clients of your example network only.
And you can restrict the list of all of your machines (zone transfer) to the clients of your network only by ***allow-transfer*** *{ 192.168.1.0/24; };*

Note: If you ever happen to mess up with this file, just re-install *caching-nameserver* package and things should be fine.

2) Now create forward zone `/var/named/mydomain.com.zone` file for your domain:

```
[root@mainserver /]# vi /var/named/mydomain.com.zone

$TTL 86400
$ORIGIN mydomain.com.
@     IN  SOA      mainserver.mydomain.com. mkazeem.isb.paknet.com.pk.(
          1 ; serial
          28800 ; refresh
          7200 ; retry
          604800 ; expire
          86400 ; ttl
)

@            IN   NS       mainserver.mydomain.com.
mainserver    IN   A        192.168.1.254
wks1         IN   A        192.168.1.1
wks2         IN   A        192.168.1.2
www          IN   CNAME    mainserver
```

3) Next create the reverse zone file for your network:

```
[root@mainserver /]# vi /var/named/1.168.192.in-addr.arpa.zone

$TTL 86400
$ORIGIN 1.168.192.in-addr.arpa.
@   IN   SOA      mainserver.mydomain.com.  mkazeem.isb.paknet.com.pk.(
         1 ; serial
         28800 ; refresh
         7200 ; retry
         604800 ; expire
         86400 ; ttl
)

@        IN       NS       mainserver.mydomain.com.
1        IN       PTR      wks1.mydomain.com.
2        IN       PTR      wks2.mydomain.com.
254      IN       PTR      mainserver.mydomain.com.
```

Note: There are no corresponding reverse entries in the reverse zone file for *CNAME* entries in forward zone file.

Also note that you can write the PTR lines as :

```
1.1.168.192.in-addr.arpa.          IN PTR   wks1.mydomain.com.
2.1.168.192.in-addr.arpa.          IN PTR   wks2.mydomain.com.
254.1.168.192.in-addr.arpa.        IN PTR   mainserver.mydomain.com.
```

4) Check existence for `/var/named/named.ca` file.

This file contains the Internet addresses of the root servers (currently 13) in the world. So nothing to be messed with in this file. It looks like :

```
[root@mainserver /]# cat /var/named/named.ca

.        518400  IN      NS      K.ROOT-SERVERS.NET.
.          518400  IN      NS      L.ROOT-SERVERS.NET.
.          518400  IN      NS      M.ROOT-SERVERS.NET.
.          518400  IN      NS      I.ROOT-SERVERS.NET.
.          518400  IN      NS      E.ROOT-SERVERS.NET.
.          518400  IN      NS      D.ROOT-SERVERS.NET.
.          518400  IN      NS      A.ROOT-SERVERS.NET.
.          518400  IN      NS      H.ROOT-SERVERS.NET.
.          518400  IN      NS      C.ROOT-SERVERS.NET.
.          518400  IN      NS      G.ROOT-SERVERS.NET.
.          518400  IN      NS      F.ROOT-SERVERS.NET.
.          518400  IN      NS      B.ROOT-SERVERS.NET.
.          518400  IN      NS      J.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.     3600000 IN      A       193.0.14.129
L.ROOT-SERVERS.NET.     3600000 IN      A       198.32.64.12
M.ROOT-SERVERS.NET.     3600000 IN      A       202.12.27.33
I.ROOT-SERVERS.NET.     3600000 IN      A       192.36.148.17
E.ROOT-SERVERS.NET.     3600000 IN      A       192.203.230.10
D.ROOT-SERVERS.NET.     3600000 IN      A       128.8.10.90
A.ROOT-SERVERS.NET.     3600000 IN      A       198.41.0.4
H.ROOT-SERVERS.NET.     3600000 IN      A       128.63.2.53
```

```
C.ROOT-SERVERS.NET.     3600000 IN      A       192.33.4.12
G.ROOT-SERVERS.NET.     3600000 IN      A       192.112.36.4
F.ROOT-SERVERS.NET.     3600000 IN      A       192.5.5.241
B.ROOT-SERVERS.NET.     3600000 IN      A       128.9.0.107
J.ROOT-SERVERS.NET.     3600000 IN      A       192.58.128.30
```

Actually , initially this file has the same contents as shown above, but in a different layout. This file shown above was derived / created / updated using :

```
[root@mainserver /]# dig NS . @a.root-servers.net > /var/named/named.ca
```

You should update this file regularly on your system running name server, by issuing the above command on periodic basis.

5) Update your `/etc/resolv.conf` file , so that your machine where the name server is. Further, this file */etc/resolv.conf* , needs to be configured / adjusted on the server and *all* *nix clients, before starting *name* daemon. It looks like:

```
[root@mainserver /]# vi /etc/resolv.conf

nameserver 127.0.0.1
nameserver 192.168.1.254
```

or

```
domain mydomain.com
nameserver 127.0.0.1
nameserver 192.168.1.254
```

or

```
search mydomain.com
nameserver 127.0.0.1
nameserver 192.168.1.254
```

And on the client, it might / should look like:

```
domain mydomain.com
nameserver 192.168.1.254
```

or

```
search mydomain.com
nameserver 192.168.1.254
```

Note: both ***search*** and ***domain*** keywords will not work at the same time if mentioned together in the same file.

Now the configuration finishes here. Start or restart your name daemon by:

```
[root@mainserver /]# /etc/rc.d/init.d/named start
```

or

```
[root@mainserver /]# /etc/rc.d/init.d/named restart
```

or

```
[root@mainserver /]# /etc/rc.d/init.d/named reload
```

and check it's status by :

```
[root@mainserver /]# /etc/rc.d/init.d/named status
```

Remember that once you change BIND configuration files, you can / should reload it's database by:

```
[root@mainserver /]# /etc/rc.d/init.d/named reload
```

Log Files:

The following log file can be checked on the server for detecting any problems relating to BIND.

`/var/log/messages`

Documentation and Help:

Extensive Documentation is available in */usr/share/doc/bind-** and on *www.isc.org*

Testing :

You can use *nslookup* for your testing (non interactive mode):

```
[root@mainserver /]# nslookup mydomain.com
Note:  nslookup is deprecated and may be removed from future releases.
Consider using the `dig' or `host' programs instead.  Run nslookup with
the `-sil[ent]' option to prevent this message from appearing.
Server:         192.168.1.254
Address:        192.168.1.254#53

Name:   mydomain.com
Address: 192.168.1.254
```

Or start *nslookup* in interactive mode :

```
[root@mainserver /]# nslookup
Note:  nslookup is deprecated and may be removed from future releases.
Consider using the `dig' or `host' programs instead.  Run nslookup with
the `-sil[ent]' option to prevent this message from appearing.
> mydomain.com
Server:         192.168.1.254
Address:        192.168.1.254#53

Name:   mydomain.com
Address: 192.168.1.254

> set type=NS
> mydomain.com
Server:         192.168.1.254
Address:        192.168.1.254#53

mydomain.com nameserver = mainserver.mydomain.com.

> mainserver      (Note here: no NameServer defined inside mainserver)
Server:         192.168.1.254
Address:        192.168.1.254#53

*** Can't find mainserver: No answer

> set type=MX
> mydomain.com
Server:         192.168.1.254
Address:        192.168.1.254#53

mydomain.com  mail exchanger = 10 mainserver.

> set type=ANY
> mydomain.com
Server:         192.168.1.254
Address:        192.168.1.254#53

Name:   mydomain.com
Address: 192.168.1.254
mydomain.com
        origin = mainserver.mydomain.com.
        mail addr = root.mainserver.mydomain.com.
        serial = 1
        refresh = 28800
        retry = 7200
        expire = 604800
```

```
        minimum = 86400
mydomain.com  nameserver = mainserver.mydomain.com.
mydomain.com  mail exchanger = 10 mainserver.mydomain.com.

> set type=A
> mydomain.com
Server:         192.168.1.254
Address:        192.168.1.254#53

Name:   mydomain.com
Address: 192.168.1.254

> mainserver
Server:         192.168.1.254
Address:        192.168.1.254#53

Name:   mainserver.mydomain.com
Address: 192.168.1.254

> wks1
Server:         192.168.1.254
Address:        192.168.1.254#53

Name:   wks1.mydomain.com
Address: 192.168.1.1

> wks2
Server:         192.168.1.254
Address:        192.168.1.254#53

Name:   wks2.mydomain.com
Address: 192.168.1.2

> bigboss         (doesn't exist)
;; connection timed out; no servers could be reached

> exit
```

You can also run *dig:*

```
[root@mainserver /]# dig mydomain.com

; <<>> DiG 9.1.3 <<>> mydomain.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64144
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;mydomain.com.          IN      A

;; ANSWER SECTION:
mydomain.com.  86400   IN  A   192.168.1.254

;; AUTHORITY SECTION:
mydomain.com. 86400       IN       NS       mainserver.mydomain.com.

;; Query time: 17 msec
;; SERVER: 192.168.1.254#53(192.168.1.254)
;; WHEN: Fri Jul 19 14:55:35 2002
;; MSG SIZE  rcvd: 69
```

```
[root@mainserver /]# dig mainserver.mydomain.com
; <<>> DiG 9.2.1 <<>> mainserver.mydomain.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 43728
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;mainserver.mydomain.com.    IN      A

;; ANSWER SECTION:
mainserver.mydomain.com. 10800    IN       A       192.168.1.254

;; AUTHORITY SECTION:
mydomain.com.  10800      IN       NS      mainserver.mydomain.com.

;; Query time: 120 msec
;; SERVER: 192.168.1.254#53(192.168.1.254)
;; WHEN: Mon Jul 14 00:39:37 2003
;; MSG SIZE  rcvd: 63
```

- To check IP of any machine by querying a proper nameserver you can use:

```
[root@mainserver /]# dig @mainserver.mydomain.com   wks2.mydomain.com A

; <<>> DiG 9.2.1 <<>> @mainserver.mydomain.com wks2.mydomain.com A
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34767
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
;wks2.mydomain.com.     IN      A

;; ANSWER SECTION:
wks2.mydomain.com.   10800   IN      A       192.168.1.2

;; AUTHORITY SECTION:
mydomain.com.   10800   IN      NS      mainserver.mydomain.com.

;; ADDITIONAL SECTION:
mainserver.mydomain.com.   10800   IN    A       192.168.1.254

;; Query time: 2 msec
;; SERVER: 192.168.1.254#53(mainserver.mydomain.com)
;; WHEN: Mon Jul 14 00:43:38 2003
;; MSG SIZE  rcvd: 84
```

- Checking for a machine IP which does not exist, will return in: `ANSWER:0`

```
[root@mainserver /]# dig @mainserver.mydomain.com   bigboss.mydomain.com A

; <<>> DiG 9.2.1 <<>> @mainserver.mydomain.com bigboss.mydomain.com A
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 49812
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;bigboss.mydomain.com.       IN      A

;; AUTHORITY SECTION:
```

```
mydomain.com.   3600    IN      SOA     mainserver.mydomain.com.
➥       root.mainserver.mydomain.com. 1 10800 3600 604800 3600

;; Query time: 35 msec
;; SERVER: 192.168.1.254#53(mainserver.mydomain.com)
;; WHEN: Mon Jul 14 00:45:50 2003
;; MSG SIZE  rcvd: 82
```

- Now check for IP to name resolution (reverse name lookups):

```
[root@mainserver /]# dig @mainserver.mydomain.com 2.1.168.192.in-addr.arpa PTR

; <<>> DiG 9.2.1 <<>> @mainserver.mydomain.com 2.1.168.192.in-addr.arpa PTR
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10703
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
;2.1.168.192.in-addr.arpa.      IN      PTR

;; ANSWER SECTION:
2.1.168.192.in-addr.arpa. 10800 IN      PTR     wks2.mydomain.com.

;; AUTHORITY SECTION:
1.168.192.in-addr.arpa. 10800   IN      NS      mainserver.mydomain.com.

;; ADDITIONAL SECTION:
mainserver.mydomain.com.        10800   IN      A       192.168.1.254

;; Query time: 33 msec
;; SERVER: 192.168.1.254#53(mainserver.mydomain.com)
;; WHEN: Mon Jul 14 00:50:42 2003
;; MSG SIZE  rcvd: 106
```

- This can also be done by the following command:

```
[root@mainserver /]# dig -x 192.168.1.2

; <<>> DiG 9.2.1 <<>> -x 192.168.1.2
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2549
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
;2.1.168.192.in-addr.arpa.      IN      PTR

;; ANSWER SECTION:
2.1.168.192.in-addr.arpa. 86400 IN      PTR     wks2.mydomain.com.

;; AUTHORITY SECTION:
1.168.192.in-addr.arpa. 86400   IN      NS      mainserver.mydomain.com.

;; ADDITIONAL SECTION:
mainserver.mydomain.com. 86400  IN      A       192.168.1.254

;; Query time: 5 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Tue Feb 17 01:49:53 2004
```

```
;; MSG SIZE  rcvd: 114
```

- You can get a list of all the machines of a domain by :

```
[root@mainserver /]# dig axfr  mydomain.com

; <<>> DiG 9.2.1 <<>> mydomain.com axfr
;; global options:  printcmd
mydomain.com.           86400   IN      SOA     mainserver.mydomain.com.
➥       root.mainserver.mydomain.com. 1 28800 7200 604800 86400
mydomain.com.           86400   IN      NS      mainserver.mydomain.com.
mainserver.mydomain.com. 86400  IN      A       192.168.1.254
wks1.mydomain.com.      86400   IN      A       192.168.1.1
wks2.mydomain.com.      86400   IN      A       192.168.1.2
mydomain.com.           86400   IN      SOA     mainserver.mydomain.com.
➥       root.mainserver.mydomain.com. 1 28800 7200 604800 86400
;; Query time: 6 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Tue Feb 17 01:50:32 2004
;; XFR size: 7 records
```

- You can query the MX , SOA and NS records by using :

```
[root@mainserver /]# dig MX mydomain.com

; <<>> DiG 9.2.1 <<>> MX mydomain.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1644
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
;mydomain.com.                  IN      MX

;; ANSWER SECTION:
mydomain.com.           86400   IN      MX 10 mainserver.mydomain.com.

;; AUTHORITY SECTION:
mydomain.com.           86400   IN      NS      mainserver.mydomain.com.

;; ADDITIONAL SECTION:
mainserver.mydomain.com. 86400  IN      A      192.168.1.254

;; Query time: 57 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sat Mar 13 14:50:01 2004
;; MSG SIZE  rcvd: 87


[root@mainserver /]# dig SOA mydomain.com

; <<>> DiG 9.2.1 <<>> SOA mydomain.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 64713
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; QUESTION SECTION:
;mydomain.com.                  IN      SOA

;; ANSWER SECTION:
```

```
mydomain.com.           86400   IN      SOA     mainserver.mydomain.com.
➥       root.mainserver.mydomain.com. 1 28800 7200 604800 86400

;; AUTHORITY SECTION:
mydomain.com.           86400   IN      NS      mainserver.mydomain.com.

;; ADDITIONAL SECTION:
mainserver.mydomain.com. 86400  IN      A       192.168.1.254

;; Query time: 5 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sat Mar 13 14:51:37 2004
;; MSG SIZE  rcvd: 112
```

```
[root@mainserver /]# dig NS mydomain.com

; <<>> DiG 9.2.1 <<>> NS mydomain.com
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26538
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; QUESTION SECTION:
;mydomain.com.                  IN      NS

;; ANSWER SECTION:
mydomain.com.           86400   IN      NS      mainserver.mydomain.com.

;; ADDITIONAL SECTION:
mainserver.mydomain.com. 86400  IN      A       192.168.1.254

;; Query time: 22 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sat Mar 13 14:54:22 2004
;; MSG SIZE  rcvd: 71
```

Ping Testing:

Ping is basically used for physical connectivity test. It has nothing to do with DNS ***directly***.

```
[root@mainserver /]# ping mainserver.mydomain.com
PING mainserver.mydomain.com (192.168.1.254) from 192.168.1.254 : 56(84) bytes of
data.
Warning: time of day goes back, taking countermeasures.
64 bytes from mainserver.mydomain.com (192.168.1.254): icmp_seq=0 ttl=255 time=685
usec
64 bytes from mainserver.mydomain.com (192.168.1.254): icmp_seq=1 ttl=255 time=91
usec
64 bytes from mainserver.mydomain.com (192.168.1.254): icmp_seq=2 ttl=255 time=91
usec

--- mainserver.mydomain.com ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.091/0.289/0.685/0.280 ms
```

```
[root@mainserver /]# ping wks1.mydomain.com
PING wks1.mydomain.com (192.168.1.1) from 192.168.1.254 : 56(84) bytes of data.
Warning: time of day goes back, taking countermeasures.
64 bytes from wks1.mydomain.com (192.168.1.1): icmp_seq=0 ttl=255 time=1.121 msec
64 bytes from wks1.mydomain.com (192.168.1.1): icmp_seq=1 ttl=255 time=516 usec
64 bytes from wks1.mydomain.com (192.168.1.1): icmp_seq=2 ttl=255 time=506 usec
```

```
--- wks1.mydomain.com ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.506/0.714/1.121/0.288 ms
```

You should also be able to ping from clients to server and other clients without their local ***/etc/hosts*** file having any entries.

Note:

In windows computers, Network Configuration -> DNS -> **Host** means the same computer (windows computer) you are sitting at, NOT the server. And DNS means the DNS domain name implemented on the network.

Terminology / abbreviations used in named zone configuration files:

SOA:
Start Of Authority. The SOA record indicates that this name server is the best source of information for the data within this zone. There can be one and only one SOA record in the zone data file.

IN:
IN stands for ***Internet*** address. This is one class of address. Others exist but are not in widespread use. Used to assign / specify IP address (A) or nameserver (NS) or pointer (PTR) or Canonical Name (CNAME) specification to machine names.

NS:
NS stands for ***NameServer***. Used to specify the name of the name server of the zone.

```
mydomain.com. IN NS       mainserver.mydomain.com.
```

or

```
@       IN       NS       mainserver.mydomain.com.
```

MX:

Mail Exchanger. Assuming that we want to use the machine **`mainserver.mydomain.com`** as our mail exchanger server as well. Then the following configuration needs to be done in the **`mydomain.com.zone`** file:

```
@ IN     NS       mainserver.mydomain.com.
@ IN     MX       10       mainserver.mydomain.com.
```

In the above line having MX in it, the number 10 is the mail delivery priority. It can be any positive number. The smaller the number, the higher the priority. Remember that there are no corresponding reverse entries in the reverse zone file for the MX entires in the forward zone files.

A:
Address. Indicates / specifies the IP address of the machine.

```
wks1.mydomain.com. IN     A        192.168.1.1
```

PTR:
Indicates ***Pointer***. Means what name the IP address points to. The name of the machines in the PTR record must be fully qualified domain name.

```
1.1.168.192.in-addr.arpa. IN      PTR      wks1.mydomain.com.
```

or

```
1        IN       PTR      wks1.mydomain.com.
```

Aliases / CNAME:
All names provided in the zone files having **A** record type are in fact ***canonical names.*** . e.g. in the following line wks1 is canonical name.

```
wks1     IN       A        192.168.1.1
```

once we want to use another name for a machine which is already assigned a name then we use aliases. e.g. Assume we want to use our ***`mainserver`*** as ***`www`*** as well.

We will use the following lines in our ***`mydomain.com.zone`*** file:

```
wks1             IN       A        192.168.1.1
mainserver       IN       A        192.168.1.254
www              IN       CNAME    mainserver
```

Appending domain names (*"the game of dot"* shall I call it ?) :
The domain name is the *Origin* of all the data in the zone files. Since the origin is appended to names, instead of entering the following line in ***mydomain.com.zone***:

```
wks1.mydomain.com. IN     A        192.168.1.1
```

we can enter:

```
wks1     IN       A        192.168.1.1
```

So the domain name *mydomain.com* will automatically be appended to the computer name which does not end with a period (. dot). Also instead of following line in file ***1.168.192.in-addr.arpa.zone*****:**

```
1.1.168.192.in-addr.arpa. IN       PTR      wks1.mydomain.com.
```

we can enter:

```
1        IN       PTR      wks1.mydomain.com.
```

The @ notation:
If the domain name is same as origin, the name can be specified as **@** . This is most often seen in SOA records. So the following line in the ***mydomain.com.zone*** file,

```
mydomain.com.     IN       SOA      mainserver       root.mainserver (
```

can be written as:

```
@ IN     SOA      mainserver root.mainserver (
```

TTL:
Time To Live. The TTL on a resource record, is the time in seconds any server can cache that record. So if the TTL for a particular resource record is 3600 (seconds), and a server outside your domain caches that record, it will have to remove the entry from its cache after an hour. If it needs the same data after the hour is up, it'll have to query your name servers again. You can have a separate TTL for SOA resource record and as separate one for your zone data in the A type resource records. Compare these two below:

```
$TTL 86400
$ORIGIN mydomain.com.
@ IN     SOA      mainserver.mydomain.com. root.mainserver (
                  1 ; serial
                  28800 ; refresh
                  7200 ; retry
                  604800 ; expire
```

```
                86400 ; minimum ttl
)

wks1    3600    IN      A       192.168.1.1
wks2            IN      A       192.168.1.2
```

This example is showing that the default TTL for the entire zone is 86400 seconds (or 24 hours). The TTL for the SOA record is also 86400 seconds. The TTL for wks1 is 3600 seconds (or 1 hour). And the TTL for wks2 is not specified. This entire scenario means that any other name server querying our name server will keep the queried information in it's cache for 24 hours by default; and will come to us again only when 24 hours have been passed since last query. This is true for the SOA information or information about wks2. But if the queried information was about wks1, then the other nameserver will have to lookup again to our server if 3600 seconds (one hour) has been passed since last query performed for wks1.

Refresh:

IN case master-slave DNS servers are implemented, this directive (refresh interval) tells the slave: “how often to check that its data are up to date?”

Retry:

Again used in master-slave DNS server implementation. If the slave fails to reach the master name server(s) after the refresh period (the host(s) could be down), then it starts trying to connect every retry seconds. Normally, the retry interval is shorter than the refresh interval, but it doesn't have to be.

Expire:

If the slave fails to contact the master server(s) for expire seconds, the slave expires its data. Expiring the data means the slave stops giving out answers about the data because the data are too old to be useful. Essentially, this field says: at some point, the data are so old that having no data is better than having stale data. The expiration time should always be much larger than the retry and refresh intervals; if the expire time is smaller than the refresh interval, your slaves will expire their data before trying to load new data.

Chapter 5: **NFS**

NFS stands for Network File System. Used for file sharing between *nix machines only. If you want to share files between Windows and Unix machines then you need to use SAMBA for that.

Packages:

Following packages needs to be installed on all those machines on which directories need to be shared.

`nfs-utils-a.b`
`portmap-a.b`

Scenario:

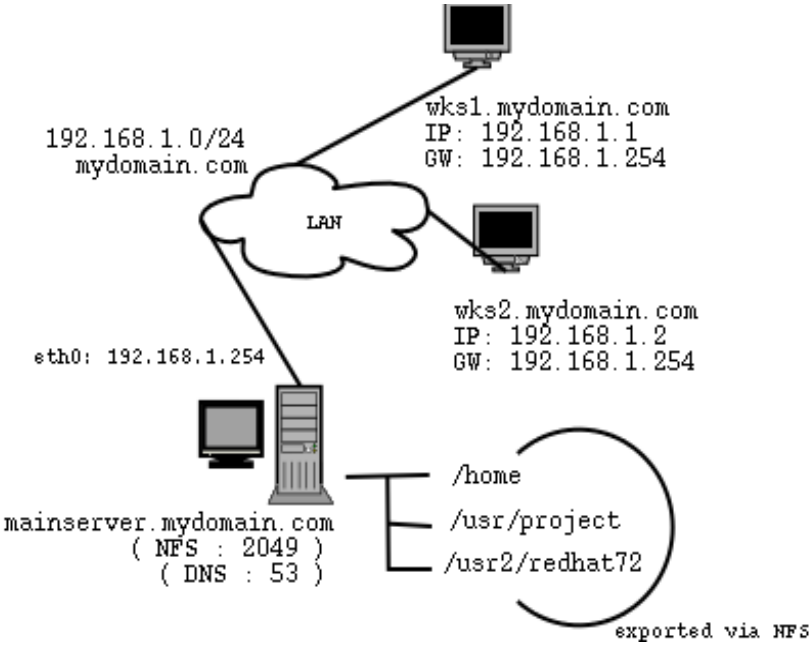


Fig 1

We need to share the following directories on the server for the entire domain **mydomain.com**.

