Filters

A filter is a command/program which gets its input from **stdin** (standard input), and send its output to **stdout** (standard output). A filter command can be used anywhere in a pipeline. Some simple filters are - pr, head, tail, cut, sort, uniq, and tr. Some filters using regular expression are grep, egrep and sed. Note: **Input streams** provide input to programs, usually from terminal keystrokes. **Output streams** print text characters, usually to the terminal.

Some simple filter commands

1. **pr command**: It is a print formatting filter command. It converts text file for printing. It usually writes files to stdout. It can do paginating, outputting in multi-column format, merging all files, printing all files in parallel one per column, etc. By default a 5 line header is printed at each page. The third line of the header prints the date, the file name, and the page number, and other four lines are blank. The default footer also has 5 lines, all of which are blank.

Syntax: pr [OPTION] … [FILE] …

Options:

-d	double space the output
-D	use FORMAT for the header date
-h	use a centered HEADER instead of filename in page header
-l	set the page length
-m	print all files in parallel, one in each column,
-W	set page width

Examples:

1) pr file1.txt > lpr : Prints the file1.txt
2)pr –l 50 file1.txt : sets the page length to 50
3) pr -2 file1 : displays the file as 2 column text
4) pr –h “Student Data” file1 > student.txt : prepare with header and write into student.txt which can be then printed.
5) pr -l 50 -W 80 -h " Linux " –D "%A, %d %B %Y %T" notes.txt : displays the contents of the file notes.txt breaking into different pages with 50 lines per page and each line having 80 character width. There will be a five line header with its third line as “Friday, 29 August 2014 11:30:35 Linux Page 1”, and a five line footer.

2. **head command**: It outputs the first part of files. It displays the first 10 lines of each FILE to standard output.

Syntax: head [OPTION]... [FILE]...

Options:

-c	print the first K bytes of each file
-n	print the first K lines instead of the first 10;
-q	never print headers

Examples:

1) head file1.txt : displays the first 10 lines of file1.txt
2) head -13 file1.txt : displays the first 13 lines of the file file1.txt
3) head *.txt | less : displays the first 10 lines of all the .txt files in the working directory

3) **tail command**: It outputs the last part of files. It displays the last 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name

Syntax: tail [OPTION]... [FILE]...

Options:

-c	output the last K bytes
-n	output the last K lines, instead of the last 10;
-q	never output headers giving file names
-v	always output headers giving file names

Ecaxamples:

1) tail file1 : displays the last 10 lines of file1
2) tail -12 header.php : to display the last 12 lines of the file header.php
3) tail *.txt | more : displays the last 10 lines of all the .txt files in the working directory

4. **cut command**: The cut command can extract fields from files. It is very useful for parsing system configuration files. We can specify the field separator and field wanted or we can specify to breakup a line based on bytes.

Syntax: **cut OPTION... [FILE]...**

Options:
- -b select only these bytes
- -c select only these characters
- -f select only these fields
- -s do not print lines not containing delimiters
- -d use the specified delimiter instead of TAB

Use one, and only one of -b, -c or -f. Each range is one of:

Examples: 1) cut –c5 file1 : cut 5th character from the file1

2) cut –f2 file1 : cut 2nd field from the file1

3) cut –c5-10 file1 : cut from 5th to 10th character from the file1

4)grep /home /etc/passwd cut -d':' -f6 : The grep command gives a list of regular users from the /etc/passwd file. The 6th field is displayed as delimited by a :

5. **sort command**: It sorts lines of text files. It writes sorted concatenation of all FILEs to standard output.

Syntax: **sort [OPTION]... [FILE]...**

Options:
- -d consider only blanks and alphanumeric characters
- -f ignore-case (lower case or upper case characters)
- -g compare according to general numerical value
- -M month-sort (unknown < `JAN' < ... < `DEC')
- -h compare human readable numbers (e.g., 2K 1G)
- -n compare according to string numerical value
- -r reverse the result of comparisons

Examples: **1)** sort file

2) sort file1 > file1.sort

3) sort file*.txt > allfiles.sort

4) sort –f file1

5) sort –r file1 : sort in reverse order.

6) sort -k2 test.txt **:** sort the file "test.txt" according to the characters starting at the second column

6. **uniq command:** It removes duplicate lines from a sorted file.

Syntax: uniq [*OPTION*]... [*INPUT* [*OUTPUT*]]

uniq filters out adjacent, matching lines from input file *INPUT*, writing the filtered data to output file *OUTPUT*.

Options:
- -c : Prefix lines with a number representing how many times they occurred.
- -d : Only print duplicated lines.
- -i : ignore-case
- -s N : Avoid comparing the first N characters of each line when determining uniqueness.
- -u : Only print unique lines.