`/home`	to be shared with read-write permissions for the entire lan,
`/usr/project`	project directory of some developers.
`/usr2/redhat72`	contains all files copied from Redhat 7.2 installation CDs

with read-only permissions.

Server side configuration:

Following file needs to be configured on the machine on which you want to share the directories.

```
[root@mainserver /]# /etc/exports
```

The format of this file is:

`DirectoryPath  hostlist(permissions)`

Where path is the absolute path of the directory you want to share and host list can be one or more hosts that can access the share. Each host to be separated by a white space. Hosts can be specified in the following ways:

- By using a hostname, if the host is part of local domain. eg. *`wks1`*
- By using a fully qualified domain name. eg. *`wks1.mydomain.com`*
- By using a hostname containing a wild card. e.g. *`*.mydomain.com`*
- By using IP address. e.g. *`192.168.1.1`*
- By using a network address. e.g. *`192.168.1.0/255.255.255.0`* or *`192.168.1.0/24`*

And permissions can be (***`ro`***) for *read only* and (***`rw`***) for *read & write*. DO NOT specify any white space between host specifier and permissions.
Also w.r.t permissions, by default, if a client is logged in as root on his own machine, then he will NOT be given root permissions on the machine hosting the shares (he will be squashed !) . Instead, such requests are mapped to user id 65535 (which of-course doesn't exist on a normal system). If you want to permit remote ***root*** acting as local root on the machine holding shares, then you need to enable a special permission as ***(`no_root_squash`).*** e.g.

```
/etc             wks1.mydomain.com(rw,no_root_squash)
```

This line means that if root logs in from wks1.mydomain.com, and he tries to write some files in the `/etc` directory of the computer holding the shares, then he will be ALLOWED to do all the editing to the files which are write able by root of local machine.

So for now the file looks like this:

```
[root@mainserver /]# cat /etc/exports
/usr2/redhat72            192.168.1.0/24(ro,no_root_squash)
/home                     *.mydomain.com(rw)
/usr/project              *.mydomain.com(rw)
```

Once `/etc/exports` file is edited, you need to run:

```
[root@mainserver /]# /etc/rc.d/init.d/portmap restart
```

```
[root@mainserver /]# /etc/rc.d/init.d/nfs start
```

Client side configuration:

Whatever the name resolution method being used in the domain, it should be working correctly / configured correctly else NFS mount will normally fail and will also give weird messages. If you occasionally need to access the shared directories on the server, then there is nothing to be configured on the client side. You just need to give the following command each time to want to access a share on the server.

```
[root@mainserver /]# mount -t nfs mainserver.mydomain.com:/usr2/redhat72
        ➥       /mnt/cdrom
```

The above line means that */usr2/redhat72* directory from the server mainserver.mydomain.com will be mounted on a local directory (mount point) */mnt/cdrom* using ***nfs*** protocol, or simply nfs file-system. And if you need to mount the share regularly or on every boot, then probably you should make a separate mount point for it on your local machine. e.g. You are working on a project with your team, and the directory */usr/project* is on the nfs server mainserver.mydomain.com, then you should make a mount point in your local work folder as */work/project* (note that the directory being used as a mount point should always be empty) and edit your */etc/fstab* file and put the following line in it.

```
mainserver.mydomain.com:/usr/project /work/project nfs defaults      0 0
```

The script `/etc/rc.d/init.d/netfs` runs at boot time, automatically mounting the NFS directories specified in the */etc/fstab* file. Also the ***nfslock*** service provides file locking services for NFS clients. So enable it if the client needs to lock files.

Log files:

```
/var/log/messages
```

Documentation and help:

```
www.tldp.org/HOWTO/NFS-HOWTO
/usr/share/doc/nfs-utils-*
/usr/share/doc/portmap-*
```

Testing:

To check if the NFS service is working properly the tests are performed from the server and client side, both.

Testing from server:

Check that at least the following three services are shown in the ***rpcinfo*** command: **portmapper**, **mountd** and **nfs**. If any one of them is missing then NFS will not work. Give the following command on the server, to see a list of services responding to remote procedure calls.

```
[root@mainserver /]# rpcinfo -p

   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100024    1   udp   1024  status
    100024    1   tcp   1024  status
    391002    2   tcp   1025  sgi_fam
    100011    1   udp    836  rquotad
    100011    2   udp    836  rquotad
    100011    1   tcp    839  rquotad
    100011    2   tcp    839  rquotad
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100021    1   udp   1026  nlockmgr
    100021    3   udp   1026  nlockmgr
    100021    4   udp   1026  nlockmgr
    100005    1   udp   1027  mountd
    100005    1   tcp   1028  mountd
    100005    3   udp   1027  mountd
    100005    3   tcp   1028  mountd
```

Now check if the directories you put in */etc/exports* are now available as shares or not by:

```
[root@mainserver /]# exportfs -v

/usr2/redhat72    *.mydomain.com(ro,wdelay,no_root_squash)
/home             *.mydomain.com(rw,wdelay,root_squash)
/usr/project      *.mydomain.com(rw,wdelay,root_squash)
```

and also by:

```
[root@mainserver /]# showmount -e

Export list for mainserver:
/home             *.mydomain.com
/usr2/redhat72    *.mydomain.com(ro,wdelay,no_root_squash)
/usr/project              *.mydomain.com(rw,wdelay,root_squash)
```

If you have made NFS shares available to the entire LAN then you should be able to mount the shares on the server itself, in some other directory / mount point:

```
[root@mainserver /]# mount -t nfs mainserver.mydomain.com:/usr2/redhat72
/mnt/cdrom
```

Note: If you edit your `/etc/exports` file later, then you need to run **`exportfs`** command again with a **-r** switch to refresh the share list. i.e.

```
[root@mainserver /]# exportfs -r
```

Client side testing and mounting:
You should be able to get *rpcinfo* from the server, which should be same as the command given on the server:

```
[root@mainserver /]# rpcinfo -p mainserver.mydomain.com

   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100024    1   udp   1024  status
    100024    1   tcp   1024  status
    391002    2   tcp   1025  sgi_fam
    100011    1   udp    836  rquotad
    100011    2   udp    836  rquotad
    100011    1   tcp    839  rquotad
    100011    2   tcp    839  rquotad
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100021    1   udp   1026  nlockmgr
    100021    3   udp   1026  nlockmgr
    100021    4   udp   1026  nlockmgr
```

```
    100005    1   udp   1027  mountd
    100005    1   tcp   1028  mountd
    100005    2   udp   1027  mountd
    100005    2   tcp   1028  mountd
    100005    3   udp   1027  mountd
    100005    3   tcp   1028  mountd
```

Now check which shares are available on the server by:

```
[root@mainserver /]# showmount -e mainserver.mydomain.com

Export list for mainserver:
/home            *.mydomain.com
/usr2/redhat72   *.mydomain.com
/usr/project             *.mydomain.com
```

Once shares are visible then, you are ready to mount the files on the local mount point. e.g.:

```
[root@mainserver /]# mount -t nfs mainserver.mydomain.com:/usr/project
/work/project
```

If the above command gives the prompt back, quietly, then it means that mount was successful. If mount was not successful, then check your NFS server side settings, and most importantly, the IP and hostname of the client computer, where you are trying to mount the server shares. As if the IP is not of the same subnet , then your host will not be in the same domain as per defined in DNS table on the server side. So NFS mount won't work. Also if your hostname is not correct, i.e. not defined in the DNS zone file or reverse entry for a computer is missing in the reverse zone file, then again your mount request will be declined. You may get ***RPC time out*** type of errors in such cases.

Also note the error caught in the /var/log/messages on the NFS server, below:

```
Apr  2 00:40:52 mainserver rpc.mountd: authenticated mount request from
➥       klaptop.mydomain.com:952 for /data (/data)
Apr  2 00:40:52 mainserver rpc.mountd: getfh failed: Operation not permitted
```

This error is encountered when you will try to mount a remote (exported) share over NFS, which is in-fact a directory mount point. e.g. The above error came by giving the following command on the client side:

```
[root@mainserver /]# mount -t nfs 192.168.1.254:/data /mnt/nfs
```

In my case, the ***`mount`*** command (on the server side), showed that `/data` is a mount point and `/dev/hda2` is mounted on it. So a directory under `/data` , like

`/data/rh9nfs` should be exported via NFS instead of `/data` and that directory in turn can be used / mounted by remote clients.

```
[root@mainserver /]# mount

/dev/hda6 on / type ext3 (rw)
none on /proc type proc (rw)
usbdevfs on /proc/bus/usb type usbdevfs (rw)
/dev/hda1 on /boot type ext3 (rw)
/dev/hda2 on /data type ext3 (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
none on /dev/shm type tmpfs (rw)
```

When some other (exported) directory under the hierarchy of this mount point, is mounted , then it is mounted properly:

```
Apr  2 00:41:01 mainserver rpc.mountd: authenticated mount request from
➥       klaptop.mydomain.com:953 for /data/rh9nfs (/data/rh9nfs)
Apr  2 00:42:29 mainserver rpc.mountd: authenticated unmount request from
➥       klaptop.mydomain.com:966 for /data/rh9nfs (/data/rh9nfs)
```

Chapter 6: **NIS**

Network Information Service. Formerly known as Yellow Pages or YP. NIS is used to provide login authentication to *nix machines. It is NOT used to provide login authentication for windows clients, for that we use SAMBA.

Packages:

ypserv-a.b (NIS server side)
ypbind-a.b (NIS client side)
yp-tools-a.b (Client server both)

Scenario

In the example network above, we need to implement NIS. NIS domain is always different from the DNS domain and these two should ***never*** be the same name. i.e. you cannot use the DNS ***mydomain.com*** in this case for NIS , the name of the authentication domain for NIS on our network will be ***logindomain***.

Configuration:

Server side configuration:

The following steps need to be configured / adjusted on the server:

1) Set domain name of this domain by using **ypdomainname** command

```
[root@mainserver /]# ypdomainname logindomain
```

2) Start ***ypserv*** daemon by:

```
[root@mainserver /]# /etc/rc.d/init.d/ypserv start
```

3) Go to `/var/yp` directory and create a new directory named **logindomain**

```
[root@mainserver /]# cd /var/yp
[root@mainserver yp]#  mkdir logindomain
```

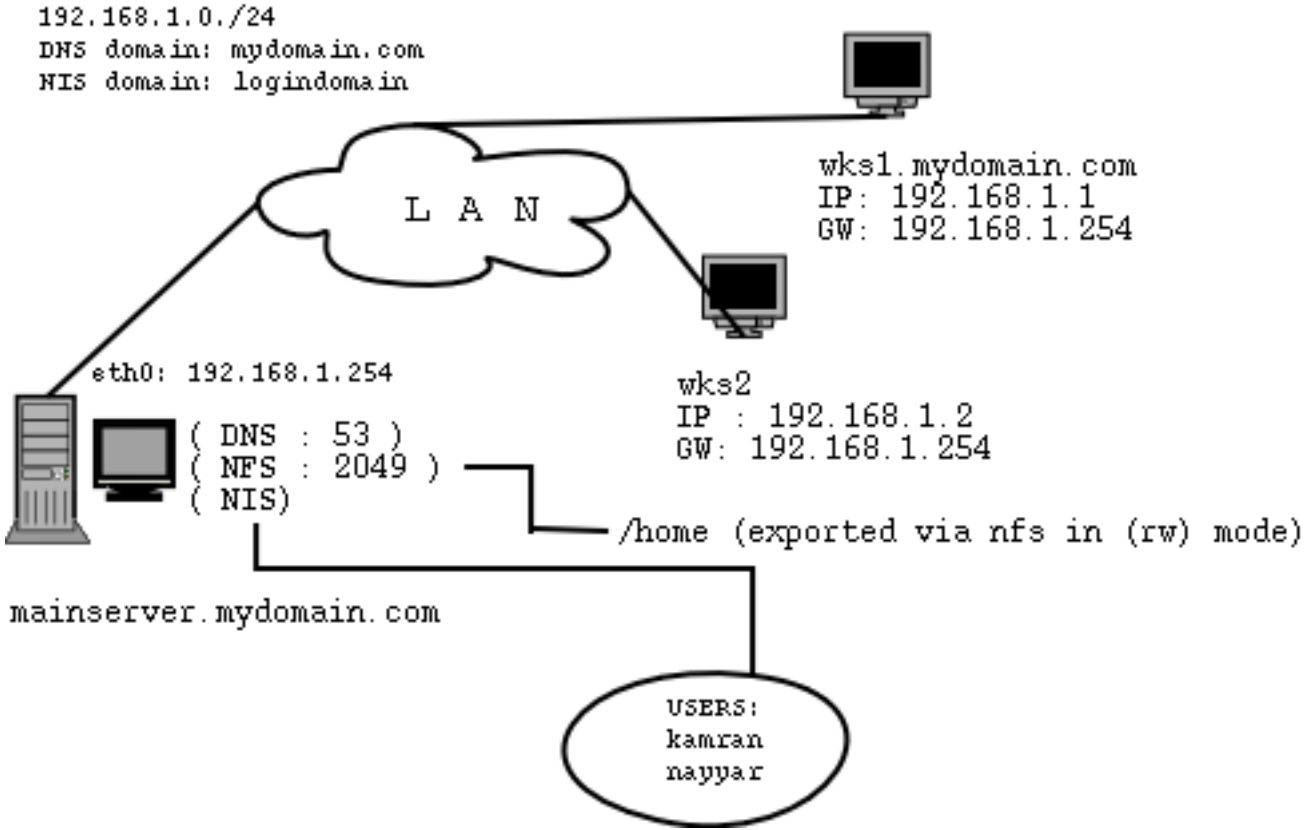

192.168.1.0./24
DNS domain: mydomain.com
NIS domain: logindomain
L A N
wks1.mydomain.com
IP: 192.168.1.1
GW: 192.168.1.254
wks2
IP : 192.168.1.2
GW: 192.168.1.254
eth0: 192.168.1.254
(DNS : 53)
(NFS : 2049)
(NIS)
/home (exported via nfs in (rw) mode)
mainserver.mydomain.com
USERS:
kamran
nayyar

Fig 1

4) Now run */usr/lib/yp/ypinit* to generate maps here, DO NOT enter the directory logindomain yourself. Press Ctrl-D in response to *next host to add:* and press ***Y*** in response to the question ***Is this correct? [y/n: y]*** :

```
[root@mainserver yp]#  /usr/lib/yp/ypinit -m

At this point, we have to construct a list of the hosts which will run NIS
servers.  mainserver is in the list of NIS server hosts.  Please continue to add
the names for the other hosts, one per line.  When you are done with the
list, type a <control D>.
        next host to add:  mainserver.mydomain.com
        next host to add:
The current list of NIS servers looks like this:

mainserver.mydomain.com

Is this correct?  [y/n: y]  y
We need a few minutes to build the databases...
Building /var/yp/logindomain/ypservers...
Running /var/yp/Makefile...
gmake[1]: Entering directory `/var/yp/logindomain'
Updating passwd.byname...
Updating passwd.byuid...
Updating group.byname...
Updating group.bygid...
Updating hosts.byname...
Updating hosts.byaddr...
Updating rpc.byname...
Updating rpc.bynumber...
Updating services.byname...
Updating services.byservicename...
Updating netid.byname...
Updating protocols.bynumber...
Updating protocols.byname...
Updating mail.aliases...
gmake[1]: Leaving directory `/var/yp/logindomain'

mainserver.mydomain.com has been set up as a NIS master server.

Now you can run ypinit -s mainserver.mydomain.com on all slave server.
```

5) It will make the maps of user names and passwords, etc. If it gives any errors during this, repeat the steps from step 1.

6) ***yppasswdd*** daemon is used to listen for password change requests from client computers and updates the ***/etc/passwd*** *and* ***/etc/shadow*** files on the NIS server. Start it by:

```
[root@mainserver /]# /etc/rc.d/init.d/yppasswdd start
```

7) You should also make sure that these services (***ypserv*** *and* ***yppasswdd***) are started by default at boot up time. So run ***setup*** **or** ***ntsysv*** and enable these services.

```
[root@mainserver /]# setup
[*]      ypserv
[*]      yppasswdd
```

or

```
[root@mainserver /]# chkconfig --level 35 ypserv on
[root@mainserver /]# chkconfig --level 35 yppasswdd on
```

8) Update ***/etc/sysconfig/network*** file and add the line at the end,

```
[root@mainserver /]# vi /etc/sysconfig/network

NETWORKING=yes
HOSTNAME=mainserver.mydomain.com
NISDOMAIN=logindomain
```

Client side configuration:

1) Modify the ***/etc/sysconfig/network*** and add the line at the end:

```
 NISDOMAIN=logindomain

[root@mainserver /]# vi /etc/sysconfig/network

NETWORKING=yes
HOSTNAME=wks2
GATEWAY=192.168.1.254
NISDOMAIN=logindomain
```

2) Modify the file ***/etc/nsswitch.conf*** and update as follows:

```
[root@mainserver /]# vi /etc/nsswitch.conf

passwd:          nis      files
shadow:          nis      files
group:           nis      files

hosts:           dns      files    nis
```

3) Run the ***ypbind*** service and also put it in the startup by:

```
[root@mainserver /]# /etc/rc.d/init.d/ypbind start
```

```
[root@mainserver /]# setup
[*]      ypbind
```

or

```
[root@mainserver /]# chkconfig –-level 35 ypbind on
```

Since this is an authentication service it is better to restart the client machine. Once restarted, on the login screen give in the login name and password of the user already created on the NIS server. Client machine should log the user with an error :

```
could not find the home directory of the user, logging on with home directory as
/.
```

This is not so important and can be solved. Basically the system tried to login to the home directory on the local machine but since this user did not exist at all on the local computer, the home directory also did not exist. Thus failure to go directly in the home directory. This can be solved by either creating a home directory for this user manually in the */home*; or mount the home directory of this user from the NIS server to the local machine using NFS. Assuming our example network, and assuming you want to load home directories from the NIS server permanent and regular basis, you need to add the following line in the */etc/fstab* file.

```
192.168.1.254:/home       /home    nfs      defaults 0       0
```

Also once the user wants to change his password from a client machine, he needs to use the ***yppasswd*** command. Simple ***passwd*** command will ***not*** work as it is designed to handle login/password stored in local **/etc/passwd** file only.

Documentation:

```
/usr/share/doc/yp-bind-*
/usr/share/doc/yp-serv-*
/usr/share/doc/yp-tools-*
```

Log files:

```
/var/log/messages
```

Chapter 7: **SAMBA**

SAMBA is derived from **SMB,** which stands for Session Message Block. It is a protocol used to exchange data between windows and *nix computers. SMB was built on Microsoft's NetBIOS protocol, which can be run over TCP/IP.

Packages:

```
samba-a.b          (server side)
samba-client-a.b   (client side)
samba-common-a.b   (client server both)
```

Scenario:

In fact there will be three scenarios.

1. Accessing shares on windows computers from *nix computers
2. Providing access to shares on the SAMBA server to users logging on from Windows computers, in a work-group model
3. Providing access to shares and logon authentication to users logging on from Windows computers, in a Domain model

Scenario1 (workgroup model, shares on windows machine):

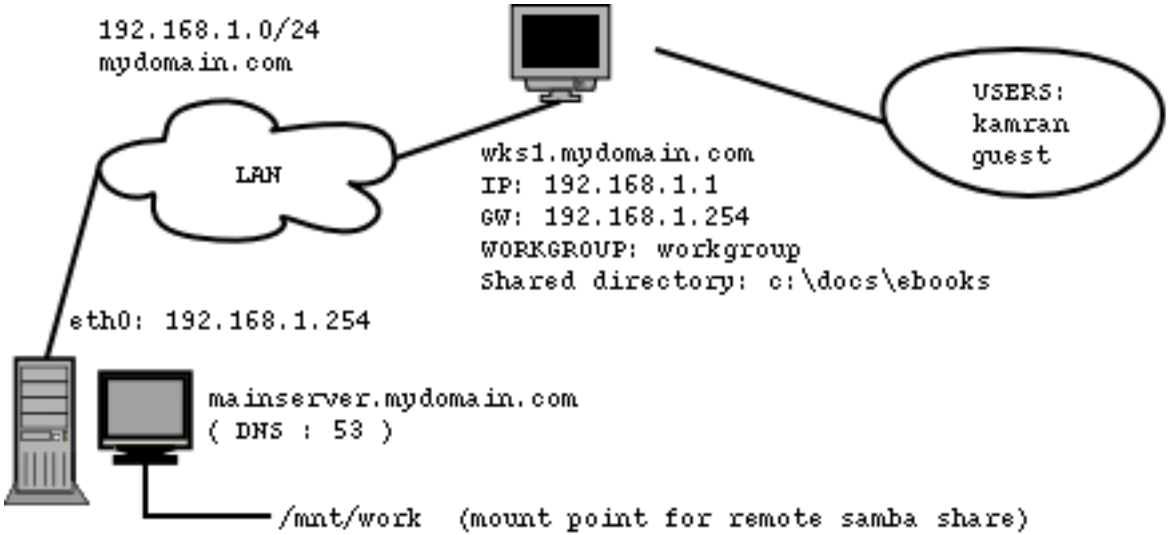
We have some share (`ebooks`) available on windows computer (*`wks1.mydomain.com`*) and want to access them from a linux machine.

Windows side configuration:
Nothing special, just enable sharing on the directories / folders you want to share. Optionally set permissions of these shares.

Linux side configuration:
There is no configuration required. Only the commands ***`smbclient`*** and ***`smbmount`*** will be used to access the windows machine. So the related packages must already be installed on the client side.

To see the list of shares available on the windows computer:

```
[root@mainserver /]# smbclient -L wks1.mydomain.com
```



SCENARIO : 1

Fig 1

Just press enter on the password prompt and you would be given a list of shares available on this computer.

Note: In case of windows 2000 / xp , simple pressing enter will not do the trick. You will be denied access to list of shares in this way. In order to see a list of shares on a windows 2000/xp, you need to specify a valid *username* already defined and enabled on the windows 2000/xp machine. When asked for password, you have to supply the password of that user. In case of a guest account, which is normally available and enabled on a windows network, you can use guest and a blank password with it:

```
[root@mainserver /]# smbclient -L wks1 -U guest
```

To connect to the share **ebooks** on the windows machine, with a windows user *kamran*, use:

```
[root@mainserver /]# smbclient //wks1/ebooks -U kamran
```

If you see the ***smb:\>*** prompt, you have successfully logged in.

If you don't use the `-U <username>` switch then the username and password of the currently logged in linux user will be sent to the windows machine, which may get rejected. So it is better to use the username and password of the user already created on the windows machine you are connecting to.

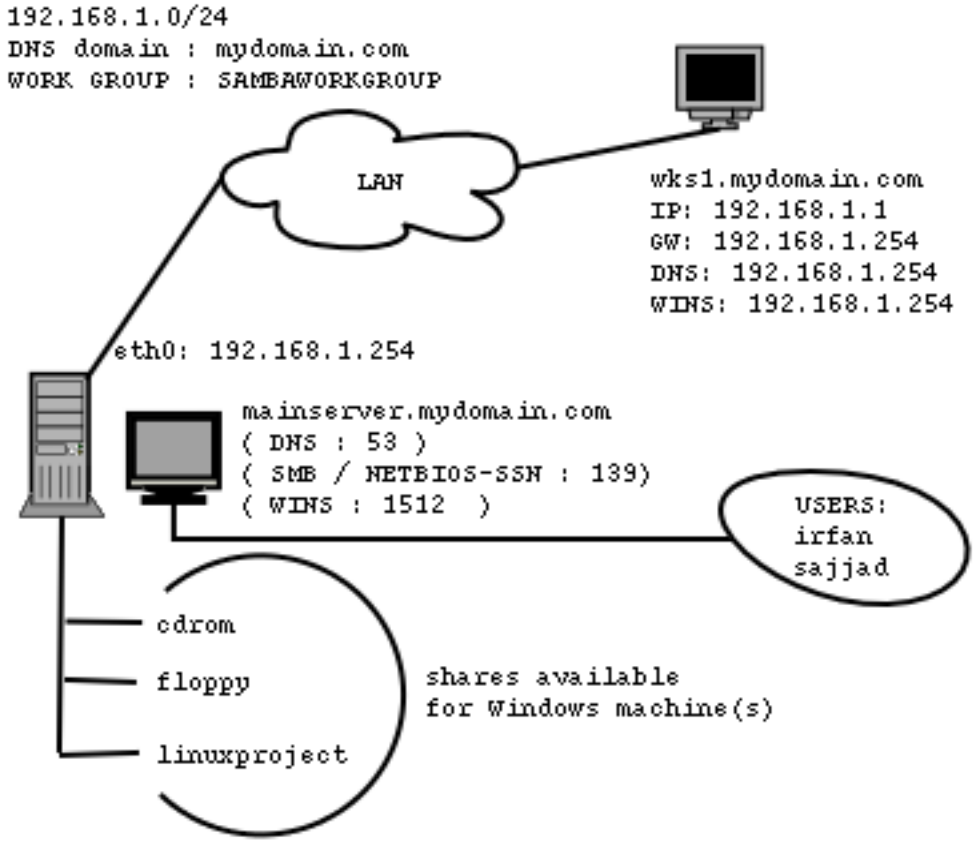
Once you are logged in, type **help** for a list of commands. Like FTP client, you can use ***get*** or ***mget*** and ***put*** or ***mput*** commands to copy files from and to the windows computer respectively. To exit `smbclient`, type ***quit*** at the `smb:\>` prompt.

If you want, you can mount the remote windows share ***ebooks*** on some local mount point on your machine, say **/mnt/work .** Do do so, use:

```
[root@mainserver /]# smbmount //wks1/ebooks /mnt/work -o username=kamran
```
or
```
[root@mainserver /]# mount -t smbfs //wks1/ebooks /mnt/work -o username=kamran
```

Scenario2 (workgroup model, shares on linux machine):

SAMBA server will be configured on our machine **mainserver.mydomain.com**. The ***workgroup*** name can be same as the one



SCENARIO 2

Fig 1

already being used in the current windows setup. However, we will create a new workgroup named *SAMBAWORKGROUP* and configure the windows clients to be a part of this workgroup.

Server side configuration:

1) Modify the `/etc/samba/smb.conf` file and give the following entries:

```
[root@mainserver /]# vi /etc/samba/smb.conf

[global]
workgroup = SAMBAWORKGROUP
comment = samba server on the local lan placed in the server room
hosts allow = 192.168.1.  192.168.2.       127.
security = user
load printers = yes
os level = 64
encrypt passwords = yes
smb passwd file = /etc/samba/smbpasswd
preferred master = yes
domain master = yes
wins support = yes

[cdrom]
comment = CDROM on SAMBA server
path = /mnt/cdrom
read only = yes
browseable = yes
public = yes

[floppy]
comment = Floppy drive on SAMBA server
path = /mnt/floppy
read only = no
browseable = yes
valid users = @students            ; note that the group students exists in /etc/groups

[linuxproject]
comment = project on linux by MS 4 students
path = /users/ms4/linuxproject
read only = no
browseable = yes
valid users = nayyar, inam
```

Note: All paths defined in the `smb.conf` file must exist. If they don't create them yourself.

2) Test your `smb.conf` file by running ***testparm*** command which is only available for SAMBA. This command should show all the shares without any errors.

```
[root@mainserver /]# testparm | less
```

3) If the above command works fine then run the ***smb*** services now by:

```
[root@mainserver /]# /etc/rc.d/init.d/smb start
```

4) Create users *inam* and *nayyar* (on the local linux machine, if not already created) and generate SMB passwords for them which will be used to access the shares defined on this computer from the windows client. Use the following commands to generate SMB passwords:

```
[root@mainserver /]# cat /etc/passwd | mksmbpasswd.sh > /etc/samba/smbpasswd
[root@mainserver /]# smbpasswd -a inam
[root@mainserver /]# smbpasswd -a nayyar
```

Note: The first command in step 4 needs to be given once.

Client side configuration:

On the client side, i.e. on the computer having Microsoft Windows, open the network neighborhood properties, and in the *identification* tab, type in *SAMBAWORKGROUP* in the workgroup text box. Also set the other network related properties, e.g. IP addresses, address of DNS and name of Primary WINS server which is the IP address of SAMBA server. Press OK and restart the windows machine. Once restarted, type in user name ***inam*** and his password. You will be logged on. Now you should be able to see the SAMBA server in the *Network Neighborhood* window. Double click it, and you should be able to see the shares: `inam, cdrom, floppy` and `linuxproject.` If you did not log in as a regular linux user, then **o**n double clicking on the SAMBA server will give you a login screen, here again you have a chance to provide a regular linux user account and it's samba password. If still you do not supply a regular login and you just press OK, then you will be presented with the shares on the server without the share ***inam*** displayed. In fact you will be logged in as user ***nobody*** , and this will be a lot restrictive access than assumed.

To assign a drive letter to a share, you can either right click on the share and select ***Map network drive*****.** Or you can use the command on the windows DOS shell prompt:

```
C:\> net use F: \\mainserver.mydomain.com\cdrom
```

Scenario 3 (SMB server as domain controller):

SAMBA server will be configured on our machine **`mainserver.mydomain.com`**. The workgroup name can be same as the domain name already being used in the current windows setup (provided the NT server for the domain is off line), BUT it should ***never*** be of the name name as your DNS domain name. We will create a new workgroup named ***`SAMBADOMAIN`*** on SAMBA machine (the server), and configure the windows clients to be a part of this **domain**. (Confusing? Well, on a SAMBA machine, in `/etc/samba/smb.conf`, the word **workgroup** serves for workgroup as well as domain for windows computers).

Server side configuration:

1) Modify the `/etc/samba/smb.conf` file and give the following entries:

```
[root@mainserver /]# vi /etc/samba/smb.conf

[global]
workgroup = SAMBADOMAIN
comment = samba server on the local lan placed in the
server room
hosts allow = 192.168.1.      192.168.2.  127.
security = user
load printers = yes
encrypt password = yes
smb passwd file = /etc/samba/smbpasswd
domain logons = yes
os level = 64
preferred master = yes
domain master = yes
wins support = yes

[homes]
comment = home directories of users
read only = no
create mask = 0750
browseable = no

[netlogon]
comment = SAMBA network logon services for microsoft
clients
path = /users/netlogon
guest ok = yes
browseable = no
locking = no

[cdrom]
```

```
comment = CDROM on SAMBA server
path = /mnt/cdrom
read only = yes
browseable = yes
public = yes

[floppy]
comment = Floppy drive on SAMBA server
path = /mnt/floppy
read only = no
browseable = yes
valid users = @students        ; note that the group students exists in
/etc/groups

[linuxproject]
comment = project on linux by MS 4 students
path = /users/ms4/linuxproject
read only = no
browseable = yes
valid users = nayyar, inam
```

Note: All paths defined in the *smb.conf* file must exist. If they don't create them yourself. Also if you are going to use clients of Windows 98 and onwards or Windows NT service pack 3 or higher, then these systems use encrypted passwords by default. We need to provide either encrypted password support in SAMBA server or modify the registry of the windows computers and enable plain text passwords in them.

To enable support for encrypted passwords in the SAMBA server, perform the following steps:

a. Create a separate password file for Samba. To create one based on your existing `/etc/passwd` file, at a shell prompt, type the following command:

```
[root@mainserver /]# cat /etc/passwd | mksmbpasswd.sh > /etc/samba/smbpasswd
```

The *mksmbpasswd.sh* script is installed in your `/usr/bin` directory with the `samba` package.

b. Use the following command to change permissions on the Samba password file so that only root has read and write permissions:

```
[root@mainserver /]# chmod 600 /etc/samba/smbpasswd
```

c. The script does not copy user passwords to the new file. To set each Samba user's password, use the command `smbpasswd username`(replace *username* with each user's username). A Samba user account will not be active until a Samba password is set for it.

```
[root@mainserver /]# smbpasswd -a inam
```

d. The next step is to enable encrypted passwords in the Samba configuration file. In the file *smb.conf*, uncomment the following lines:

```
encrypt password = yes
smb passwd file = /etc/samba/smbpasswd
```

2) Test your `smb.conf` file by running ***testparm*** command which is only available for SAMBA. This command should show all the shares without any errors.

```
[root@mainserver /]# testparm
```

3) If the above command works fine then run the `smb` services now by:

```
[root@mainserver /]# /etc/rc.d/init.d/smb start
```

Notes:

Windows NT 4:
At present, Samba needs to have a *nix account for every entry in the '*smbpasswd*' file, so this means that each Windows NT4/2000computer needs a *nix account on the Domain Controller. Machine accounts are differentiated from user accounts by appending a $ to the end of the machine's name. For Windows NT clients, you can create these accounts manually. To make a domain machine account, issue the following commands on the samba Domain Controller as 'root':

Create the group "machines" by:

```
[root@mainserver /]# groupadd machines

[root@mainserver /]# useradd -d /dev/null -g machines -c 'Machine Account'
        ➥       -s /bin/false    -M       <NETBIOS_NAME>$

[root@mainserver /]# smbpasswd -am <NETBIOS_NAME>
```

e.g. To add a machine account of a machine win98 on the samba server, you would use:

```
[root@mainserver /]# groupadd machines

[root@mainserver /]# useradd -d /dev/null -g machines -c 'Machine Account' -s
                  ➥       /bin/false -M    win98$

[root@mainserver /]# smbpasswd -am win98
```

Windows 2000/XP:
Windows 2000 is different than NT. As far as NT is concerned, while joining the domain, you are given an option to create a computer account in the domain server. This allows machine accounts to be made on the fly. At the moment the only user that can create computer accounts automatically is root. This means we must first make a samba password for 'root' with:

```
[root@mainserver /]# smbpasswd -a root
```

It is recommended that you use a different password than real Linux password for root for security reasons. To make a domain machine account, issue the following commands on the Domain Controller as '***root***':

```
[root@mainserver /]# groupadd machines

[root@mainserver /]# useradd -d /dev/null -g machines -c 'Machine Account'
         ➥       -s /bin/false -M win2k$

[root@mainserver /]# smbpasswd -am win2k
```

Optionally in order to make the computer accounts on the fly while joining the domain, you need the following entry in the smb.conf file.

```
add user script = /usr/sbin/useradd -d /dev/null -g machines
         ➥       -c       'Machine account' -s /bin/false -M %u
```

Note the absence of ***$*** in this script, SAMBA automatically adds the *$* for you when it is configured as domain log on server. Also the group **machines** was already created manually. This also works for Windows XP.

The above solution specifically solves the problem when you are joining the Windows 2000 workstation to a samba domain and you get the following error: **"The account used is a computer account. Use your global user account or local user account to access the server"**. This problem comes with Windows 2000

machines with service pack 2. Another solution is to try changing the windows computer's machine name from Network properties, but this seldom works (depends on the mood of win 2000 machine).

Client side configuration:

On the client side, i.e. on the computer having Microsoft Windows, open the network neighborhood properties, and in the *identification* tab, type in SAMBADOMAIN in the DOMAIN textbox. Also set the other network related properties, e.g. IP addresses, address of DNS and name of Primary WINS server which is the IP address of SAMBA server. Press ok and restart the windows machine. Once restarted, type in any user name and password of the user defined on SAMBA server and select the domain SAMBADOMAIN in the drop down list. You will be greeted after a pause of 10 to 30 seconds. Restart and log on to the new domain with the user name **inam**, lets say. Now you should be able to see the SAMBA server in the Network Neighborhood window. Double click it, and you should be able to see the shares: **inam**, **cdrom, floppy** and **linuxproject.** On double clicking on the share **linuxproject** you should be able to see the contents of this share**.** You will be asked for the password if you logged on as someone else other than *inam* or *nayyar*, this will be the password of one of the users *nayyar* or *inam* (as shown above), which are defined in the machine running SAMBA. Similarly password of one of the users in the group students (defined in SAMBA machine) may be required once accessing the share floppy. To assign a drive letter to a share, you can either right click on the share and select **Map to drive.** Or you can use the command on the windows DOS shell prompt:

```
C:\> net use F: \\mainserver.mydomain.com\cdrom
```

Common problems while joining a Linux SAMBA domain from win2k client:

1) "The account used is a computer account. Use your global user account or local user account to access the server".

This problem comes with Windows 2000 machines with service pack 2. Solution is to create a machine account for each win2k computer on the SAMBA computer. Another solution is to try changing the windows computer's machine name from Network properties, but this seldom works (depends on the mood of win 2000 machine).

2) "The credentials supplied conflict with an existing set of credentials".

This is a problem on windows client side.. Windows will not let you connect as two different users to the same share on the remote computer. To over come this problem, log out of the windows computer (restart if it doesn't behave) , then again log in first as local administrator and then supply the root login and it's samba generated password while joining the domain.

3) "The domain SMBDOMAIN could not be contacted".

Check your SMB configuration. Check for presence of machine accounts (including the group **machines**) and their SMB passwords. Check the network settings of windows. Restart the windows computer. Should work.

Now something about plain text passwords:

In the Samba configurations above, we are using encrypted passwords. But if for any reason, for example, incompatibility of samba encrypted passwords and windows encrypted passwords, we can configure both sides to use plain text passwords. On samba side, we will just put a remark on the following two lines in the `smb.conf` file

```
#encrypt password = yes
#smb passwd file = /etc/samba/smbpasswd
```

And on windows side we have to edit the registry and enable plain text password support. The procedure is different for win 9x, windows NT and windows 2000.

Windows 9x registry settings:
Locate the following registry key:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\VxD\
VNETSUP
```

Select `Edit -> New` and choose to create a new `DWORD` value.

In *Regedit* insert a new **DWORD** value called *`New Value #1`* in the registry. Rename this new `bvalue` ***`EnablePlainTextPassword`*** and double click this new name. In the dialog box, enter ***`1`*** as the value and close regedit program. Restart windows.

Windows NT registry settings:
Locate the following registry key:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Rdr\
Prameters
```

Select `Edit -> New` and choose to create a new `DWORD` value.

In regedit insert a new **DWORD** value called *`New Value #1`* in the registry. Rename this new bvalue **`EnablePlainTextPassword`** and double click this new name. In the dialog box, enter ***`1`*** as the value and close regedit program. Restart windows.

Windows 2000 registry settings:
Method 1:
Under windows 2000 this can be done doing: `Start -> Settings -> ControlPanel -> Administrative Tools -> Local Security Policy`. In the `Local Security Settings` dialog box, select `Local Policies -> Security Options`, double click on `Send unencrypted password to connect to third party SMB servers` and enable it. Reboot the machine.

Method 2:
Locate the following registry key:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Lanm
anWorkStation\Parameters
```

Select `Edit -> New` and choose to create a new `REG_DWORD` value.

Regedit will insert a new **REG_DWORD** value called *New Value #1* in the registry. Rename this new *bvalue* **EnablePlainTextPassword** and double click this new name. In the dialog box, enter ***1*** as the value and close regedit program. Restart windows.

Log Files:

`/var/log/messages` *(file)*
`/var/log/samba` *(directory)*

Documentation and Help:

`/etc/samba/smb.conf`
`/usr/share/doc/samba-*/docs` *(excellent documentation)*
`/usr/share/doc/samba-*/docs/DIAGNOSIS.TXT`

To read more about *Using Samba with Windows NT 4.0 and Windows 2000*, read `ENCRYPTION.TXT`, `Win95.txt`, and `WinNT.txt` in the directory `/usr/share/doc/samba*/docs/textdocs/`

Testing:

To list the shares available on the local host, use:

```
[root@mainserver /]# smbclient -L localhost -N
```

To list the shares available on the windows machine, use:

```
[root@mainserver /]# smbclient -L wks1 -N
```

If no shares are visible, then WINS server may not be responding. To check this use:

```
[root@mainserver /]# nmblookup -B 192.168.1.255 __SAMBA__
```

(Notice the two underscores before and after SAMBA)

The output of above command should display the IP address of `mainserver.mydomain.com`.

You should also check that windows machines are properly configured to use WINS by:

```
[root@mainserver /]# nmblookup -T '*'
```

Client hosts on this network should respond with their IP addresses.

Chapter 8: **SQUID**

Squid is basically Internet caching server which also does proxying. Also known as Squid-Proxy server. It caches HTTP and FTP data. It listens for user requests for HTTP and FTP on port `3128` by default (can be changed) and contacts to origin server if the web page requested is not in the squid's cache.

Packages:

`squid-a.b.STABLE.a.b`

Scenario:

We want our network to share the Internet connection running on the `mainserver.mydomain.com`, with our clients on our network. SQUID runs on the gateway machine i.e. `mainserver.mydomain.com` .

Configuration:

Normally only very minor changes are required in the file **`/etc/squid/squid.conf`** . Main configuration is to create an ***acl*** for your network and allow it ***http*** access. Also optionally you can make squid listen on port other than default 3128. Update it as follows Only important changes are highlighted:

```
[root@mainserver /]# vi /etc/squid/squid.conf

# TAG: http_port
http_port 3128


#  TAG: cache_dir
cache_dir ufs /var/spool/squid 200 16 256

acl mynet src 192.168.1.0/24

http_access allow mynet

http_access allow localhost

http_access deny all
```

You should also set the *`visible_hostname`* tag to the fully qualified domain name of this squid machine. This name should be the one defined in your DNS. Also it ***must not*** contain any invalid characters like underscores, etc. Use the same rules which are used to assign host names in DNS. If you put any invalid character here (or

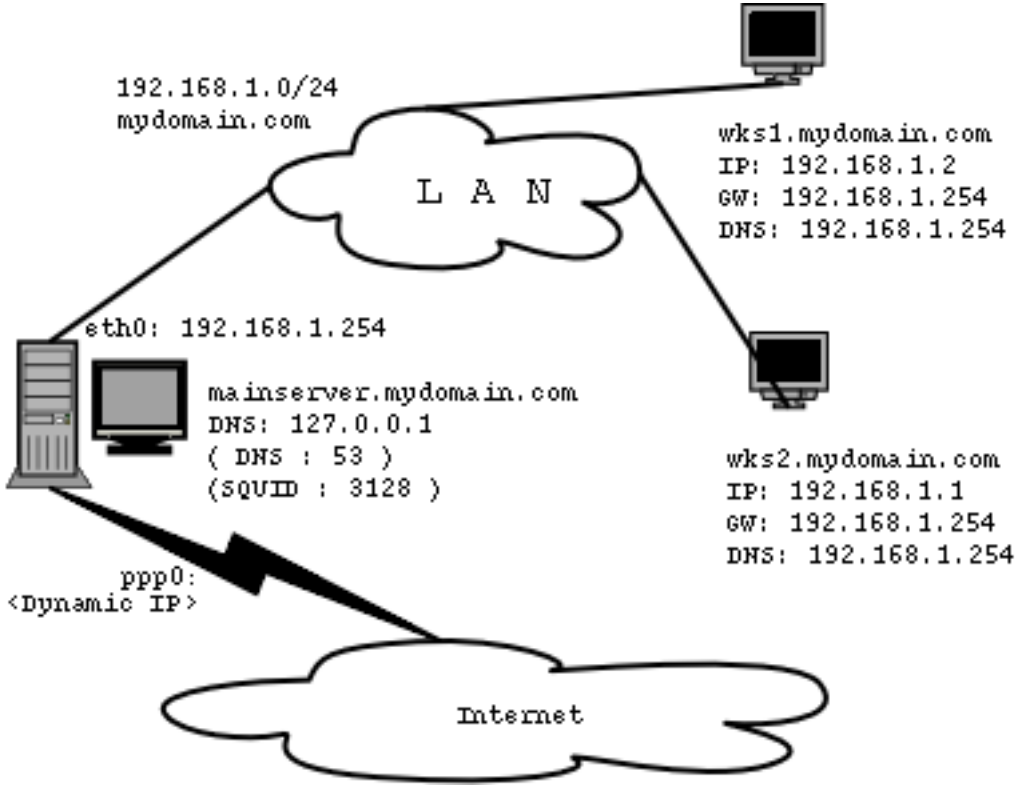

192.168.1.0/24
mydomain.com
L A N
wks1.mydomain.com
IP: 192.168.1.2
GW: 192.168.1.254
DNS: 192.168.1.254
eth0: 192.168.1.254
mainserver.mydomain.com
DNS: 127.0.0.1
(DNS : 53)
(SQUID : 3128)
wks2.mydomain.com
IP: 192.168.1.1
GW: 192.168.1.254
DNS: 192.168.1.254
ppp0:
<Dynamic IP>
Internet

Fig 1

in DNS) , you will see ugly messages like “mime error”, “error loading url”, etc, etc.

```
visible_hostname mainserver.mydomain.com
```

You should also set cache_effective_user and cache_effective_group

```
cache_effective_user squid
cache_effective_group squid
```

Now create the cache directory (defined in the **squid.conf** file) above otherwise squid service will fail (very important):

```
[root@mainserver /]# squid -z
```

Note that if you use some directory for squid's cache other than default /var/spool/squid, then make sure to change the new directory's ownership and permissions for squi:squid, otherwise the ***squid -z*** command will fail. Assuming you are using /squid/cache as your cache directory, then:

```
[root@mainserver /]# chown squid:squid /squid/cache -R

[root@mainserver /]# chmod 770 /squid/cache -R
```

After the cache directory creation is successful, run the SQUID service by:

```
[root@mainserver /]# /etc/rc.d/init.d/squid start
```

Transparent proxying:

SQUID on gateway machine:

If you are configuring SQUID to act as a transparent proxy server. i.e. You want SQUID to hijack all the requests for HTTP and HTTPS and should never let the user machine contact directly the origin server for web pages, not you want to configure each user machine and edit the proxy settings. This can be achieved by using some firewall redirection rules and some settings in SQUID.

Note the scenario is that squid is running on the gateway machine, i.e. in this case , `mainserver.mydomain.com`. *See Fig 1*. In addition to the settings above, the following entries need to be placed in the `squid.conf` file. These entries are normally found in the end of `squid.conf` file.

1) Modify `squid.conf` file and add / update as below:

```
[root@mainserver /]# vi /etc/squid/squid.conf

# HTTPD-ACCELERATOR OPTIONS
httpd_accel_host virtual
httpd_accel_port 80

httpd_accel_with_proxy on
httpd_accel_uses_host_header on
```

2) Put the following line in any of your startup scripts. Or you can make your own script and call it from your startup scripts. Normally `rc.local` script is recommended for such tasks as it runs after all the scripts and services are initialized/run at system boot up. This line redirects www (port 80) requests to local squid service running at port `3128` in this case. It also assumes that your SQUID server is connected to the hub or switch via `eth0` interface and is connected to the Internet via a modem dialup interface `ppp0`.

```
[root@mainserver /]# iptables -t nat -A PREROUTING -i eth0 -p tcp --dport
        ➥       80 -j REDIRECT --to-port 3128
```

For now apply the *`iptables`* command above on the command prompt and restart the squid service by:

```
[root@mainserver /]# /etc/rc.d/init.d/squid restart
```

Connect to the Internet via your dialup connection. You should see activity in the **`/var/log/squid/access.log`** file while the client machines try to connect to the Internet.

SQUID on one leg, another Linux mahcine as gateway:

In this case the SQUID machine is connected only with one ethernet connection (192.168.1.252) with our network. There is no other interface on this squid machine. The gateway machine (192.168.1.254) is also on linux, and iptables rules being used to redirect port 80 traffic towards our squid machine. *See Fig 2*. On the SQUID machine, just configure the SQUID for ***httpd*** acceleration. No redirection rules required here.

Transparent caching with Linux, SQUID as a node on network

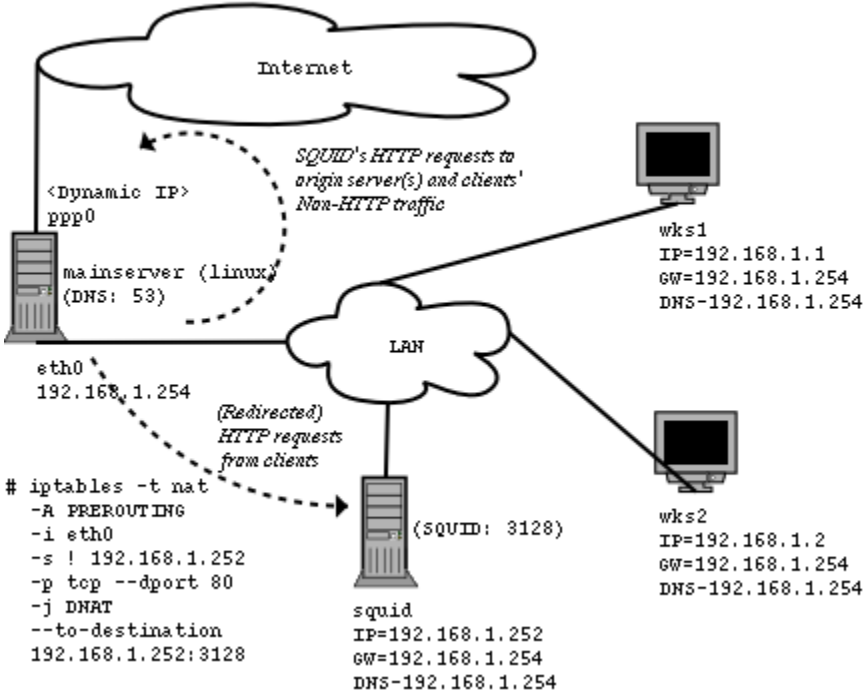


Fig 2

On the gateway machine, give the following command:

```
[root@mainserver /]# iptables -t nat -A PREROUTING -i eth0 -s ! 192.168.1.252
         ➥       -p tcp --dport 80 -j DNAT –-to-destination 192.168.1.252:3128
```

What this does is direct all traffic coming from local network, except the SQUID machine, destined for port 80 towards SQUID machine. And when SQUID machine will need to bring web content from origin server(s), then this rule will have no effect on it. Quite straight-forward right ?

SQUID on one leg, Cisco router as gateway:

If you have a router (say cisco) as your gateway(192.168.1.254), and your squid machine is again on one leg (192.168.1.252), then you need to define some *ACLs* on it to get the task done. *See Fig 3*.

Configure squid machine for *httpd* acceleration and then configure your Cisco router as following :

Define a *route map* with a name of lets say *squid-catch* and specify the IP of SQUID machine as the next hop for it.

```
!
route-map squid-catch permit 10
match ip address 110
set ip next-hop 192.168.1.252
!
```

Now define an access list to trap HTTP requests:

```
!
access-list 110 deny tcp any any neq 80
access-list 110 deny tcp host 192.168.1.252 any
access-list 110 permit tcp any any
!
```

Apply this on router's ethernet interface:

```
!
interface ethernet 0
ip policy route-map squid-catch
!
```

Transparent caching with Cisco router, SQUID as a node on network

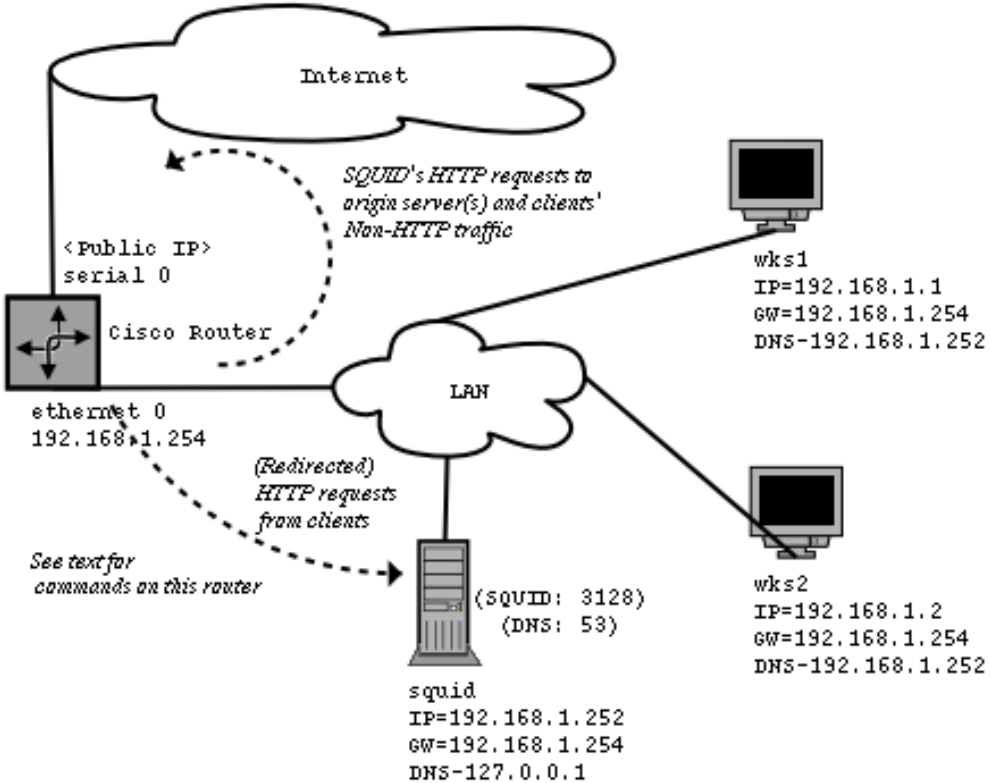


Fig 3

Stop porn for God's sake:

If you want to restrict users so that they are denied access to certain sites, like porn sites, etc. Then you can make an *acl* for bad sites put all the possible words which are normally part of url of those bad sites, in a text file with each word at separate line. Create and block the acl as follows in the `squid.conf` file:

```
acl badsites url_regex -i "/etc/squid/badsites.txt"
http_access deny badsites
```

It should be noted that the placement of *http_access* lines are very important in SQUID's expected behavior. You need to deny *badsites* BEFORE allowing the *http_access* to your client src acl.

Your configuration with blocked sites would look like this:

```
acl badsites url_regex -i "/etc/squid/badsites.txt"
acl mynet src 192.168.1.0/24

http_access deny badsites

http_access allow mynet

http_access allow localhost
http_access deny all
```

Your badistes file should contain all words on separate lines which you want to be search in the URL and then blocked by SQUID. An example `badsites.txt` file is below:

sex
xxx
movies

This is quite effective approach. You may need to watch your `/var/log/squid/access.log` for such sites and add those names to this file.

Socks, did I forget to wear them?

Unfortunately, your clients will not be able to find a SOCKS server on linux machine for their MSN or Yahoo messengers. To support that you have two options.

a) Install and configure a SOCKS package.
b) Enable IP Masquerading (NAT) on your network (easy and recommended).

a) Install and configure SOCKS server.

SOCKS stands for SOCK-et-S , its initial internal development name. There is also a "Nylon socks" project underway on the internet. SOCKS is a networking proxy protocol that enables hosts on one side of a SOCKS server to gain full access to hosts on the other side of the SOCKS server without requiring direct IP-reach ability. SOCKS is often used as a network firewall, redirecting connection requests from hosts on opposite sides of a SOCKS server. The SOCKS server authenticates and authorizes requests, establishes a proxy connection, and relays data between hosts.

Quite some information resides on these web pages. Please see to them.

```
www.linuxgazette.com/issue48/tag/27.html
www.host.biz/rpm2html/PLF/System_Servers.html
www.socks.nec.com/socksfaq.html
www.socks.nec.com
```

b) To enable IP masquerading you need to put the following two lines in one of your startup scripts (`rc.local`) or optionally you can create your own script and call it from `/etc/rc.local` script.

For *kernel 2.2* using `ipchains`, only put the following two lines :-

```
[root@mainserver /]# echo "1" > /proc/sys/net/ipv4/ip_forward

[root@mainserver /]# ipchains -A FORWARD -i ppp0 -s 192.168.1.0/24 -d
        ➥       ! 192.168.1.0/24 -j MASQ
```

For *kernel 2.4* using netfilter (`iptables`) only put the following two lines :-

```
[root@mainserver /]# echo "1" > /proc/sys/net/ipv4/ip_forward

[root@mainserver /]# iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -d
        ➥       ! 192.168.1.0/24 -o ppp0 -j MASQUERADE
```

Common problems:

1) Access denied to SQUID even the `src` ACL is defined and the URL doesn't contain any word which should get caught in any of the badsites ACL.

This happens when there are two (or more) networks which are (mis-)configured in SQUID to be able to access SQUID cache data. For example, the following ACLconfiguration will never work:

```
acl MYNET_A src 192.168.1.0/24
acl MYNET_B src 192.168.0.0/24
http_access allow MYNET_A MYNET_B
```

The above should be written as :

```
acl MYNET_A src 192.168.1.0/24
acl MYNET_B src 192.168.0.0/24
http_access allow MYNET_A
http_access allow MYNET_B
```

Or even this will work:

```
acl MYALL_NETS src 192.168.1.0/24 192.168.0.0/24
http_access allow MYALL_NETS
```

2) SQUID dead but subsystem locked.

This happens because of some configuration problem in `squid.conf` file, which puts it to stop just after it is started. Check `/var/log/messages` for any system level or squid related errors.

3) No running copy

Normally happens when squid was not able to create the cache directory in the previous steps or is unable to access the cache directories due to ownership / permissions problems. The cache directories must be owned by both user and group **`squid`** and should be writeable by **`squid:squid`** only. May also be resultant of bad DNS lookups. In bad DNS case, run squid with -D switch.

Log files:

```
/var/log/messages
/var/log/squid/access.log
/var/log/squid/cache.log
```

Documentation and Help:

Extensive help (specially check the FAQ) is available on `www.squid-cache.org`
Also at `www.visolve.com.` FAQ files in `/usr/share/doc/squid-a.b`

Testing:

On the command prompt :

```
[root@mainserver /]# /etc/rc.d/init.d/squid status
```

You can check the `access.log` file for user requests for web pages being handles by SQUID, using a separate window or terminal:

```
[root@mainserver /]# tail -f /var/log/squid/access.log
```

Chapter 9: APACHE

Apache is actually "a *PATCH*y server". It was based on some existing code and a series of "patch files" made to original NCSA server. For many people this name brings the same meaning / feeling / spirit of the Native American Indian tribe of Apache, well-known for their superior skills in warfare strategy and inexhaustible endurance. Apache is HTTP 1.1 Compliant, open-source web server that is widely used.

Packages :

Redhat 7.x

```
apache-a.b
apacheconf-a.b
apache-manual-a.b
```

Redhat 8.0 & 9

```
httpd-a.b
httpd-manual-a.b
redhat-config-httpd-a.b (gui)
```

Scenario:

In our example network, we want to implement a local web server under the name `www.mydomain.com` . I will also introduce *`ScriptAlias , Alias`* and *`UserDir`* directives. The same server `www.mydomain.com` will be used to host two virtual websites: `www.islam.edu` and `www.urdulinux.org`. All of these websites will be hosted on **`mainserver.mydomain.com`** having IP **`192.168.1.254`**. We will implement it using Name based virtual hosting. Normally you should consider using Name based virtual hosting unless there is a specific need / reason to choose IP based virtual hosting.

Configuration

Apache configuration file is `/etc/httpd/conf/httpd.conf`. The file is divided into three main sections.

1. Global Environment
2. Main configuration
3. Virtual Host Configuration

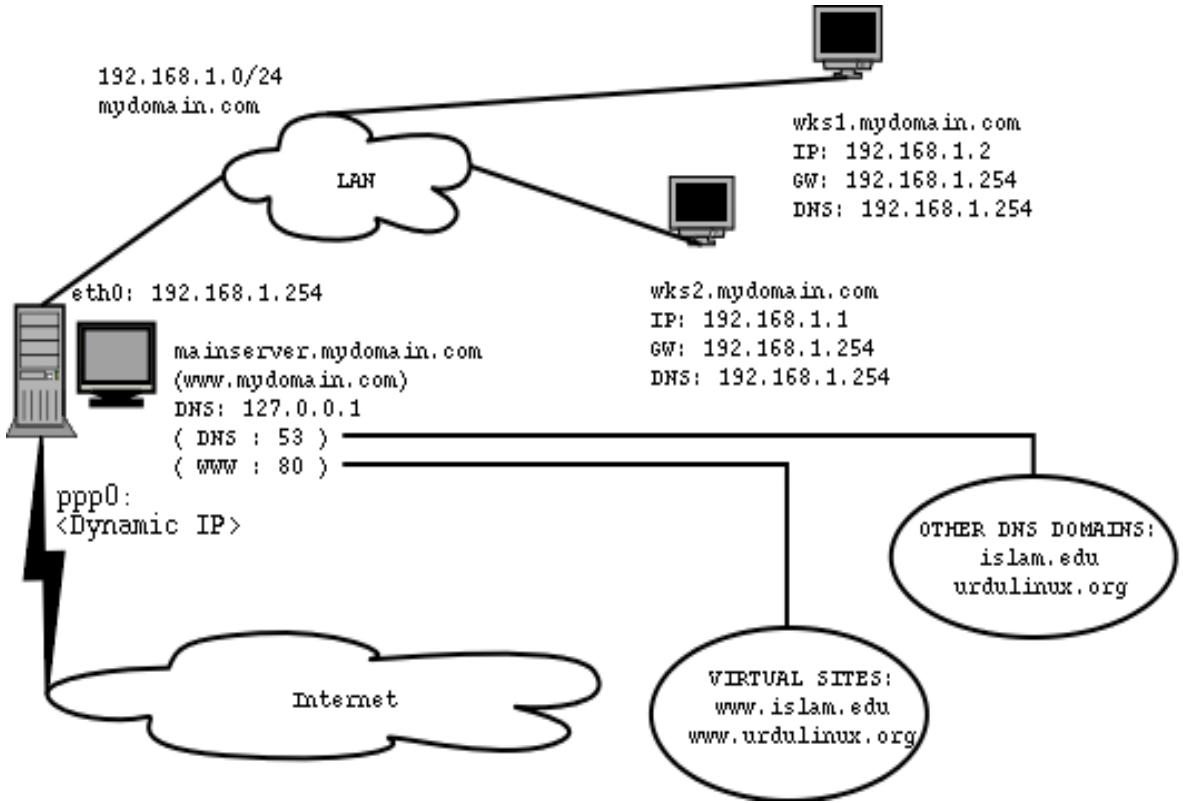

192.168.1.0/24
mydomain.com
LAN
wks1.mydomain.com
IP: 192.168.1.2
GW: 192.168.1.254
DNS: 192.168.1.254
eth0: 192.168.1.254
wks2.mydomain.com
IP: 192.168.1.1
GW: 192.168.1.254
DNS: 192.168.1.254
mainserver.mydomain.com
(www.mydomain.com)
DNS: 127.0.0.1
(DNS : 53)
(WWW : 80)
ppp0:
<Dynamic IP>
OTHER DNS DOMAINS:
islam.edu
urdulinux.org
Internet
VIRTUAL SITES:
www.islam.edu
www.urdulinux.org

Fig 1

In section 1, only the directive **`Listen`** needs to be set. This tells the Apache server, on which IP and port to listen for HTTP requests.

```
Listen 80
or
Listen 192.168.1.254:80
```

We can also make it listen on any port other than 80.

In section 2, set the `ServerName` and `DocumentRoot` directives. `ServerName` would be the name which will be used generally to access your server , i.e. `www.mydomain.com` . Set `DocumentRoot` directive to the path where the web pages of your main website reside. Similarly your `VirtualHost` configurations in section 3 will have the **`DocumentRoot`** directive set to the directories containing web content relating to respective sites sites.

```
ServerName www.mydomain.com

DocumentRoot "/var/www/html"
```

Also the default directory options for the document root directory are :

```
<Directory "/var/www/html">
        Options Indexes FollowSymLinks
        AllowOverride None
        Order allow,deny
        Allow from all
</Directory>
```

Try starting your web server now by:

```
[root@mainserver /]# /etc/rc.d/init.d/httpd start
```

Open your favorite web browser and type in the URL :

```
http://127.0.0.1
or
http://localhost
or
http://www.mydomain.com
```

You should be able to see a Apache web server Test Page. If you can see the Apache test web page that means you are all set to further configure your server. Now copy your web site related files in this directory `"/var/www/html"` make them Apache readable by setting appropriate ownership and permissions and and re-start the web server. You should be able to see your website related main web page.

Directory aliases

Assume you want to provide your users with some helpful documentation on various technologies, stored on your local hard drive. Assuming this local storage is `/data/ebooks` and you wish that your users should be able to access it as `www.mydomain.com/ebooks` , It should be kept in mind that your document home directory for `www.mydomain.com` is `/var/www/html` .

One method is to create a directory named `ebooks` under `/var/www/html` and move or copy the contents of `/data/ebooks` to `/var/www/html/ebooks` . This is cumbersome and is prone to disk space wastage.

The other method is to create a directory alias for this location like:

```
Alias /ebooks/ "/data/ebooks/"

<Directory “/data/ebooks">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
```

Restart your Apache service and you should be able to access your *ebooks* from the URL:
`www.mydomain.com/ebooks .`

ScriptAliases

Sometimes it is desired that you have another directory for your CGI scripts other than
`/var/www/cgi-bin` . For that you have special directive `ScriptAlias` .
Assume you have `/var/www/cgi-admin` directory as well, and you want to run your CGI scripts from there too.

```
ScriptAlias /cgi-admin/ "/var/www/cgi-admin/"

<Directory "/var/www/cgi-admin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>
```

UserDir*ectories*

This is additional feature which you may want to introduce in your web server configuration. This facility allows users to have their web contents published on your web server directly from their home directories and without requiring any administrative privileges or without the need to edit the `httpd.conf` file.

Using this feature anyone can access the web content published by any user by using the URL:
`www.mydomain.com/~user` . e.g. A user *Usama*'s web content can be viewed by :
`www.mydomain.com/~usama` .

As per `httpd.conf` file: *"The path to the end user account '`public_html`' directory must be accessible to the webserver userid. This usually means that `~userid` must have permissions of 711, `~userid/public_html` must have permissions of 755, and documents contained therein must be world-readable. Otherwise, the client will only receive a "403 Forbidden" message."*

This means that each user who wants to publish his web content should have his home directory permissions (`/home/usama`) set as `711` and should create a special directory named "`public_html`" (`/home/usama/public_html`) with permissions `755`. Any material / content placed in this directory should have world readable permissions.

The web server administrator (you) will have to set the user home directory permissions to 711 and edit the `httpd.conf` to enable user directory feature. To do so remark the line **`UserDir disable`** in the section *`<IfModule mod_userdir.c>`* and un-remark the line **`UserDir public_html`** in the same section.

```
<IfModule mod_userdir.c>
    #
    # UserDir is disabled by default since it can confirm the presence
    # of a username on the system (depending on home directory
    # permissions).
    #
    # UserDir disable

    #
    # To enable requests to /~user/ to serve the user's public_html
    # directory, remove the "UserDir disable" line above, and uncomment
    # the following line instead:
    #
    UserDir public_html

</IfModule>
```

More over you can setup the user public_html directories as read only to avoid any future problems.

```
# Control access to UserDir directories.  The following is an example
# for a site where these directories are restricted to read-only.

<Directory /home/*/public_html>
    AllowOverride FileInfo AuthConfig Limit
    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
    <Limit GET POST OPTIONS>
        Order allow,deny
        Allow from all
    </Limit>
    <LimitExcept GET POST OPTIONS>
        Order deny,allow
        Deny from all
    </LimitExcept>
</Directory>
```

Your main (basic) configuration of your site `www.mydomain.com` finishes here. Now lets see what needs to be done for hosting the two example websites discussed above. i.e. `www.islam.edu` and `www.urdulinux.org` .

Virtual site hosting needs configuration in your DNS first. For these two sites, we need to define them in our DNS. There is yet another problem. These sites do not fall under our network forward zone data. This means we need to define separate forward zones (define separate domains) in our DNS. Since in these new forward zones, there needs to be only on “NS” , one “A” and one “CNAME” type entries, this is going to do just fine.

So first , here is the `/etc/named.conf` file :

```
[root@mainserver /]# vi /etc/named.conf

## named.conf - configuration for bind
#
# Generated automatically by redhat-config-bind, alchemist et al.
# Any changes not supported by redhat-config-bind should be put
# in /etc/named.custom
#
controls {
        inet 127.0.0.1 allow { localhost; } keys { rndckey; };
};

include "/etc/rndc.key";

options {
         directory "/var/named/";
};

zone  "." {
         type hint;
         file  "named.ca";
};
```

```
zone "localhost" {
        type master;
        file "localhost.zone";
};

zone "0.0.127.in-addr.arpa" {
        type master;
        file "0.0.127.in-addr.arpa.zone" ;
};

zone  "1.168.192.in-addr.arpa" {
        type master;
        file  "1.168.192.in-addr.arpa.zone";
};

zone  "mydomain.com" {
        type master;
        file  "mydomain.com.zone";
};

zone  "islam.edu" {
        type master;
        file  "islam.edu.zone";
};

zone  "urdulinux.org" {
        type master;
        file  "urdulinux.org.zone";
};
```

Now here is the forward zone of my domain ***mydomain.com*** . Notice the absence of any mentioning of `www.islam.edu` or `www.urdulinux.org` . Also notice that there is no separate IP for `www`, rather `www` machine is in-fact canonical name for the actual machine named `mainserver`.

[root@mainserver /]# vi /var/named/mydomain.com.zone

```
$ORIGIN mydomain.com.
$TTL 3h
@ IN SOA mainserver.mydomain.com. root.mainserver.mydomain.com. (
                                        1 ; serial
                                        3H ; Refresh after 3 hours
                                        1H ; Retry after 1 hour
                                        1W ; Expire after 1 week
                                        1H ) ; negative caching of TTL of 1 hr

@       IN      NS      mainserver.mydomain.com.
@       IN      MX      10      mainserver.mydomain.com.

mainserver      IN      A               192.168.1.254
ws1             IN      A               192.168.1.1
ws2             IN      A               192.168.1.2
www             IN      CNAME           mainserver
```

Below is what is to be be configured for `www.islam.edu :`

[root@mainserver /]# vi /var/named/islam.edu.zone

```
$ORIGIN islam.edu.
$TTL 3h
@ IN SOA mainserver.mydomain.com. root.mainserver.mydomain.com. (
```

```
                                        1 ; serial
                                        3H ; Refresh after 3 hours
                                        1H ; Retry after 1 hour
                                        1W ; Expire after 1 week
                                        1H ) ; negative caching of TTL of 1 hr

@       IN      NS      mainserver.mydomain.com.

mainserver      IN      A               192.168.1.254
www             IN      CNAME           mainserver
```

The SOA record of this file may be confusing for some readers. Since there is only one name server in our network which is doing all the work, we need to mention that name server in the SOA record, so that any name queries performed on our name server could tell the querying application / machine, that who is the actual machine resolving all queries. In simple words , the SOA above says that “Start Of Authority for the domain `islam.edu` is the machine `mainserver.mydomain.com`.”

For the domain `urdulinux.org` we will of-course use the same format as we have used for `islam.edu` , there is no “rocket science” involved in it.

```
[root@mainserver /]# vi /var/named/urdulinux.org.zone

$ORIGIN urdulinux.org.
$TTL 3h
@ IN SOA mainserver.mydomain.com. root.mainserver.mydomain.com. (
                                        1 ; serial
                                        3H ; Refresh after 3 hours
                                        1H ; Retry after 1 hour
                                        1W ; Expire after 1 week
                                        1H ) ; negative caching of TTL of 1 hr

@               IN      NS      mainserver.mydomain.com.

mainserver      IN      A               192.168.1.254
www             IN      CNAME           mainserver
```

Virtual Host:

Now our forward zones properly configured, lets configure section 3 of our Apache web server for Virtual Hosting.

```
[root@mainserver /]# vi /etc/httpd/conf/httpd.conf

NameVirtualHost 192.168.1.254

<VirtualHost www.mydomain.com>
        ServerName www.mydomain.com
        DocumentRoot /var/www/html
</VirtualHost>

<VirtualHost www.islam.edu>
        ServerName www.islam.edu
        DocumentRoot /hostedsites/islam.edu/html
</VirtualHost>
```

```
<VirtualHost www.urdulinux.org>
        Servername www.urdulinux.org
        DocumentRoot /hostedsites/urdulinux.org/html
</VirtualHost>
```

You can also re-write the `VirtualHost` section for any virtual site as :

```
<VirtualHost 192.168.1.254>
        ServerName www.islam.edu
        DocumentRoot /hostedsites/islam.edu/html
</VirtualHost>
```

Notice that we have introduced the virtual host section for `www.mydomain.com` as well. This is because in case of miss-spelled URLs and sites which are not hosted on this server, Apache goes to the first virtual host defined in the virtual hosts section. See point 1 below.

Using an explicit IP address to listen HTTP requests in `NameVirtualHost` is better. BUT if your web server is on dynamic IP, i.e. it gets it's IP via DHCP or so, then you should not use a fixed IP. In that case you should use an `*`. Only **`ServerName`** and **`DocumentRoot`** directives are a must while configuring virtual hosts.

There are few more points to keep under consideration while configuring / hosting your web sites.

1) If a URL is typed in by the user in his web browser is mis-spelled or incorrect, then by default, Apache will use the virtual host listed first to display the page. i.e. In the above configuration, if the user types in `www.linuxdoc.mydomain.com` in his web browser, then since this URL / website does not exist in your domain tables, the first virtual host (`www.islam.edu`) will be used to display the web page, which was not intended. In order to avoid such problems, you can put a sort of dummy virtual host containing a redirector page or a page telling user that he has not reached where he wished, and that he should type in the address again.

2) Any directives defined outside the virtual host configuration are over written / neglected if the same directives are found inside that specific virtual host. eg. you have a `DocumentRoot` directive in the main configuration file as :

```
DocumentRoot "/var/www/html"
```

But this will not be honored in the virtual host `www.islam.edu` because that virtual host section has it's own `DocumentRooot` directive as :

```
        DocumentRoot /hostedsites/islam.edu/html
```

Providing support of privileged access to some web site, based on user name and password

Sometimes a user needs to be authenticated before displaying the contents of a web page. There are many different ways for doing it but I will show you the basic method here. Assume your web site *urdulinux.org* is undergoing some refinement and you want it only accessible by it's developer tauseef using his password “bravo”. The web pages of the website under discussion are located in the directory: `/hostedsites/urdulinux.org/html` . We need to make a user authentication file (password file) with the help of a utility *`htpasswd`*. This file will be placed outside the web pages directory (i.e. in the root-only readable directory `/etc/httpd/security`) so that cracking is avoided.

1) Create the file (if doesn't exist) and the user "tauseef" by :

```
[root@mainserver /]# mkdir /etc/httpd/security
[root@mainserver /]# htpasswd -c /etc/httpd/security/.htpassword tauseef
```

The dot in the beginning of the file makes it hidden so it won't show up in directory listing. Type the password twice to confirm. Also change it's ownership so that user Apache is able to read it, by:

```
[root@mainserver /]# chown apache:apache /etc/httpd/security/.htpassword
```

2) Create a `.htaccess` file with the following contents in the same directory which contains your web pages. i.e. `/hostedsites/urdulinux.org/html` , with the command :-

```
[root@mainserver /]# vi /hostedsites/urdulinux.org/html/.htaccess

AuthName "Authorized project developers only"
AuthType Basic
AuthUserFile /etc/httpd/security/.htpassword
require user tauseef
```

3) Change the file permissions of the `.htaccess` file so that only the user Apache is able to change it.

```
[root@mainserver /]# chown apache:apache /hostedsites/urdulinux.org/html/.htaccess
```

4) Also give only read permissions to users on all files in the web pages directory by:-

```
[root@mainserver /]# chmod o+r /hostedsites/urdulinux.org/html/*
[root@mainserver /]# chmod o-wx /hostedsites/urdulinux.org/html/*
```

or

```
[root@mainserver /]# chmod 774 /hostedsites/urdulinux.org/html/*
```

5) Now update the `httpd.conf` file and set the security options for your `DocumentRoot` directory in the `VirtualHost` section of `urdulinux.org`.

```
<VirtualHost www.urdulinux.org>
      Servername www.urdulinux.org
      DocumentRoot /hostedsites/urdulinux.org/html
         <Directory "/hostedsites/urdulinux.org/html"
                  AllowOverride All
         </Directory>
</VirtualHost>
```

6) Now is the time to test your setup. First restart the `httpd` deamon by

```
[root@mainserver /]# service httpd restart
```

Open the browser at the server and point to `www.urdulinux.org` and press enter. (You may need to reload the page). A popup screen should come up asking username and password. Supply username **tauseef** and password **bravo**. It should allow access, else error 401 web page will show up , which is "Authorization failed".

That is all.

Log Files

```
/var/log/httpd/access_log
/var/log/httpd/error_log
/var/log/messages
```

Documentation and Help:

Extensive documentation is available on the Apache Test web page once you simply start your apache web server without configuring it for the first time. Simpler path is `/var/www/manual/index.html`

Also the files in directory `/usr/share/doc/httpd`, and `www.apache.org/docs` website.

Chapter 10: SENDMAIL

SENDMAIL is world's #1 and most widely used MTA (Mail Transfer Agent).

Packages

`sendmail-a.b` (on server side)
`sendmail-cf-a.b` (on server side)
`sendmail-doc-a.b` (on server side)
`imap-a.b` (on server side, to install both pop3 and imap services)

Scenario

We want to establish a mail server in our example network. We will provide a mailing system so that our linux clients can send and receive inter-office (within the *mydomain.com*) mails as well as mails coming and and going out to rest of the world, while connected to the server via NIS and our windows clients will be able to send and receive mails using SMTP, POP and/or IMAP services. In short it is assumed that all the users have their mail accounts on the mail server, so they have to connect to the server, to send and receive mails. They can send mail using our mail server as a relay even if they don't have an account on our mail server. In our example network, our domain name is mydomain.com. Our mail server is `mainserver.mydomain.com` on the IP `192.168.1.254` . We have SMPT, POP3 and IMAP services running on our mail server.

Configuraiton

It is said in *nix world that "***You are not a system administrator if you have not configured sendmail once in your life. And you are crazy if you have tried to do it twice"**. Well, that used to be said once ***m4*** macro utility was not known. Sendmail configuration is possible using one of three methods:

1)By using **linuxconf** utility, which is now obsolete.
2)By using **m4** macro configuration utility
3)By editing the sendmail.cf file **manually**

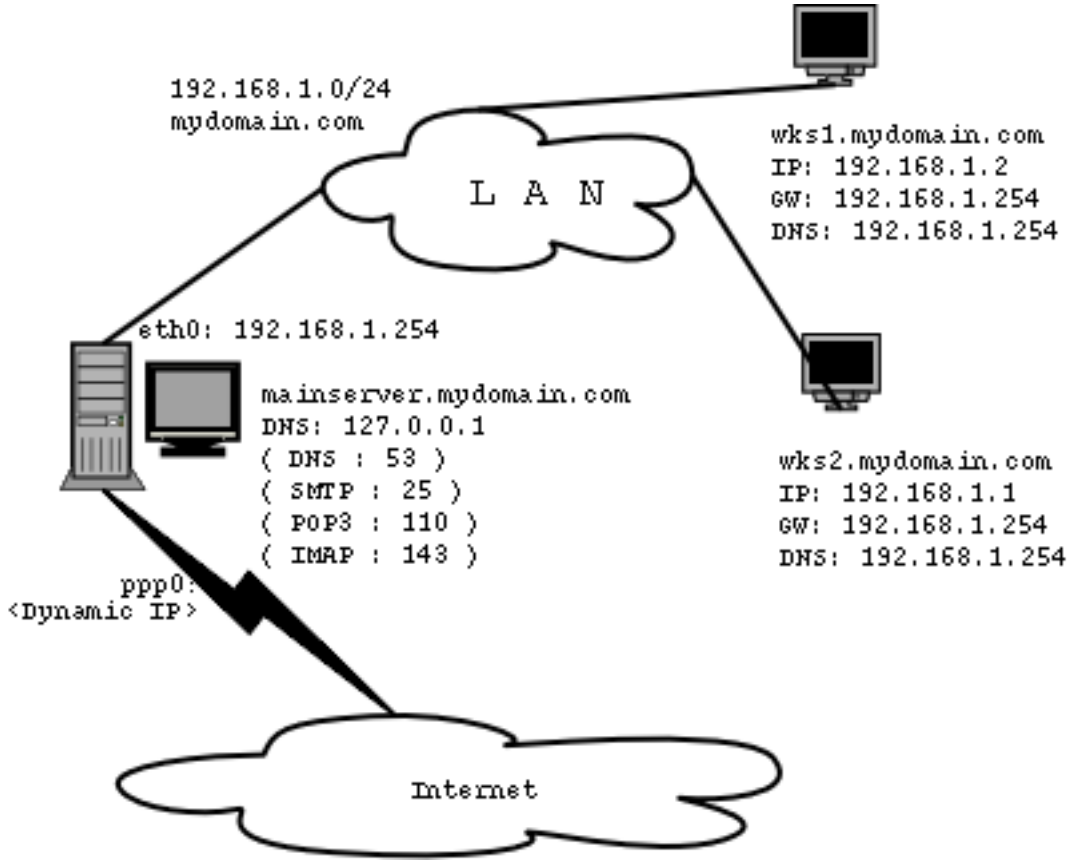

192.168.1.0/24
mydomain.com
L A N
wks1.mydomain.com
IP: 192.168.1.2
GW: 192.168.1.254
DNS: 192.168.1.254
eth0: 192.168.1.254
mainserver.mydomain.com
DNS: 127.0.0.1
(DNS : 53)
(SMTP : 25)
(POP3 : 110)
(IMAP : 143)
wks2.mydomain.com
IP: 192.168.1.1
GW: 192.168.1.254
DNS: 192.168.1.254
ppp0:
<Dynamic IP>
Internet

Fig 1

First of all you should have a proper MX record set up for your mail server in the DNS zone files. Refer to the chapter DNS for this. In short , in addition to NS record you should have lines as following in your forward zone (`mydomain.com.zone`) file.

```
@       IN      NS      mainserver.mydomain.com.
@       IN      MX      10      mainserver.mydomain.com.
```

The number 10 in the example above is the priority of this mail server in case there are more than one mail server in your setup. The lesser the number, the higher the priority.

The configuration file **sendmail.cf** used to be found in `/etc`. In newer versions like Redhat Linux 8.0 and 9, it is found in `/etc/mail` directory.

The first method I am going to use is by **m4** macro utility. There is a file present by default in your `/etc/mail` directory by the name `sendmail.mc` . Open this file and only change the lines which are shown below:

```
[root@mainserver /]# vi /etc/mail/sendmail.mc
```

```
dnl DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
CLIENT_OPTIONS(`Family=inet,Address=192.168.1.0')dnl
MASQUERADE_AS(`mydomain.com')dnl
MASQUERADE_DOMAIN(localhost)dnl
MASQUERADE_DOMAIN(localhost.localdomain)dnl
```

The changes above indicate that SMTP daemon will listen on all interfaces of this machine. Notice that I have removed the `127.0.0.1` restriction by commenting the DAEMON_OPTIONS line. Similarly I have introduced my network address `192.168.1.0` in the CLIENT_OPTIONS indicating that only machines of my network can use the SMTP services of this machine. Any mail coming from `localhost` or `localhost.localdomain` as well as from any client machine will be translated / re-written as from **`mydomain.com`** .

Now generate the `sendmail.cf` file by :

```
[root@mainserver /]# m4 /etc/mail/sendmail.mc > /etc/mail/sendmail.cf
```

Also open the `/etc/mail/local-host-names` file and add all possible names of your mail server on a new line in this file.

```
[root@mainserver /]# vi /etc/mail/local-host-names
mydomain.com
```

```
mainserver.mydomain.com
localhost
localhost.localdomain
```

By default your client computers will be denied relaying. That is that by default your client computers / users will not be able to send any mail destined outside `mydomain.com`. To enable this feature you can use **`/etc/mail/access.db`** file . This file is binary and cannot be directly edited. So you have to edit it's text version (**`/etc/mail/access`**) and then generate the binary file from it.

```
[root@mainserver /]# vi /etc/mail/access
192.168.1                RELAY
mydomain.com             RELAY
```

Generate the binary *access.db* file from this file by :

```
[root@mainserver /]# makemap hash /etc/mail/access.db < /etc/mail/access
```

Start / restart your sendmail server by :

```
[root@mainserver /]# service sendmail start
```

The other method to configure sendmail is by *manually* editing `sendmail.cf` . Open it with any text editor and change as following:

```
[root@mainserver /]# vi /etc/mail/sendmail.cf

##################
#   local info   #
##################

# my LDAP cluster
# need to set this before any LDAP lookups are done (including classes)
#D{sendmailMTACluster}$m

Cwlocalhost mainserver.mydomain.com mydomain.com    ; (alternate / possible names
for this machine)

# file containing names of hosts for which we receive email
Fw/etc/mail/local-host-names

# my official domain name
# ... define this only if sendmail cannot automatically determine your domain
#Dj$w.Foo.COM

CP.

# "Smart" relay host (may be null)
Dsmainserver.mydomain.com          # (machine handling all the mail, optional)


# class E: names that should be exposed as from this host, even if we masquerade
# class L: names that should be delivered locally, even if we have a relay
```

```
# class M: domains that should be converted to $M
# class N: domains that should not be converted to $M
#CL root
C{E}root
C{M}localhost
C{M}localhost.localdomain #( Masquerade what ?)

DMmydomain.com    # (Masquerade as what domain? Emails will have this on the right
side of @ sign)

# my name for error messages
DnMAILER-DAEMON
```

Then somewhere in the middle of the file, under the section OPTIONS, search for SMTP daemon options and modify (disable by putting remark) as follows:

```
###############
#   Options   #
###############


# SMTP daemon options

#O DaemonPortOptions=Port=smtp,Addr=127.0.0.1, Name=MTA
#or
O DaemonPortOptions=Name=MTA


# SMTP client options
O ClientPortOptions=Family=inet, Address=192.168.1.0
```

That's it. Save the file and start the service by:

```
[root@mainserver /]# /etc/rc.d/init.d/sendmail start
```

or

```
[root@mainserver /]# service sendmail restart
```

Configuration of IMAP and POP3 services:

Kernel 2.2:
Edit the file `/etc/inetd.conf` and find the remarked lines with `pop-3` and `imap`. Un-remark them (enable them by removing the remark so they look like following:

```
[root@mainserver /]# vi /etc/inetd.conf

pop-3    stream   tcp      nowait   root     /usr/sbin/tcpd   ipop3d
imap     stream   tcp      nowait   root     /usr/sbin/tcpd   imapd
```

These services are listed in the file `/etc/services` with their respective port numbers. For `pop3` it is 110 and for imap it is 143.

Once done modifying, restart *inetd* by:

```
[root@mainserver /]# killall -HUP inetd
```

Kernel 2.4
In the */etc/xinetd.d* directory you have separate files dedicated for each service. This is unlike the `inetd.conf` setup in kernel 2.2 , in which all services are listed in one file. So edit the respective files in the `/etc/xinetd.d` directory and change the **`disable=yes`** line to **`disable = no`**, as follows:

```
[root@mainserver /]# vi /etc/xinetd.d/ipop3

# default: off
# description: The POP3 service allows remote users to access their mail \
#              using an POP3 client such as Netscape Communicator, mutt, \
#               or fetchmail.
service pop3
{
        disable  = no
        socket_type             = stream
        wait                    = no
        user                    = root
        server                  = /usr/sbin/ipop3d
        log_on_success   += HOST DURATION
        log_on_failure   += HOST
}
```

Similarly modify the `imap` file as well as :

```
[root@mainserver /]# vi /etc/xinetd.d/imap

# default: off
# description:The IMAP service allows remote users to access their mail using\
#             an IMAP client such as Mutt, Pine, fetchmail, or Netscape \
#             Communicator.
```

```
service imap
{
        disable  = no
        socket_type             = stream
        wait                    = no
        user                    = root
        server                  = /usr/sbin/imapd
        log_on_success    += HOST DURATION
        log_on_failure    += HOST
}
```

Now restart the `xinetd` services by:

```
[root@mainserver /]# service xinetd restart
```

We will see in the *testing* section that how can we verify if these services are up and working.

Client side configuration:

Windows clients:

Configuring windows clients is not difficult. Open any email program like outlook express or Netscape mail. Create a new account for POP3 based service. Type in the user name and password you have on the mail server. In the outgoing SMTP server type in the IP address of your mail server as 192.168.1.254 on port 25, or if the DNS is working properly, then you can also type in the name of the server as `mainserver.mydomain.com` on port 25. Select the POP3 receive mail server and specify your mail server as `192.168.1.254` on port `110` or `mainserver.mydomain.com` on port `110`. Similar configuration can be used to configure an IMAP account. But port number for IMAP server is 143.

Linux clients:

Linux clients using X windows system running GNOME or KDE (or any other) can again configure their accounts as they would have in microsoft windows tools. You can use KMAIL, or Evolution. The text clients can use ***`pine`*** or ***`mail`*** or other tools available widely for this purpose. The settings are similar as above. Create a new account for POP3 based service. Type in the user name and password you have on the mail server. In the outgoing SMTP server type in the IP address of your mail server as 192.168.1.254 on port 25, or if the DNS is working properly, then you can also type in the name of the server as `mainserver.mydomain.com` on port 25. Select the POP3 receive mail server and specify your mail server as `192.168.1.254` on port `110` or `mainserver.mydomain.com` on port `110`.

Similar configuration can be used to configure an IMAP account. But port number for IMAP server is 143.

Log files:

```
/var/log/messages
/var/log/maillog
```

Documentation and help

`/usr/share/doc/sendmail` (directory)

The website and FAQs on the website `www.sendmail.org`

On `www.tldp.org/HOWTO` search for :-

1)The Linux Electronic Mail Administrator HOWTO
2)The Linux Mail User HOWTO
3)Chapter 17 and 18 in the book Linux Network Administrator Guide 2nd Edition on the guides page on `www.tldp.org`

Testing:

After setting up the `sendmail.cf` file by whatever method, and setting up the POP3 and IMAP services, you should be able to connect to the server using telnet, from server and client machines. Telnet service (on port 23) is not mandatory for these tests.

To test the SMTP server, use:

```
[root@mainserver /]# telnet 192.168.1.254 25
or
[root@mainserver /]# telnet mainserver.mydomain.com 25
```

You should see a message showing the SMTP server is responding.

```
Trying 192.168.1.254...
Connected to mainserver.mydomain.com (192.168.1.254).
```

```
Escape character is '^]'.
220 mainserver.home ESMTP Sendmail 8.12.5/8.12.5; Sun, 10 Nov 2002 15:37:47 +0500
```

Press ^]here and type quit on the `telnet>` prompt.

```
^]

telnet> quit
Connection closed.
```

Now to test the POP3 and IMAP servers, use the same telnet command using port `110` and `143` respectively.

```
[root@mainserver /]# telnet mainserver.mydomain.com 110

Trying 192.168.1.254...
Connected to mainserver.mydomain.com (192.168.1.254).
Escape character is '^]'.
+OK POP3 [192.168.1.254] v2001.78rh server ready
^]

telnet> quit
Connection closed.


[root@mainserver /]# telnet mainserver.mydomain.com 143

Trying 192.168.1.254...
Connected to mainserver.mydomain.com (192.168.1.254).
Escape character is '^]'.
* OK [CAPABILITY IMAP4REV1 LOGIN-REFERRALS STARTTLS AUTH=LOGIN] [192.168.1.254]
IMAP4rev1 2001.315rh at Sun, 10 Nov 2002 15:43:57 +0500 (PKT)
^]

telnet> quit
Connection closed.
```

Sending mail using Sendmail directly, using telnet:

Below is a telnet session which will help you understand how to send email directly using *Sendmail*.

```
[root@mainserver /]# telnet mainserver.mydomain.com 25
Trying 192.168.1.254...
Connected to mainserver.mydomain.com (192.168.1.254).
Escape character is '^]'.
220 mainserver.mydomain.com ESMTP Sendmail 8.12.5/8.12.5; Fri, 18 Jul 2003
01:00:14 +0600
helo mydomain.com
250 mainserver.mydomain.com Hello mainserver.mydomain.com [192.168.1.254], pleased
to meet you
mail from: kamran@mydomain.com
250 2.1.0 kamran@mydomain.com... Sender ok
rcpt to: mkazeem@isb.paknet.com.pk
250 2.1.5 mkazeem@isb.paknet.com.pk... Recipient ok (will queue)
data
354 Enter mail, end with "." on a line by itself
This is a test mail being sent from lab environment to my email address !
.
250 2.0.0 h6HJ0D2i009597 Message accepted for delivery
```

```
quit
221 2.0.0 mainserver.mydomain.com closing connection
Connection closed by foreign host.
```

In the scenario below I am connected to my ISP *wtmeca.net* to send mail to my Paknet email-address.

```
[root@mainserver /]# telnet  mail.wtmeca.net 25
Trying 202.179.143.13...
Connected to mail.wtmeca.net (202.179.143.13).
Escape character is '^]'.
hel220 mail.wtmeca.net ESMTP Sendmail 8.12.4/8.12.4; Thu, 21 Nov 2002
14:45:57 +0500
helo
501 5.0.0 helo requires domain address
helo isb.paknet.com.pk
250 mail.wtmeca.net Hello [202.179.156.106], pleased to meet you
mail from: kamran@isb.paknet.com.pk
250 2.1.0 kamran@isb.paknet.com.pk... Sender ok
rcpt to: kamran@hotmail.com
250 2.1.5 kamran@hotmail.com... Recipient ok
data
354 Enter mail, end with "." on a line by itself
test mail
.
250 2.0.0 gAL9jvCu006927 Message accepted for delivery
quit
221 2.0.0 mail.wtmeca.net closing connection
Connection closed by foreign host.
```

And below is yet another telnet session, this time with my Paknet mail server which had some error in it's configuration. This error was pointed to the system administrator of Paknet and was rectified.

```
[root@mainserver /]# telnet isb.paknet.com.pk 25
Trying 203.135.0.3...
Connected to isb.paknet.com.pk (203.135.0.3).
Escape character is '^]'.
220 Welcome to Paknet Mail Services.Please do not ABUSE MAIL SERVER
ESMTP
helo isb.paknet.com.pk
250 Welcome to Paknet Mail Services.Please do not ABUSE MAIL SERVER
mail from: kamran@isb.paknet.com.pk
250 ok
rcpt to: kamran@hotmail.com
553 sorry, that domain isn't in my list of allowed rcpthosts (#5.7.1)
rcpt to: inam@isb.paknet.com.pk
250 ok
data
354 go ahead
Test mail using paknet server, having God knows what kind of settings at paknet
end !!!
.
250 ok 1037872119 qp 15513
quit
221 Welcome to Paknet Mail Services.Please do not ABUSE MAIL SERVER
Connection closed by foreign host.
```

If at any instance the connection is refused then check :

1)the inetd or xinetd services, they should be enabled

2)check the sendmail.cf file for any errors
3)check the firewall settings. There might be firewall rules which are blocking connections to the mail server

While connecting from client machines, using mail programs like outlook or Netscape mail, you might encounter problems like ***cannot find server*** **`mainserver.mydomain.com`**. If this is the case then try the IP address of the mail server in place of the name. If this works then it means that your DNS is not properly configured. Check for MX records in the `mydomain.com.zone` file. Also check the reverse zone file for your domain.

It should be noted here that there are various configuration files / databases in the `/etc/mail` directory.

These are:

`access`	List of machines that can use sendmail for outbound mail.
`domaintable`	Specifies domain name mapping.
`local-host-names`	All possible names of the mail server for which the mail server should receive mails for.
`mailertable`	Special instructions for mail routing between various domains.
`virtusertable`	Mainly used in virtual hosting.
`relay-domains`	Used to permit any domain to relay mail using this mail server, denied by default. Like y.com wants to send mail to z.org using your mail server.

Relaying denied error

Normally the first (and worst) problem for an entry-level system administrator. Shakes him once at least. Thankfully, not much a problem.

Re-create `/etc/mail/access.db` file by the following procedure:

```
[root@mainserver /]# vi /etc/mail/access

localhost.localdomain                   RELAY
localhost                               RELAY
127.0.0.1                               RELAY
192.168.1                               RELAY
mydomain.com                            RELAY
```

Note that it is incorrect to write lines like the following in the file above:

```
192.168.1.0             RELAY           (incorrect)
*.mydomain.com          RELAY           (incorrect)
```

After getting the prompt back, update the `/etc/mail/access.db` file by:

```
[root@mainserver /]# makemap hash /etc/mail/access.db < /etc/mail/access
```

Now restart the Sendmail server.

If you ever need to allow other domains to send mail to other domains using your mail server then you would configure `/etc/mail/relay-domains` like this:

```
[root@mainserver /]# vi /etc/mail/relay-domains

192.168.1
mydomain.com
```

Note that it is incorrect to write lines like the following in the file above:

```
192.168.1.0             RELAY           (incorrect)
192.168.1               RELAY           (incorrect)
*.mydomain.com          RELAY           (incorrect)
*.mydomain.com                          (incorrect)
```

After getting the prompt back, restart the Sendmail server.

Chapter 11: DHCP

Quite some useful service. Hands out IP, Gateway, DNS server,etc, to the needy ones. I mean the IP hunger of computers are satisfied by this service.

Packages

`dhcp-devel-a.b`
`dhcp-a.b`

Scenario

You have a simple network with a mainserver (192.168.1.254) trying to cope up with fulfilling all needs of the users. But now users want to have their IP assigned to them automatically whenever they boot and whatever operating system they boot up on their machines.

Server side configuration:

Normally you have a sample dhcpd file lying in `/usr/share/doc/dhcp-3.0pl1` directory of your machine as `dhcpd.conf.sample`. Copy this file in your `/etc` directory and rename it as `dhcpd.conf` . Many people do a mistake here, they rename it to ***`dhcp.conf`*** and expect `dhcpd` service to work. It won't. Once done, edit it as following:

```
[root@mainserver /]# vi /etc/dhcpd.conf

ddns-update-style interim;
ignore client-updates;
subnet 192.168.1.0 netmask 255.255.255.0 {

         option routers                     192.168.1.254;
         option subnet-mask                 255.255.255.0;

         option nis-domain         "logindomain";
         option domain-name                 "mydomain.com";
         option domain-name-servers         192.168.1.254;

         option time-offset                 -18000;  # Eastern Standard Time
         option netbios-name-servers        192.168.1.254;   # SAMBA WINS service

         range dynamic-bootp 192.168.1.1 192.168.1.200;
         default-lease-time 21600;
         max-lease-time 43200;
}
```

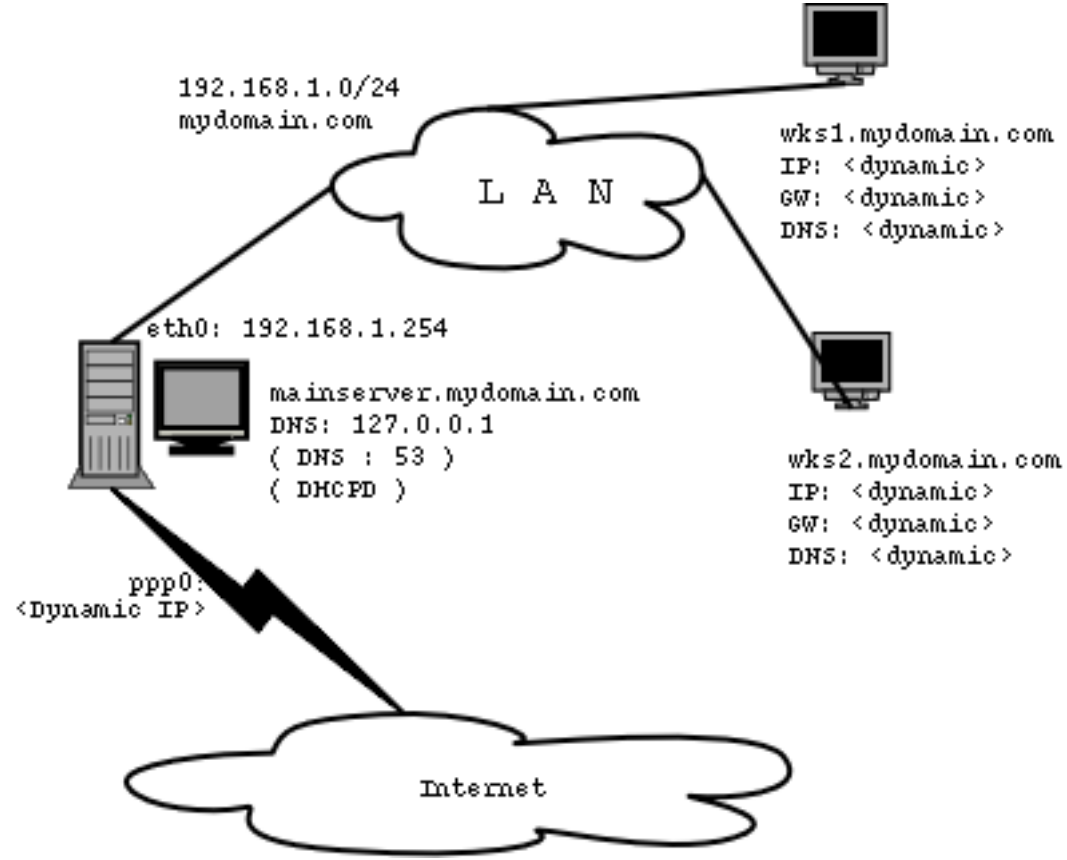

192.168.1.0/24
mydomain.com
L A N
wks1.mydomain.com
IP: <dynamic>
GW: <dynamic>
DNS: <dynamic>
eth0: 192.168.1.254
mainserver.mydomain.com
DNS: 127.0.0.1
(DNS : 53)
(DHCPD)
wks2.mydomain.com
IP: <dynamic>
GW: <dynamic>
DNS: <dynamic>
ppp0:
<Dynamic IP>
Internet

Fig 1

That is it. Quite easy haan ?

Start dhcpd service by :

```
[root@mainserver /]# service dhcpd start
```

Client side configuration:

a) Windows client:
This is no big science. Almost every window user knows that. Go to Network Neighborhood -> Properties -> TCP/IP [*your adapter*] -> Properties -> Obtain IP address automatically . Also select obtain DNS automatically. That is it. Depending on the version (and mood) of Windows, you may need to restart your client.

b) Linux client:

Edit `/etc/sysconfig/network-scripts/ifcfg-eth0` file and set it up as following:

```
[root@wks1 /]# vi /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
DHCP_HOSTNAME=wks1
```

Please note that it is *incorrect* to write the FQDN here like this:
`DHCP_HOSTNAME=wks1.mydomain.com`

Once done , restart the network service of the client. Your client should pick up IP from DHCP server.

```
[root@wks1 /]# service network restart
```

Log files

`/var/log/messages`

Chapter 12: FTP

File Transfer Protocol. FTP is the easiest way to send and receive files over the network. Think of situations when you cannot send large files as attachments to some one. Even if you can, you never know that when that file / message will be sent and whether the other person's mail box was having enough room to accommodate your files attached with the mail message. Amongst many FTP servers, I will be discussing only the *VSFTPD* server in this section.

Packages:

`vsftpd-a.b`

Something about FTP ports, Active mode and Passive mode FTP:

Summary:
In *Active mode FTP*, the FTP server connects to IP and port announced by client, using server's FTP data port (20). In *Passive mode* FTP, the server assigns / opens port on the server, tell the client about it, and then let the client connect to the new port.

More detail:
The FTP protocol first opens up a single connection that is called the FTP control session. When we issue commands through this session, other ports are opened to carry the rest of the data related to that specific command. These connections can be done in two ways, either actively or passively.

When a connection is done actively, the FTP client sends the server a port and IP address to connect to. After this, the FTP client opens up the port and the server connects to that specified port from its own port 20 (known as FTP-Data) and sends the data over it.

Passive FTP works the opposite way. The FTP client tells the server that it wants some specific data, upon which the server replies with an IP address to connect to and at what port. The client will, upon receipt of this data, connect to that specific port, from its own port 20(the FTP-data port), and get the data in question.

Scenario:

We have a small LAN, which has the server running FTP along DNS to provide file transfer facility to the users.

Server side configuration:

VSFTPD has the config file as `/etc/vsftpd/vsftpd.conf` , in RedHat 9 . In RedHat 8, it is `/etc/vsftpd.conf.` This file is quite straight-forward to configure. The default configuration will allow any user to connect to this ftp server as *anonymous*.

If you want to restrict anonymous logins then set the following directives in this file as below:

```
[root@mainserver /]#  vi /etc/vsftpd/vsftpd.conf
anonymous_enable=NO
```

To enable local users to login to this FTP server, use:

```
local_enable=YES
```

Set up the welcome banner:

```
ftpd_banner=Welcome to  FTP service at mydomain.com.
```

This will make sure that users cannot change directory outside their home directories:

```
chroot_local_user=YES
```

Now start the vsftpd server by:

```
[root@mainserver /]#  service vsftpd start
```
(RedHat9)

Client side configuration:

Nothing special is required. You can use any FTP client like: *cuteFTP* f/windows, *gFTP* f/Linux, or even *ftp* command from command line.

Below is my command line ftp session.

Here is how I connected to my server:

```
[root@mainserver /]# ftp mainserver.mydomain.com
```

```
Connected to mainserver.mydomain.com.
220 Welcome to FTP service at mydomain.com
530 Please login with USER and PASS.
530 Please login with USER and PASS.
KERBEROS_V4 rejected as an authentication type
Name (mainserver.mydomain.com:root): kamran
331 Please specify the password.
Password:
230 Login successful. Have fun.
Remote system type is UNIX.
Using binary mode to transfer files.
```

Here is what I got when I checked my directory listing and also tried to change directories. Notice that ***chroot*** environment prevents me from going outside my home directory:

```
ftp>ls
227 Entering Passive Mode (127,0,0,1,52,103)
150 Here comes the directory listing.
lrwxrwxrwx    1 500      500            12 Jun 05 04:22 kamran -> /home/kamran
-rw-rw-r--    1 500      500          4847 Jun 05 04:25 nbbn.sxw
226 Directory send OK.

ftp> cd ..
250 Directory successfully changed.

ftp> ls
227 Entering Passive Mode (127,0,0,1,192,249)
150 Here comes the directory listing.
lrwxrwxrwx    1 500      500            12 Jun 05 04:22 kamran -> /home/kamran
-rw-rw-r--    1 500      500          4847 Jun 05 04:25 nbbn.sxw
226 Directory send OK.

ftp> cd /
250 Directory successfully changed.

ftp> ls
227 Entering Passive Mode (127,0,0,1,53,220)
150 Here comes the directory listing.
lrwxrwxrwx    1 500      500            12 Jun 05 04:22 kamran -> /home/kamran
-rw-rw-r--    1 500      500          4847 Jun 05 04:25 nbbn.sxw
226 Directory send OK.
```

Now I try to upload a file (`install.log`) from my local directory (`/root`).

```
ftp> !ls
anaconda-ks.cfg  evolution                    install.log
OpenOffice.org1.1.1
Desktop          evolution_29may2004.tar.bz2  install.log.syslog

ftp> put install.log
local: install.log remote: install.log
227 Entering Passive Mode (127,0,0,1,72,196)
150 Ok to send data.
226 File receive OK.
43909 bytes sent in 0.084 seconds (5.1e+02 Kbytes/s)

ftp> ls
227 Entering Passive Mode (127,0,0,1,158,232)
150 Here comes the directory listing.
-rw-r--r--    1 500      500         43909 Jul 08 06:09 install.log
```

```
lrwxrwxrwx    1 500      500            12 Jun 05 04:22 kamran -> /home/kamran
-rw-rw-r--    1 500      500          4847 Jun 05 04:25 nbbn.sxw
226 Directory send OK.
```

Now I try to download a file (`nbbn.sxw`) from my ftp server directory to my local directory (`/root`)

```
ftp> ls
227 Entering Passive Mode (127,0,0,1,141,229)
150 Here comes the directory listing.
-rw-r--r--    1 500      500         43909 Jul 08 06:09 install.log
lrwxrwxrwx    1 500      500            12 Jun 05 04:22 kamran -> /home/kamran
-rw-rw-r--    1 500      500          4847 Jun 05 04:25 nbbn.sxw
226 Directory send OK.

ftp> !ls
anaconda-ks.cfg  evolution                    install.log
OpenOffice.org1.1.1
Desktop          evolution_29may2004.tar.bz2  install.log.syslog

ftp> get nbbn.sxw
local: nbbn.sxw remote: nbbn.sxw
227 Entering Passive Mode (127,0,0,1,94,221)
150 Opening BINARY mode data connection for nbbn.sxw (4847 bytes).
226 File send OK.
4847 bytes received in 0.0097 seconds (4.9e+02 Kbytes/s)

ftp> !ls
anaconda-ks.cfg  evolution                    install.log         nbbn.sxw
Desktop          evolution_29may2004.tar.bz2  install.log.syslog
OpenOffice.org1.1.1
ftp>
```

Log files:

The log file for vsftp is `/var/log/vsftpd.log` . The two operations I just performed are logged in this file as :

```
[root@mainserver /]# tail -f /var/log/vsftpd.log
Thu Jul  8 12:09:28 2004 1 127.0.0.1 43909 /install.log b _ i r kamran ftp 0 * c
Thu Jul  8 12:26:56 2004 1 127.0.0.1 4847 /nbbn.sxw b _ o r kamran ftp 0 * c
```

Chapter 13: FIREWALL

Firewall is a machine / mechanism , capable of protecting your network from any outside (or inside) attacks and abuses. It also help you implement what all services you want to allow on your network. Or in other words, it allows you to decide that which all data packets should go out of your network and which should not, and vice versa.

Packages:

ipchains-a.b	(For kernel 2.2.x)
iptables-a.b	(For kernel 2.4.x)

Scenario:
We have a small network, which has the server connected to the outer world by a modem. This means that our server (`mainserver.mydomain.com` in this case) will be the gateway for our network. In this scenario the server has two interfaces, `eth0` and `ppp0`. We need to establish a firewall on this gateway machine in order to protect our network from outside attacks. Primarily we want the following:

- Nobody should be able to establish a *telnet* or *secure shell* connection to our network from Internet.
- Nobody should be able to initiate (and establish) a connection from Internet with any of your machines other than our web server.
- Also we want to implement **IP Masquerading** or **NAT**, so that our clients can conveniently use their messenger tools like msn messenger and yahoo messenger, etc.

Configuration For kernel 2.2:

Kernel 2.2 has three paths (chains) for data. INPUT, OUTPUT and FORWARD.

List your rules in a script on the gateway machine, i.e. `mainserver.mydomain.com`, as these rules are flushed or erased at each computer reboot. You can save your rules in a script say **`/etc/rc.firewall`** and call it from `/etc/rc.local` script. Or you can even put them directly (and simply) in `/etc/rc.local` .

Now, to block any incoming telnet requests from outside (not from our network), destined for our network, we will use the following command on our `mainserver.mydomain.com`, which is the gateway computer for our network:

```
[root@mainserver /]# ipchains -A input -i ppp0 -p tcp --dport 22 -j DENY
[root@mainserver /]# ipchains -A input -i ppp0 -p tcp --dport 23 -j DENY
```

To allow any web requests to your web server, use:

```
[root@mainserver /]# ipchains -A input -i ppp0 -p tcp --dport 80 -j ACCEPT
```

To allow our network clients to access the Internet for IRC, MSN messenger services, etc, follow:

1)Add the following line to our firewall script, which says that rewrite all packet headers to that IP which is assigned to us from our ISP, originating from our lan and are destined anywhere other than our lan:

```
[root@mainserver /]# ipchains -A forward -i eth0 -o ppp0 -p tcp -j MASQ
```

2)Enable IP Forwrding in the file `/proc/sys/net/ipv4/ip_forward` by :

```
[root@mainserver /]# echo 1 >  /proc/sys/net/ipv4/ip_forward
```

Configuration For kernel 2.4:

Kernel 2.4 have five paths (chains) for data. INPUT, OUTPUT, FORWARD, PREROUTING and POSTROUTING. All commands are case sensitive.

List your rules in a script on the gateway machine, i.e. `mainserver.mydomain.com`, as these rules are flushed or erased at each computer reboot. You can save your rules in a script say **`/etc/rc.firewall`** and call it from `/etc/rc.local` script. Or you can even put them directly (and simply) in `/etc/rc.local` .

To block any incoming telnet requests from Internet (not from our network), destined for our network, we will use:

```
[root@mainserver /]# iptables -A INPUT -i ppp0 -p tcp --dport 23 -j DROP
```

also

```
[root@mainserver /]# iptables -t nat -A PREROUTING -i ppp0 -p tcp --dport 23  -j
DROP
```

To allow any web requests to and from your web server, use the following two commands:

```
[root@mainserver /]# iptables -A INPUT -i ppp0  -p tcp --dport 80 -j ACCEPT
[root@mainserver /]# iptables -A OUTPUT -o ppp0 -p tcp -d 0/0 --dport 80 -j ACCEPT
```

The following allows connections (initiated by outsider) seeking port 80 from outside on your computer but not any other.

```
[root@mainserver /]# iptables -A INPUT -i ppp0 -p tcp -syn --dport !80 -j DROP
```

But to deny any connections from any of the outsider to any of your machine on port 80, other than your server, use:

```
[root@mainserver /]# iptables -A FORWARD -i ppp0 -o eth0 --dport 80 -j DROP
```

To allow our network clients to access the Internet for IRC, MSN messenger services, etc, follow :

1)Add the following line to our firewall script, which says that rewrite all packet headers to that IP which is assigned to us from our ISP, originating from our lan and are destined anywhere other than our lan:

```
[root@mainserver /]# iptables -t nat -A POSTROUTING -o ppp0 -p tcp  -j MASQUERADE
```

2)Enable IP Forwarding in the file **`/proc/sys/net/ipv4/ip_forward`** by :

```
[root@mainserver /]# echo 1 >  /proc/sys/net/ipv4/ip_forward
```

Transparent proxy:

If you want to run SQUID as a transparent proxy server and your SQUID machine is the gateway machine in your network, then you need to pickup any request for port 80 from your network before they get to the Internet and redirect to the local proxy server, SQUID, running on port 3128.

In order for transparent proxy to work, you need to have *transparent proxy support* compiled into your kernel.

Use the following command on kernel 2.2 machine:

```
[root@mainserver /]# ipchains -A input -p tcp -i eth0 --dport 80 -j REDIRECT 3128
```

Use the following command on kernel 2.4 machine:

```
[root@mainserver /]# iptables -t nat -A PREROUTING -i eth0 -p tcp
          ➥      --dport 80       -j REDIRECT --to-port 3128
```

For more information and configuration of SQUID as transparent proxy server, refer to chapter: *"SQUID"*.

Now some explanation of DNAT , SNAT and MASQUERADE targets (including diagrams)

MASQUERADE:

If you have a network of computers and your dialup machine is linux based and each time you connect to your ISP , you get a different IP address (dynamic public IP), then you need masquerading to allow your client computers web traffic to go out on the internet. It should be noted here that even you don't have SQUID or you don't want to run SQUID and just want to enable windows internet connection sharing type of thing to be implemented, then all you need is the following two commands on your linux dialup machine:

```
[root@mainserver /]# echo 1 > /proc/sys/net/ipv4/ip_forward
[root@mainserver /]# iptables -t nat -A POSTROUTING -o ppp0 -p tcp -j MASQUERADE
```

Note that *MASQUERADE*ing also works with dynamic private IPs.

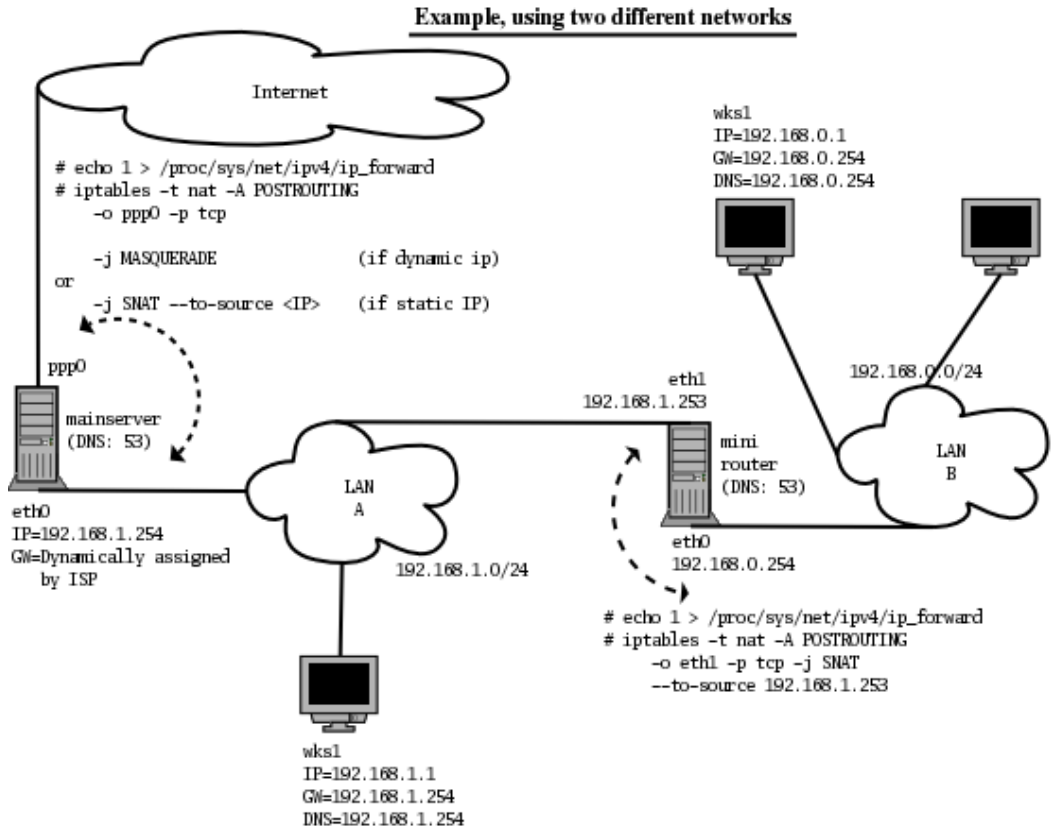

Example, using two different networks
Internet
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING
-o ppp0 -p tcp
-j MASQUERADE (if dynamic ip)
or
-j SNAT --to-source <IP> (if static IP)
ppp0
mainserver
(DNS: 53)
eth0
IP=192.168.1.254
GW=Dynamically assigned
by ISP
LAN
A
192.168.1.0/24
wks1
IP=192.168.1.1
GW=192.168.1.254
DNS=192.168.1.254
eth1
192.168.1.253
mini
router
(DNS: 53)
eth0
192.168.0.254
echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -t nat -A POSTROUTING
-o eth1 -p tcp -j SNAT
--to-source 192.168.1.253
wks1
IP=192.168.0.1
GW=192.168.0.254
DNS=192.168.0.254
192.168.0.0/24
LAN
B

Fig 1

SNAT:

SNAT is just like MASQUERADE but it is twice fast as compared to MASQUERADE. It is used where we have a the ***same IP*** each time you connect to your ISP (static public IP). *See Fig 1*. In this case you don't want time to be wasted by checking each time the IP of the outgoing interface, because you already know it. So just mention that in the iptables command. Assume you get 203.135.10.15 as IP of your modem interface (ppp0), each time when you connect to your ISP, then the "*internet connection sharing*" commands will be:

```
[root@mainserver /]# echo 1 > /proc/sys/net/ipv4/ip_forward

[root@mainserver /]# iptables -t nat -A POSTROUTING -o ppp0 -p tcp
                -j SNAT –-to-source 203.135.10.15
```

As told earlier, SNAT is twice fast / efficient than MASQUERADE , but works with static IPs only (private and public both).

DNAT:

In a small network, normally people use SQUID on the same machine which is their dialup machine. Or in other words, they normally run SQUID on the gateway machine. But what if they do not want to run SQUID on the gateway machine ? *See Figure 2.* Such a case is when you have SQUID on a non-gateway machine on the network with only one network interface card (eth0, with IP:192.168.1.252, on port 3128) . In such case any web traffic coming from the LAN needs to be directed towards this machine, if transparent proxy setup is desired. This can be achieved by the following commands on the (linux) gateway machine:

```
[root@mainserver /]# iptables -t nat -A PREROUTING -i eth0  -s ! 192.168.1.252
        ➥       -p tcp --dport 80 -j DNAT –-to-destination 192.168.1.252:3128
```

DNAT can also be used where you have a setup such as you have your web server on the LAN with only one interface (eth0, with IP:192.168.1.251) . *See Figure 3.* You have a permanent (public) IP assigned to your dialup linux machine (202.135.10.18) , each time you connect to your ISP. You have your web server registered on the Internet against this IP, BUT you don't have any web server running on this IP !!!!!! . Why ? Because as I already told you you have your webs server in your LAN on a private IP 192.168.1.251 . Now you want to redirect any web requests for your web server coming from internet to your server lying inside your LAN. You can do it by:

```
[root@mainserver /]# iptables -t nat -A PREROUTING -i ppp0 -d 202.135.10.18
        ➥       -p tcp --dport 80 -j DNAT –-to-destination 192.168.1.251:80
```

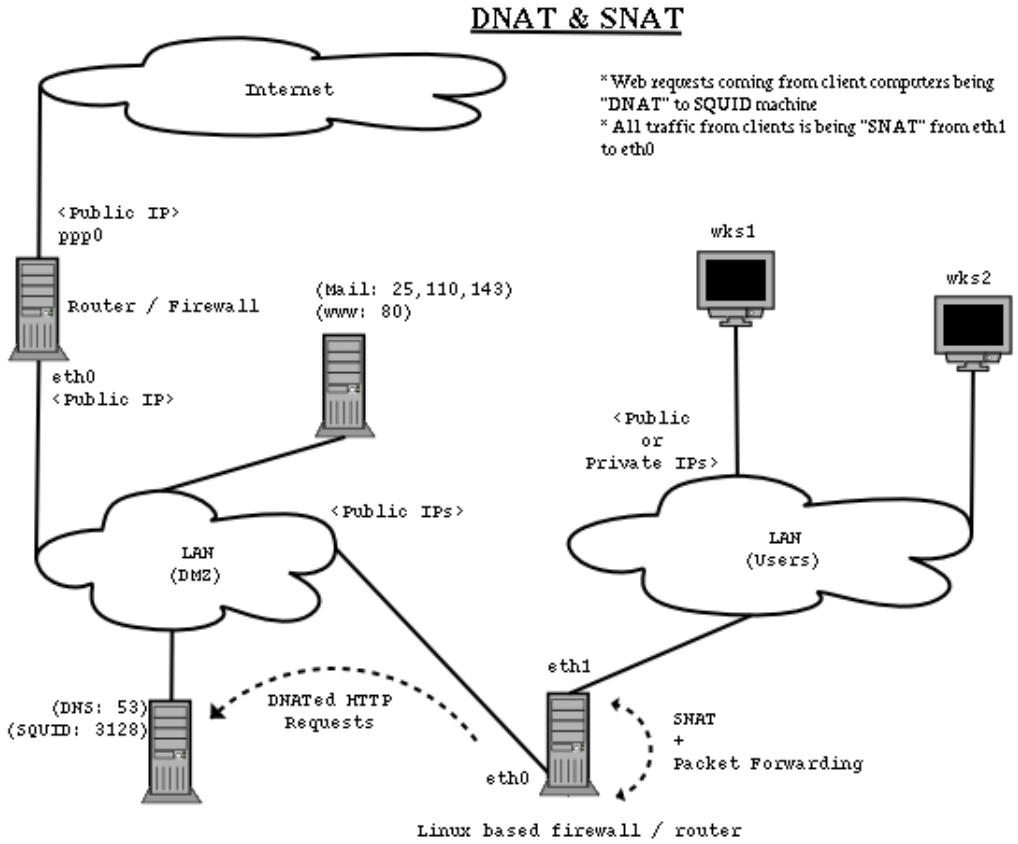

DNAT & SNAT
Internet
* Web requests coming from client computers being "DNAT" to SQUID machine
* All traffic from clients is being "SNAT" from eth1 to eth0
<Public IP>
ppp0
Router / Firewall
eth0
<Public IP>
(Mail: 25,110,143)
(www: 80)
wks1
wks2
<Public or Private IPs>
<Public IPs>
LAN (DMZ)
LAN (Users)
eth1
(DNS: 53)
(SQUID: 3128)
DNATed HTTP Requests
SNAT + Packet Forwarding
eth0
Linux based firewall / router

Fig 2

Another example of DNAT & SNAT

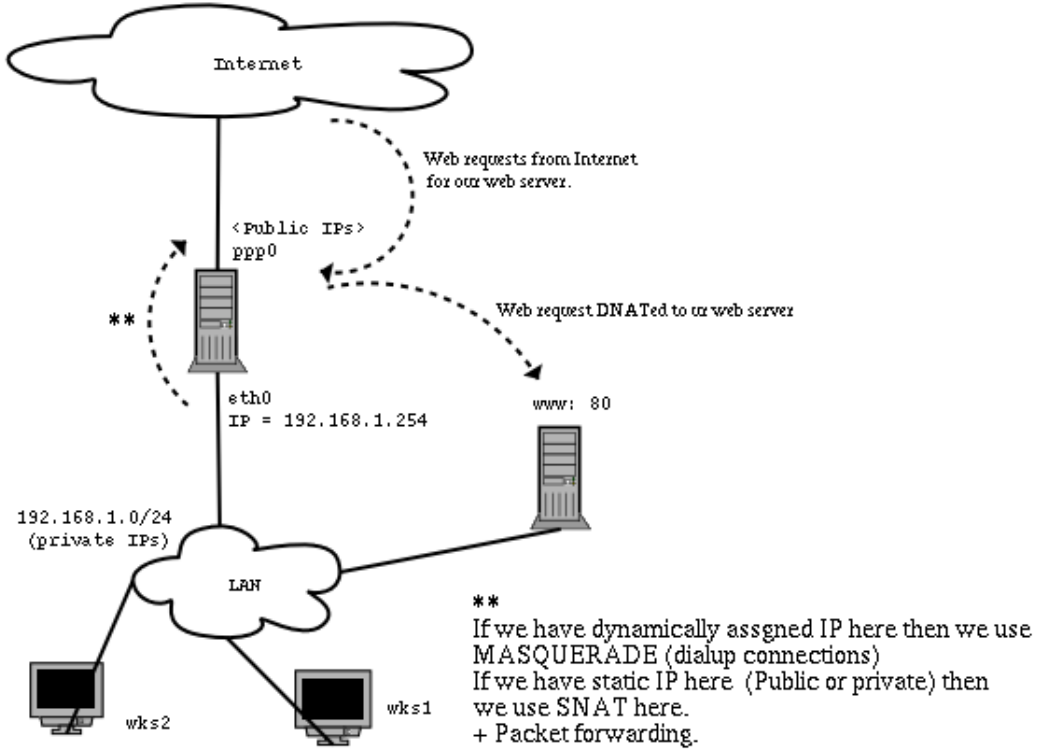


Fig 3

Sample firewall configuration / script:

A common request. Specially people ask how to block all those ICMP requests coming in our network from outside (and/or inside) ? Plus administrators are too irritated to see people trying to connect to their servers from outside using *telnet* or *ssh*, etc. Below is a sample script which you can use for both in networked scenario and standalone scenario. Save this as a script, preferably as `/etc/rc.firewall`, make it executable and call it from `/etc/rc.local`

```
[root@mainserver /]# vi /etc/rc.firewall

#!/bin/bash

#following three lines provides Masquerading for our network.
# You don't need these on a stand alone computer.

echo Enabling Masquerading ...

echo 1 > /proc/sys/net/ipv4/ip_forward

iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -d ! 192.168.1.0/24 -o ppp0
         ➥       -j MASQUERADE

# following two lines redirects www (port 80) requests to local squid
# service. Do not use if you don't want to enable transparent        proxying or
if
# you don't know about transparent proxying.
# Needs some configuration in squid.conf file as well.

echo Enabling redirection for port 80 assuming Firewall + SQUID on same machine
...
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 3128
iptables -t nat -A PREROUTING -i eth0 -p udp --dport 80 -j REDIRECT --to-port 3128


#iptables -t nat -A PREROUTING -i eth0 -s ! 192.168.1.251 -p tcp --dport 80
         ➥       -j DNAT --to-destination 192.168.1.251:3128
#iptables -t nat -A PREROUTING -i eth0 -s ! 192.168.1.251 -p udp --dport 80
         ➥       -j DNAT --to-destination 192.168.1.251:3128

# Drop  connections from internet for ports 0-1023, which are targeting this
machine
iptables -A INPUT -i ppp0 -p tcp --destination-port 0:1023 -j DROP
iptables -A INPUT -i ppp0 -p udp --destination-port 0:1023 -j DROP


# lets drop traffic other than 20,21,80,1863 MSN ,5050 yahoo

iptables -A FORWARD -i eth0 -o ppp0 -p tcp -m multiport
         ➥         --dport 20,21,110,1863,443,11999,5050,2002,2004 -j ACCEPT
iptables -A FORWARD -i eth0 -o ppp0 -p udp -m multiport
         ➥         --dport 20,21,110,1863,443,11999,5050,2002,2004 -j ACCEPT


# the line right next is for FTP ESTABLISHED sessions
iptables -A FORWARD  -p tcp -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -p udp -m state --state ESTABLISHED,RELATED -j ACCEPT

# 11999 is yahoo games
# MSN needs both 1863 and 443, Yahoo needs 5050 (and probably 443 as well)

iptables -A FORWARD -i eth0 -o ppp0 -p tcp -m multiport
```

```
        ➥       --dport 20,21,110,1863,443,11999,5050,2002,2004 -j ACCEPT

#iptables -A FORWARD -i eth0 -o ppp0 -p tcp -d 0/0 --dport 81:65535 -j DROP
iptables -A FORWARD -p tcp -m multiport --destination-port 135,139,445 -j DROP
iptables -A FORWARD -p udp -m multiport --destination-port 135,139,445 -j DROP


# Also drop connections to this machine from internet for the following multiple
ports
# my SQUID , postgresql database server and X11 interface

iptables -A INPUT -i ppp0 -p tcp -m multiport --destination-port 3128,5432,6000 -j
DROP
iptables -A INPUT -i ppp0 -p udp -m multiport --destination-port 3128,5432,6000 -j
DROP

# And the following DROPS icmp packets coming from internet

iptables -A INPUT -i ppp0 -p icmp -j DROP
```

Assign execute permission to the above script by:

```
[root@mainserver /]# chmod u+x /etc/rc.firewall
```

Edit `/etc/rc.local`, and add a line to call the above script:

```
[root@mainserver /]# vi /etc/rc.local

#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.
touch /var/lock/subsys/local
/etc/rc.firewall
```

You can download the above firewall script from my web site, `http://www.geocities.com/mk_azeem/samplefirewall.txt`

Log files:

There are no log files as such, but still you can get most out of `/var/log/messages` and all log files related to particular services, to check whether desired connections are being made or not.

Documentation and Help:

- Chapter 9 and 11 in the book **Linux Network Administrator's Guide** (Second Edition) available on the guides section on `www.tldp.org`
- `www.flounder.net/ipchains/ipchains-howto.html`
- Iptables tutorial 1.1.19 by Oskar Andreason
- Iptables Basics NHF by prince_kenshi

Testing

You can test your Firewall by directly running the related applications like your web browser or try connecting from outside to your computer using telnet, etc.

In order to check the current state of your rules you can use:

```
[root@mainserver /]# ipchains -L          (For kernel 2.2)
```

or

```
[root@mainserver /]# iptables -L          (For kernel 2.4)
[root@mainserver /]# iptables -t nat -L   (For Kernel 2.4)
```

At any time you can Flush your current rule set by:

```
[root@mainserver /]# ipchains -F          (For Kernel 2.2 )
```

or

```
[root@mainserver /]# iptables -F          (For Kernel 2.4 )
[root@mainserver /]# iptables -t nat -F   (For Kernel 2.4 )
```

Chapter 14: KERNEL MODIFICATION AND COMPILATION

There are times when you need to recompile your kernel with more (or less) features.

Packages:

`kernel-source-a.b`
`kernel-utils-a.b`
`glibc-kernheaders-a.b`

Scenario:

Assuming that you want to add *NTFS* read feature to your kernel.

Procedure:

Check your kernel version by:

```
[root@mainserver /]# uname -r
```

Then go to the kernel source directory `/usr/src` and further into the directory which has the same name / number as your kernel version number, returned to you by the *uname* command above. On a RedHat 8 system it is **`2.4.18-14`**

```
[root@mainserver /]# cd /usr/src/linux-2.4.18-14
```

Now you can run the kernel configuration by one of three methods:

1) By `make config` from a command prompt, which will ask you a lot of Yes and No questions. This takes up a lot of time and is rather now obsolete method.

```
[root@mainserver linux-2.4.18-14]# make config
```

2) By `make menuconfig` from the command prompt, which will show a text based menu system and you can select different options and save changes.

```
[root@mainserver linux-2.4.18-14]# make menuconfig
```

3) And by `make xconfig` from a command terminal running in Xwindows. Very user friendly.

```
[root@mainserver linux-2.4.18-14]# make xconfig
```

So for now we will use `make menuconfig`. Once you are done with your settings of NTFS read feature, you can save the kernel configuration while exiting out of the program. After save and exit, you have to run the following commands in the order in which they appear:

```
[root@mainserver linux-2.4.18-14]# make dep
[root@mainserver linux-2.4.18-14]# make clean
[root@mainserver linux-2.4.18-14]# make bzImage
[root@mainserver linux-2.4.18-14]# make modules
```

It may take from few minutes to few hours, depending on the power of your particular machine.

After all of above steps are completed and the prompt returns, you have to copy newly created **`bzImage`** to your `/boot` directory:

```
[root@mainserver linux-2.4.18-14]# cp arch/i386/boot/bzImage
        ➥       /boot/vmlinuz-ntfs
```

Then issue the commands :

```
[root@mainserver linux-2.4.18-14]#  make modules_install
[root@mainserver linux-2.4.18-14]#  cp System.map /boot/System.map-ntfs
[root@mainserver linux-2.4.18-14]#  cp .config /boot/config-ntfs
```

The last command is not really necessary. But is is useful to keep the kernel configuration at a separate place as a backup.

Then create a boot driver RAM disk file (optional):

```
[root@mainserver linux-2.4.18-14]# mkinitrd /boot/initrd-ntfs.img 2.4.18-14
```

Then you need to update your boot loader configuration as well.

LILO:

In case you are running LILO, update the `/etc/lilo.conf` and add a section for your new kernel:

```
[root@mainserver /]# vi /etc/lilo.conf

prompt
timeout=50
default=linux
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
message=/boot/message
linear

image=/boot/vmlinuz-2.4.18-14
        label=linux
        initrd=/boot/initrd-2.4.18-14.img
        read-only
        append="root=LABEL=/"

image=/boot/vmlinuz-ntfs
        label=linux with NTFS support
        initrd=/boot/initrd-ntfs.img
        read-only
        append="root=LABEL=/"

other=/dev/hda2
        optional
        label=DOS
```

After saving the LILO configuration, run LILO boot map installer:

```
[root@mainserver /]# /sbin/lilo -v
```

Now you are done, reboot your machine and there will be a new entry in the boot menu **Linux with NTFS support**, select that and press enter to boot from this kernel.

GRUB:

In case you have GRUB, then you need to update the `/boot/grub/grub.conf` file and add a section relating to your new kernel image.

```
[root@mainserver /]# vi /boot/grub/grub.conf

# NOTICE:  You have a /boot partition.  This means that
#          all kernel paths are relative to /boot/
default=0
timeout=30
splashimage=(hd0,0)/grub/splash.xpm.gz

title Red Hat Linux (2.4.18-14)
        root (hd0,0)
        kernel /vmlinuz-2.4.18-14 ro root=/dev/hda3
        initrd /initrd-2.4.18-14.img

title Red Hat Linux with ntfs support (2.4.18-14)
        root (hd0,0)
```

```
        kernel /vmlinuz-ntfs ro root=/dev/hda3
        initrd /initrd-ntfs.img
```

Now reboot your machine. There is no need to rerun grub program itself to update the boot record with new entries. GRUB reads it's config file at each startup and acts accordingly.

For emergencies, you should also make a boot disk before and after kernel compilation:

```
[root@mainserver /]# mkbootdisk --device /dev/fd0 --verbose 2.4.18-14
```

Notes:

If, after kernel compilation, you forgot to run `/sbin/lilo` and if you did not have an entry for an older kernel in `lilo.conf`, you have a problem. In many cases, you can boot your Red Hat Linux system from the Red Hat Linux boot disk with your root file system mounted and ready to go. Enter the following command at the boot disk's boot: prompt:

```
linux single root=/dev/hdXX initrd=
```

In the above command, replace the `XX` in `/dev/hdXX` with the appropriate letter and number for your root partition. This command, first, starts the boot process in single-user mode, with the root partition set to your root partition. The empty initrd specification bypasses the installation-related image on the boot disk, which will cause you to enter single-user mode immediately. But if your system is SCSI-based, you will not be able to do this. In that case, you will have to access rescue mode using the **`linux rescue`** command by using the installation CD # 1, explained in the next chapter.

Chapter 15: Single user and Rescue mode

Rescue mode:

Rescue mode provides the ability to boot a small Linux environment entirely from a diskette, CD-ROM, or using some other method. Normally, you will need to get into rescue mode for one of two reasons:

- You are unable to boot Linux. (Happens normally when you install some operating system over / after installing linux, like installation of windows over / after installing Linux)
- You are having hardware or software problems, and you want to get a few important files off your system's hard drive before starting “OPERATION UNDER KNIFE”.

To boot your system in rescue mode, boot from a Red Hat Linux boot disk or Red Hat Linux CD-ROM #1, and enter the following command at the installation boot prompt:

```
boot: linux rescue
```

After booting off a boot disk or Red Hat Linux CD-ROM #1 and providing a valid rescue image, you will see the following message:

```
The rescue environment will now attempt to find your Red Hat
Linux installation and mount it under the directory
/mnt/sysimage.  You can then make any changes required to your
system.  If you want to proceed with this step choose
'Continue'.
If for some reason this process fails you can choose 'Skip'
and this step will be skipped and you will go directly to a
command shell.
```

If you select *Continue*, it will attempt to mount your file system under the directory **`/mnt/sysimage`**. If it fails to mount a partition, it will notify you. If you select *Skip*, your file system will not be mounted. Choose *Skip* if you think your file system is corrupted. Once you have your system in rescue mode, a `bash#` prompt appears. If you selected *Continue* to mount your partitions automatically and they were mounted successfully, you are in single-user mode. From here you can do some of your normal tasks like copying files, changing root password, changing network settings, etc.

If your file system was mounted properly during rescue boot, and you want to make your system the root partition, use the command: `chroot /mnt/sysimage`. This is useful if you need to run commands such as *rpm* that require your root partition to be mounted as /. To exit the `chroot` environment, type `exit`, and you will return to the *bash#* prompt.

```
# chroot /mnt/sysimage
```

However, if your partitions could not be mounted automatically. You can first check the list of partitions and try to identify them by:

```
# fdisk -l
```

Then you can mount them by first creating some temporary mount point like `/tempusr` and mount your partition `/dev/hda5` on that mount point (which is your `/usr` infact).

```
# mkdir /tempusr
# mount -t ext2 /dev/hda5 /tempusr
```

Your partition can be of any type, ext2 or ext3, etc.

Now if your MBR was over written by some operating system, which you installed after installing Linux, and you were using lilo as boot loader then you can restore / recreate your boat loader by using **`lilo -r`** as :

```
# lilo -r /mnt/sysimage    (if chroot command is not already issued)
```

or simply

```
# lilo -r /      (if chroot command is already given and you are in chroot
                  environment)


Added DOS
Added Linux-ntfs
Added Linux-2.4.18-14
```

And if you were using GRUB then you would restore your GRUB boot loader by:

```
# grub-install /dev/hda
```

TO synchronize, issue **`sync`** command by:

```
# sync
```

And then **exit** twice to reboot the system.

```
# exit
# exit
```

The system will reboot at this point. If using GRUB as boot loader, and you want to recreate your boot loader then you have two options (in detail) to follow:

Method1: Installing GRUB using floppy:

You need a GRUB boot floppy to handle the situation. Create one and reboot your computer with it. Once started, GRUB will show the command-line interface. First, set the GRUB's "root device" to the boot directory, like this:

```
grub> root (hd0,0)
```

If you are not sure which partition actually holds these files, use the command `find' like this:

```
grub> find /boot/grub/stage1
```

This will search for the file name "/boot/grub/stage1" and show the devices which contain the file. Once you've set the root device correctly, run the command `setup'

```
grub> setup (hd0)
```

This command will install GRUB on the MBR in the first drive. If you want to install GRUB into the "boot sector" of a partition instead of the MBR, specify a partition into which you want to install GRUB:

```
grub> setup (hd0,0)
```

If you install GRUB into a partition or a drive other than the first one, you must chain-load GRUB from another boot loader. Refer to the manual for the respective boot loader to know how to chain-load GRUB. Now you can boot GRUB without a GRUB floppy.

Method2: Using grub-install to install GRUB:

The usage is basically very easy. You only need to specify one argument to the program, namely, where to install GRUB. The argument can be either of a device file or a GRUB's drive/partition. So, the following will install GRUB into the MBR of the first IDE disk under Linux:

```
# grub-install /dev/hda
```

If it is the first BIOS drive, this is the same as well:

```
# grub-install '(hd0)'
```
(similar to previous command)

But all the above examples assume that you use GRUB images under the root directory. If you want GRUB to use images under a directory other than the root directory, you need to specify the option `` `--root-directory' ``. The typical usage is that you create a GRUB boot floppy with a file system. Here is an example:

```
# mke2fs /dev/fd0
# mount -t ext2 /dev/fd0 /mnt/floppy
# grub-install –root-directory=/mnt/floppy '(fd0)'
# umount /mnt/floppy
```

Another example is in case that you have a separate boot partition which is mounted at `` `/boot' ``. Since GRUB is a boot loader, it doesn't know anything about mount points at all. Thus, you need to run `` `grub-install' `` like this:

```
# grub-install --root-directory=/boot /dev/hda
```

By the way, as noted above, it is quite difficult to guess BIOS drives correctly under a UNIX-like OS. Thus, `` `grub-install' `` will prompt you to check if it could really guess the correct mappings, after the installation. Please be careful enough. If the output is wrong, your computer may get stuck at boot-up.

Note that `grub-install' is actually just a shell script and the real task is done by the grub shell `grub'. Therefore, you may run `grub' directly to install GRUB, without using `grub-install'. Don't do that, however, unless you are very familiar with the internals of GRUB. Installing a boot loader on a running OS may create unwanted problems.

Single-User mode:

Normally used when you forget your root password and you have no other way to get in and change it. If you are using GRUB, use the following steps to boot into single-user mode:

- If you have a GRUB password configured, type `p` and enter the password.
- Select Red Hat Linux with the version of the kernel that you wish to boot and type `e` for edit. You will be presented with a list of items in the configuration file for the title you just selected.
- Select the line that starts with `kernel` and type `e` to edit the line.
- Go to the end of the line and type `single` or `1` as a separate word (press the [Space bar] and then type **`single`** or **`1`**). Press [Enter] to exit edit mode.
- Back at the GRUB screen, type `b` to boot into single user mode.

If you are using LILO, just specify `linux single` or `linux 1` or `linux emergency` at the LILO boot prompt (if you are using the graphical LILO, you must press [Ctrl]-[x] to exit the graphical screen and go to the boot: prompt), as :

```
boot: linux single
boot: linux emergency
```

In single-user mode, you computer boots to ***runlevel 1***. Your local file systems will be mounted, but your network will not be activated. You will have a usable system maintenance shell.

In emergency mode, you are booted into the most minimal environment possible. The root file system will be mounted read-only and almost nothing will be set up. The main advantage of *emergency mode* over *single user mode* is that your init files are not loaded. If init is corrupted or not working, you can still mount file systems to recover data that could be lost during a re-installation.

FSCK:

Also known as as *"the 'F' word"* by many, specially when you are not careful. *File System ChecK* (*fsck*) is a utility like *scandisk* or *chkdsk* utility in DOS. This can be used to check (and repair) errors on partition level (*or to create them if you wish!*). But this is to be kept in mind that "***YOU SHOULD NEVER RUN FSCK ON A MOUNTED FILESYSTEM***". If you do that, it may (and probably will) destroy your data. If you ever want to run fsck on a file system, first unmount it and then run fsck on it. In case of /boot or /usr or other mount points this may be done by un-mounting them. But you will be unable to un-mount root (/) mount point to run fsck on it. For that it is better to restart in single user mode. In single user mode the root (/) file system is mounted read-only. By read only I do not mean that no writing will be allowed on the root partition. Rather the point here is that linux uses *Asynchronous File System* by default. So in a normal mounted file system, the state of a file might be different in memory as compared to on disk at any given time. Assuming you changed a file and saved it. For you you have saved it but linux has lied to you and is in fact waiting for a proper time to flush the (buffer) changes to the disk. This is change of state of a file in memory and on disk. So if you run fsck on such a file system, the files may get repaired / deleted for inconsistencies, still having their related data in memory (dirty buffers). And you get a (very ugly) and unwanted (corrupted) file system rather repaired one. So what read-only mounted file system means is that writing is allowed as normal, but there will be no data in the buffers. So fsck does what it is supposed to do, that is repair a file system.

If you want you can remount a file system read-only at any time by:

```
[root@mainserver /]# mount -o remount,ro /
```

Normally you run `fsck` like:

```
[root@mainserver /]# fsck /dev/hda1
```

Here is a sample on how I run `fsck` on my `/boot` partition which is `/dev/hda1` on my system.

```
[root@mainserver /]# fsck /dev/hda1

fsck 1.32 (09-Nov-2002)
e2fsck 1.32 (09-Nov-2002)
/dev/hda1 is mounted.

WARNING!!!  Running e2fsck on a mounted filesystem may cause
SEVERE filesystem damage.

Do you really want to continue (y/n)? no
```

```
check aborted.


[root@mainserver /]# umount /boot


[root@mainserver /]# fsck /dev/hda1

fsck 1.32 (09-Nov-2002)
e2fsck 1.32 (09-Nov-2002)
/boot: clean, 41/26104 files, 12623/104391 blocks


[root@mainserver /]# mount /dev/hda1 /boot
```

Second Hard drive attached , abnormal boot up:

It is some times seen that if a second (linux installed) hard drive is attached to an already working linux machine, then linux may boot from the other / new hard drive, which is un-expected and of-course un-wanted. What happens is that you have partition labels, not partition numbers in the `/etc/inittab` file . Once you add another drive which has these partition labels as well, then for some reason, linux picks up that partition as root partition which has label: `"/"` and has a lesser partition number in both drives. e.g. On drive `/dev/hda` , you have root `"/"` partition on `/dev/hda7` . Once you add another hard drive , say `/dev/hdc`, and that had linux already installed on it with a root `"/"` partition as `/dev/hdc2`, then on the boot up time, linux will use this root (`/dev/hdc2`) as root partition and a lot of havoc will be created. You may be asked for root password or to run *fsck* , etc , etc. And you might be wondering that just minutes ago everything was working perfectly fine !!! . So what you can do to avoid such situation is either :

a) boot with a special boot time kernel parameter, this may still not get you desired results
b) edit your boot loader files (and `/etc/fstab` optionally) before attaching the second hard drive

or

a) Boot with boot time kernel parameter:
On LILO (on my machine) , in the above described scenario, pass an extra parameter :

```
boot: linux root=/dev/hda7
```

This will over ride the default root partition label mentioned in the /etc/lilo.conf as:

```
append="root=LABEL=/"
```

On GRUB, edit the kernel line :

```
kernel /boot/vmlinuz ro root=LABEL=/
```

and change it to :

```
kernel /boot/vmlinuz ro root=/dev/hda7
```

b) Solve this problem permanently by editing your boot loader files as per above discussion. Also since your `/etc/fstab` file is still having labels for your non-root partitions as well, this might still cause problems during booting even if you override root partition location by passing boot time parameter as discussed above. So edit your
`/etc/fstab` file and change the lines having labels to actual partition numbers as per your system. e.g assume you have part of your `/etc/fstab` as :

```
LABEL=/          /               ext3    defaults        1 1
LABEL=/boot      /boot           ext3    defaults        1 2
```

then change it to:

```
/dev/hda7        /               ext3    defaults        1 1
/dev/hda1        /boot           ext3    defaults        1 2
```

Installing another version of linux on another partition and boot problems related to that:

This is a very important scenario, and believe me many people just get frustrated during this situation. This kind of situation arises when you already have linux, say RedHat9 installed on your machine, along with windows and you happen to install another operating system such as *Fedora core* on another partition. Here is what I have and how I would explain that to you.

Below is the output of `fdisk -l` command on my machine.

```
[root@mainserver /]# fdisk -l

   Device Boot      Start         End      Blocks   Id  System
/dev/hda1               1          13      104391   83  Linux
/dev/hda2              14        2308    18434587+  83  Linux
/dev/hda3   *        2309        2818     4096575    c  W95 FAT32 (LBA)
/dev/hda4            2819        4865    16442527+   f  W95 Ext'd (LBA)
/dev/hda5            2819        3583     6144831   83  Linux
/dev/hda6            3584        3599      128488+  82  Linux swap
/dev/hda7            3600        4865    10169113+  83  Linux
```

The scenario is like this, I had Redhat9 installed on my machine on `/dev/hda1 (/boot)` , `/dev/hda7 (/)` and `/dev/hda6 (swap)` . I also had windows installed on my machine on `/dev/hda3` . I already had `/dev/hda5` reserved for various experimentation. So I used this partition to install *Fedora*. So the Fedora was installed on `/dev/hda5 (/)` and `/dev/hda6 (swap)` , with no separate `/boot` partition. After installation, Fedora over-wrote my MBR with new GRUB and at the next boot up I had only *Fedora core 2* and *Others* in the GRUB menu. On selecting *Others* , the computer started booting Windows. This was very undesirable, as I was unable to see (or run) my Redhat9 installation in the menu. When booted up in Fedora, I examined the `/etc/grub.conf` file. It looked like this:

```
[root@mainserver /]# /etc/grub.conf

# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You do not have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /, eg.
#          root (hd0,4)
#          kernel /boot/vmlinuz-version ro root=/dev/hda5
#          initrd /boot/initrd-version.img
#boot=/dev/hda
default=0
timeout=10
splashimage=(hd0,4)/boot/grub/splash.xpm.gz
title Fedora Core (2.6.5-1.358)
        root (hd0,4)
        kernel /boot/vmlinuz-2.6.5-1.358 ro root=LABEL=/1 rhgb quiet
        initrd /boot/initrd-2.6.5-1.358.img
title Other
        rootnoverify (hd0,2)
        chainloader +1
```

It should be noted here that I installed Fedora using only two partitions `(/)` and `(swap)` . It doesn't utilize any `/boot` partition . This further means that the kernel image is not on a separate partition , but rather withing the `( / )` partition in the `/boot` directory. The first line below the *title* line in the section above shows: **`root (hd0,4)`** . This line means that kernel boot files , will be on the `/dev/hda5` partition (hard drive 0 , partition number 4, if counted from zero) . Notice that this is

in fact our (/) partition. Since boot files are on the (/) partition, that is why the kernel line below it contains the absolute path:

```
kernel /boot/vmlinuz-2.6.5-1.358 ro root=LABEL=/1 rhgb quiet
```

I will come to this point again in the next text.

Since the current boot loader is of Fedora, I need to mention my RedHat installation path in this file. I add the following section to this file.

```
title Redhat 9
        root (hd0,0)
        kernel /vmlinuz-2.4.20-8 ro root=/dev/hda7
        initrd /initrd-2.4.20-8.img
```

Before trying to understand this piece of code, please recall that my Redhat9 installation uses three partitions, which are :

```
        /dev/hda1 ( /boot ) or (hd0,0)
        /dev/hda6 ( swap ) or (hd0,5)
        /dev/hda7 ( / )  or (hd0,6)
```

Since my kernel boot files resides on a separate (/boot) partition, which is /dev/hda1, that is why I have placed **root (hd0,0)** below title Redhat line. And since the boot partition is separate, all paths are relative to that. So that is why the kernel and initrd lines do not have the /boot mentioned in them.

Once I updated my /etc/grub.conf of my Fedora installation and rebooted, I was able to see and run my Redhat9 installation without any problems.

Alternative method:

There is another method to make this all work. That method is based on the assumption that you want to restore the boot loader of Redhat9 installation and call / access / boot the newly installed Fedora installation from there. For that you need to follow the following steps.

Boot from Redhat9 CD 1, on the install boot prompt, type: **linux rescue** . The rescue mode will ask you that it will try to find your Redhat9 partitions and mount them. Choose ***continue*** here. You will be presented a prompt, with an information that your Redhat9 installation has been found and has been mounted under the mount point: (/mnt/sysimage) . You need to ***chroot*** to this mount point by:

```
bash $ chroot  /mnt/sysimage
[root@mainserver /]#
```

This is your Redhat9 installation area. Once there , edit your `/etc/grub.conf` and introduce the section for your Fedora installation, like:

```
[root@mainserver /]# vi /etc/grub.conf
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/hda7
#          initrd /initrd-version.img
#boot=/dev/hda
default=0
timeout=10
splashimage=(hd0,0)/grub/splash.xpm.gz
title Red Hat Linux (2.4.20-8)
        root (hd0,0)
        kernel /vmlinuz-2.4.20-8 ro root=/dev/hda7  hdd=ide-scsi
        initrd /initrd-2.4.20-8.img
title Windows
        rootnoverify (hd0,2)
        chainloader +1
title Fedora Core (2.6.5-1.358)
        root (hd0,4)
        kernel /boot/vmlinuz-2.6.5-1.358 ro root=/dev/hda5 rhgb quiet
        initrd /boot/initrd-2.6.5-1.358.img
```

Notice the presence of **`/boot`** in the *kernel* and *initrd* lines in this new section. And notice the absence of `/boot` in the *kernel* and *initrd* lines in the Redhat9 section.

Once you update your `/etc/grub.conf` in this Redhat9 installation area. You need to re-install your Redhat9 version of GRUB boot loader on the MBR of your hard drive before rebooting. Do this by :

```
[root@mainserver / ]# grub-install /dev/hda
```

Once done, you need to exit this *chroot* environment and reboot normally. For that use :

```
[root@mainserver / ]# sync

[root@mainserver / ]# exit

[root@mainserver / ]# exit
```

The second exit above will reboot the system. And you will get your system back again with added menu for Fedora.

Chapter 16: **Advance permissions**

I decided to add this chapter at the end. It is mainly derived from the O'Reilly book "Practical Security" . I would briefly describes *setgid*, *setuid*, *sticky bits*, *umask* and *sudo*.

Setuid, setgid and sticky bit:

Till now you have used three octal numbers for permissions for three types of users, i.e. Owner, group and others. These three octal numbers / digits have values from 000 till 777 , representing: "*no permissions to any one*" till "*all permissions for everyone*", respectively. There are in fact **four** octal number positions designed for file and directory permissions. The fourth number happens to be on the left side of any permission set, and if not present, then is assumed to be 0 (zero). This number can be 1 or 2 or 4 maximum and it's special meaning are below:

1 Sticky bit
2 Set group ID on execution
4 Set user ID on execution

Now first **sticky bit**:

If sticky bit permission is applied on a file like :

```
[root@mainserver / ]# chmod 1755 myfile.txt
```

, then as per the man page:

"On older Unix systems, the sticky bit caused executable files to be hoarded in swap space. This feature is not useful on modern VM systems, and the Linux kernel ignores the sticky bit on files. Other kernels may use the sticky bit on files for system-defined purposes. On some systems, only the superuser can set the sticky bit on files."

What this means is that sticky bit on files is useless in linux.

Sticky bit on a file can be checked by the *ls* command. The " ***T*** " indicates that this file has sticky bit on.

```
[root@mainserver / ]# ls -l
-rw-r--r-T    1 root     root         4847 Jul  8 12:26 myfile.txt
```

And if sticky bit is implemented on a directory like:

```
[root@mainserver / ]# chmod 1777 mydirectory
```

, then as per the man page:

"When the sticky bit is set on a directory, files in that directory may be unlinked or renamed only by root or their owner. Without the sticky bit, anyone able to write to the directory can delete or rename files. The sticky bit is commonly found on directories, such as /tmp, that are world-writable."

What this means is that sticky bit on a directory allows users to create files in that directory with a facility that only root or the user who created the file will be able to delete or rename the file.

Sticky bit on a directory can be checked by the *ls* command. The "***t***" indicates that this file has sticky bit on. A common example is the `/tmp` directory on Linux.

```
[root@mainserver / ]# ls -l
drwxrwxrwt   21 root     root         4096 Jul 13 17:39 tmp
```

Now **setuid** and **setgid**:

Sometimes, unprivileged users must be able to accomplish tasks that require privileges. An example is the *passwd* program, which allows you to change your password. Changing a user's password requires modifying the password field in the `/etc/passwd` file. However, you should not give a user access to change this file directly- the user could change everybody else's password as well! Likewise, the mail program requires that you be able to insert a message into the mailbox of another user, yet you should not to give one user unrestricted access to another's mailbox.

To get around these problems, linux allows programs to be implemented with special privileges. Processes executing these programs can assume another UID or GID

when they're running. A program that changes its UID is called a SUID program (set-UID); a program that changes its GID is called a SGID program (set-GID). A program can be both SUID and SGID at the same time.

When a SUID program is run, its effective UID becomes that of the owner of the file, rather than of the user who is running it.

If *setuid* permission is applied on a file like :

```
[root@mainserver / ]# chmod 4511 myprogram.sh
```

then *myprogram.sh* has the ownership and permissions as :

```
-r-s--x--x    1 root     admin       16336 Feb 14  2003 myprogram.sh
```

,then this means that whenever any user, like *kamran* logs on to the system and runs this program then this program is run as not user *kamran* but user *root* !!!!. The " ***s*** " in the owner's permission set indicates that this file has *setuid* bit on.

If *setgid* permission is applied on a file like :

```
[root@mainserver / ]# chmod 2711 myprogram.sh
```

then *myprogram.sh* has the ownership and permissions as :

```
-rwx--s--x    1 root     admin       16336 Feb 14  2003 myprogram.sh
```

,then this means that any process that executes this SGID program has its effective GID changed to the program's GID (*admin* in this case). Files created by the process can have their primary group set to this GID as well, depending on the permissions of the directory in which the files are created. The " ***s*** "in the group permission set indicates that this file has *setgid* bit on.

The *setgid* bit is normally useful on directories. For example you have a directory named *newproject* and it has ownership as **`root:students`** and permissions like:

```
[root@mainserver / ]# chmod 2770 myprogram.sh
[root@mainserver / ]# ls -l
drwxrws---    2 root     students     4096 Jul 13 21:38 newproject
```

This means that any user who creates an object within this directory will effectively create that object with *students* in the group ownership.

In other words, the SGID bit on a directory controls the way that groups are assigned for files created in the directory. If the SGID bit is set, files created in the directory have the same group as the directory if the process creating the file also is in that group. Otherwise, if the SGID bit is not set, or if the process is not in the same group, files created inside the directory have the same group as the user's effective group ID (usually the primary group ID).

Setuid and *setgid* can be security risk you can find files which have setuid and setgid set on them by:

```
[root@mainserver / ]# find / \(-perm -004000 -o -perm -002000 \)
        ➥       -type f -print
```

This *find* command starts in the root directory (/) and looks for all files that match mode 002000 (SGID) or mode 004000 (SUID). The -type f option causes the search to be restricted to files. The -print option causes the name of every matching file to be printed.

Note that if you are using NFS, you should execute find commands only on your file servers. You should further restrict the find command so that it does not try to search networked disks. Otherwise, use of this command may cause an excessive amount of NFS traffic on your network. To restrict your find command, use the following:

```
[root@mainserver / ]# find / \( -local -o -prune \)
        ➥       \( -perm -004000 -o -perm -002000 \) type f -print
```

Alternatively, if your find command has the -xdev option, you can use it to prevent find from crossing filesystem boundaries. To search the entire filesystem using this option means running the command multiple times, once for each mounted partition.

Be sure that you are the superuser when you run find, or you may miss SUID files hidden in protected directories.

Umask:

The umask (UNIX shorthand for "user file-creation mode mask") is a four-digit octal number that UNIX uses to determine the file permission for newly created files. Every process has its own umask, inherited from its parent process.

The *umask* specifies the permissions you do ***not*** want given by default to newly created files and directories. *umask* works by doing a bitwise AND with the bitwise complement of the umask. Bits that are set in the umask correspond to permissions that are not automatically assigned to newly created files.

By default, most UNIX versions specify an octal mode of 666 (any user can read or write the file) when they create new files. Likewise, new programs are created with a mode of 777 (any user can read, write, or execute the program).

The most common umask values are 022, 027, and 077. A umask value of 022 lets the owner both read and write all newly created files, but everybody else can only read them:

```
 0666    default file-creation mode
(0022)   umask
 0644    resultant mode
```

A umask value of 077 lets only the file's owner read all newly created files:

```
 0666    default file-creation mode
(0077)   umask
 0600    resultant mode
```

A simple way to calculate umask values is to remember that the number 2 in the umask turns off write permission, while 7 turns off read, write, and execute permission.

A umask value of 002 is commonly used by people who are working on group projects. If you create a file with your umask set to 002, anyone in the file's group will be able to read or modify the file. Everybody else will only be allowed to read it:

```
 0666   default file-creation mode
(0002)  umask
 0664   resultant mode
```

On many UNIX systems, the default umask is 022. This is inherited from the init process, as all processes are descendants of init. Some systems may be configured to use another umask value, or a different value may be set in the startup files.

The designers of these systems chose this umask value to foster sharing, an open computing environment, and cooperation among users. Most prototype user accounts shipped with UNIX operating systems specify 022 as the default umask, and many computer centers use this umask when they set up new accounts. Unfortunately, system administrators frequently do not make a point of explaining the umask to novice users, and many users are not aware that most of the files they create are readable by every other user on the system.

A recent trend among computing centers has been to set up new accounts with a umask of 077, so a user's files will, by default, be unreadable by anyone else on the system unless the user makes a conscious choice to make them readable.

Common umask settings and their effects:

umask	User Access	Group Access	Other
0000	all	all	all
0002	all	all	read, execute
0007	all	all	none
0022	all	read, execute	read, execute
0027	all	read, execute	none
0077	all	none	none

Sudo:

Best explanation is the man page itself:

“sudo allows a permitted user to execute a command as the superuser or

another user, as specified in the sudoers file. The real and effective uid and gid are set to match those of the target user as specified in the passwd file (the group vector is also initialized when the target user is not root). By default, sudo requires that users authenticate themselves with a password (NOTE: by default this is the user's password, not the root password). Once a user has been authenticated, a timestamp is updated and the user may then use sudo without a password for a short period of time (5 minutes unless overridden in sudoers).

sudo determines who is an authorized user by consulting the file `/etc/sudoers`. *By giving sudo the -v flag a user can update the time stamp without running a command. The password prompt itself will also time out if the user's password is not entered within 5 minutes (unless overridden via sudoers)."*

What this says is that sudo is used / can be used, by administrator (root), to grant execute permissions to specific users. For example, users can be granted to execute the mount command, which is by default only executable by root. The benefit of using sudo is that you don't have to mess with the *setuid* or *setgid* bits of the target program, e.g. Mount . Rather you just mention the user or group in the `/etc/sudoers` file and that is it. Users can use sudo command along mount command to get their work done.

The `/etc/sudoers` file is quite easy to understand. But remember you have to edit it ***NOT*** with ***vi***. You have to use ***visudo*** command.

```
[root@mainserver / ]# visudo
```

Assume you want to give cdrom mount and unmount ability to the group named *students* along the ability to shutdown the local machine.

You would add the following lines to the /etc/sudoers file.

```
[root@mainserver / ]# visudo

%students  ALL=/bin/mount /dev/cdrom,/bin/umount /mnt/cdrom
%students  localhost=/sbin/shutdown -h now
```

Now the users (members of group students) would use the *sudo* command to mount various file systems.

```
[root@mainserver / ]# sudo mount /dev/cdrom /mnt/cdrom
[root@mainserver / ]#
```

Appendix A (Wiring / cabling)

Here are few things which might prove handy at the time of need.

Wiring schemes / cable connector schemes

If you hold an RJ-45 connector in your hand in such a way that the eight pins are facing you, the locking clip is away from you or towards back of your hand and the channel to insert cable in this connector is facing downwards, then the pins facing towards you are numbered 1 to 8 from left to right.

In a 10 Mbps or 100 Mbps network layout, only pins 1,2,3 and 6 are used. Rest are *NOT* used. It is a miss conception amongst many that if four wires are used then it is a 10 Mbps or Half duplex cable and if it is an eight wire cable then it is 100 Mbps or Full duplex cable. It is *NOT* like that.

1) Straight through cable

Used to connect dis-similar / dislike devices. PC to Hub/Switch. Router's ethernet port to Switch. Up linking Switch to switch by using one switch's up link port and the other's non-uplink / normal port.

To make a 4 wire (2 pair) straight through cable use:

1	-	1	(White Green	-	White Green)
2	-	2	(Green	-	Green)
3	-	3	(White Orange	-	White Orange)
6	-	6	(Orange	-	Orange)

To make a 8 wire (4 pair) straight through cable use:

1	-	1	(White Green	-	White Green)
2	-	2	(Green	-	Green)
3	-	3	(White Orange	-	White Orange)
4	-	4	(Blue	-	Blue)
5	-	5	(White Blue	-	White Blue)
6	-	6	(Orange	-	Orange)
7	-	7	(White Brown	-	White Brown)
8	-	8	(Brown	-	Brown)

2) Cross over cable

Used to connect similar devices. Like PC to PC. Router's ethernet port to PC's ethernet port directly. Up linking switches by using non-uplink / normal ports of both switches.

To make a 4 wire (2 pair) cross over cable use:

1 (White Green)	-	3 (White Green)
2 (Green)	-	6 (Green)
3 (White Orange)	-	1 (White Orange)
6 (Orange)	-	2 (Orange)

To make a 8 wire (4 pair) cross over cable use:

1 (White Green)	-	3 (White Green)
2 (Green)	-	6 (Green)
3 (White Orange)	-	1 (White Orange)
4 (Blue)	-	4 (Blue)
5 (White Blue)	-	5 (White Blue)
6 (Orange)	-	2 (Orange)
7 (White Brown)	-	7 (White Brown)
8 (Brown)	-	8 (Brown)

3) Roll over cable

Also known as console cable. Used to connect Router's console port to PC's serial port for Router administration. It is normally available in the form such as it has RJ-45 connectors on both ends and one of the end is further converted by using a RJ-45 to DB9 converter.

Here is the scheme of this cable (RJ-45 to RJ 45):

1	-	8
2	-	7
3	-	6
4	-	5
5	-	4

6 - 3
7 - 2
8 - 1

In case you want to make one yourself, get an RJ-45 and a DB9 pin-less connector and connect as below:

RJ45 - DB9
1 - 8
2 - 6
3 - 3
4 - 5
5 - 5
6 (unused)
7 - 4
8 - 7

One more thing I would like to mention here, just in case you forget it. Cisco Routers and Switches, once you connect to them on their console ports, need 9600 8N1 setting in your connecting software, like *Minicom* (linux) or *Hyper terminal* (windows)

4) PC serial console cable

Also known as null modem cable. Used to connect PCs over Serial port. Also can be used to log on to a headless Linux PC using another computer using *Hyper terminal* (windows) or *Minicom* (linux).

Both sides of the cable have DB9 (9pin) connectors (pin-less connectors). (The ones like serial mouse has). connect as:-

1 - 4
2 - 3
3 - 2
4 - 1
5 - 5
6 (unused)
7 - 8
8 - 7
9 - 9

This is an eight wire cable and you can easily make it of the maximum length 15 meters , using commonly available cat-5 ether-net cable (as it has eight wires too).

*Appendix **B*** **(Network Addresses / Classes)**

Network addresses and classes are also sometimes forgotten by administrators. Here are the schemes:

	Class A	***Class B***	***Class C***	***Class D (Multi cast)***	***Class E (Research)***
First Octet Range	1 to 126	128 to 191	192 to 223	224 to 239	240 to 255
Valid Network Numbers	1.0.0.0 to 126.0.0.0	128.1.0.0 to 191.254.0.0	192.0.1.0 to 223.255.254.0		
Number of Networks	2^7 -2	2^{14} -2	2^{21} -2		
Number of Hosts per Network	2^{24} -2	2^{16} -2	2^8 -2		
Size of Network part of Address	1	2	3		
Size of Host part of Address	3	2	1		
Size of Network part, in bits	8	16	24		
Size of Host part, in bits	24	16	8		
Default Subnet mask	255.0.0.0	255.255.0.0	255.255.255.0		
Reserved Network Numbers	0.0.0.0 127.0.0.0	128.0.0.0 191.255.0.0	192.0.0.0 223.255.255.0		
Valid Private IP Addresses	10.0.0.0 to 10.255.255.255	172.15.0.0 to 172.31.255.255	192.168.0.0 to 192.168.255.255		

Index

Alphabetical Index