Examples: 1) uniq file1 : removes duplicate lines from file1 and displays on screen

2) uniq file1 file2 : removes duplicate lines from file1 and writes to file2

3) uniq –c file1 4) uniq –d file1 5) uniq –u file1

7. **tr command:** The tr command is a character based translator used to replace one character or a set of characters with another. It is also used to remove a character from the text.

Syntax: tr *string1 string2*

Options:
- -C Complement the set of characters in string1 (except)
- -c Same as -C but complement the set of values in string1.
- -d Delete characters in string1 from the input.
- -s squeezing repeated characters

Examples: Let s="AbCd"

1) echo $s | tr ['A-Z'] ['a-z'] : To translate all upper case letters to lower case letters.

3) cat file1 | tr –d '\t' > file2

4) tr '{}' '()' < file1 >fil2

Filters using regular expression

A regular expression is a string that can be used to describe several sequences of characters. Some filter commands using regular expression are grep, egrep, sed etc.

1. grep command: (general regular expression parser) : It is used to find patterns in files. It is a useful search tool. It searches filenames for lines that match the regular expression and displays them. That is, it processes text in the file line by line and prints any line which matches the pattern.

Syntax: **grep [OPTIONS] PATTERN [FILE...]**

Options:

-i : ignore case distinctions
-w : force PATTERN to match only whole words
-x : force PATTERN to match only whole lines
-v : select non-matching lines
-c : print only a count of matching lines per FILE
-n : print line number with output lines
-h : suppress the file name
-o : show only the part of a line matching PATTERN
-E **:** Treats pattern as an extended regular expression (ERE)

Examples:

1) grep /home /etc/passwd : It displays a list of all regular user accounts. It searches for all lines that contain the text /home in the /etc/passwd file.
2) env | grep ^HO :To find all environment variables that begin with HO
3) grep d$ football :To find words ending with d
3) grep int student.cpp :find the keyword int in the file student.cpp
4) grep -vn int student.cpp :lines (with line numbers), that do not contain the searched keyword int
5) grep -i 'hello world' menu.h main.c :lines containing the pattern “hello world” in menu.h and main.
6) ls | grep php :files with php in their names in the current directory
7) grep ‘s*’ student : find lines with words starting with ‘s’
8) grep ‘[Ss]’ : find lines with words starting with ‘s’ or ‘S’
9) grep ‘a|b’ football` : find lines with words containing ‘a|b’

2. egrep command: egrep is extended version of grep. egrep is equal to grep -E.
egrep supports Extended Regular Expression (ERE) patterns.
In basic regular expressions the metacharacters **?**, +, **{**, |, **(**, and **)** lose their special meaning; instead use the backslashed versions **?**, \+, **{**, \|, **(**, and **)**.

Some extended regular expressions are:

Expression	Description
\+	**Matches one or more occurrence of the previous character**
\?	**Matches zero or one occurrence of the previous character**

egrep is efficient and fast. It treats meta-characters as is and doesn’t substitute them as strings like in grep. Hence we are free from the burden of escaping them as in grep. In case of egrep, even if you do not escape the meta-characters, it would treat them as special characters and substitute them for their special meaning instead of treating them as part of string.

Example : Following 3 examples search for lines which contains exactly two consecutive p characters:

```
1) egrep p{2} *                    -- Output: Knoppix
2) grep pp *                       -- Output: Knoppix
3) grep -E p{2} *                  -- Output: Knoppix
4) egrep   '(f|g)ile' check_file          --  Output: It is a binary file
5) grep        ‘a|b’ football`          -- find lines with words containing ‘a’ or ‘b’ ( considering the
                                          special meaning of |
```

Syntax: egrep [options] [regexp] [files]

The options of egrep are same as grep. Few of them are shown below.

-v : select non-matching lines
-c : print only a count of matching lines per FILE
-n : print line number with output lines
-o : show only the part of a line matching PATTERN

3. sed command:

sed is a **s**tream **ed**itor for filtering and transforming text. It can perform many functions on file like, searching, find and replace, insertion or deletion. Most common use of SED command is for substitution or for find and replace. It supports regular expression to perform complex pattern matching.

By default, the sed command replaces the first occurrence of the pattern in each line and it won't replace the second, third…occurrence in the line.

Syntax: sed OPTIONS... [SCRIPT] [INPUTFILE...]

Eg:

1) sed 's/unix/linux/' file.txt : Here the "s" specifies the substitution operation. The "/" are delimiters. The "unix" is the search pattern and the "linux" is the replacement string.

2) **sed 's/unix/linux/2' geekfile.txt** : To replace the 2th occurrence of a pattern in a line. The /1, /2 etc flags are used to replace the first, second occurrence of a pattern in a line.

3) **sed 's/unix/linux/g' geekfile.txt** : To Replace all the occurrence of the pattern in a line. /g is for global replacement.

4) **sed 's/unix/linux/3g' geekfile.txt** : Replacing from 3rd occurrence to all occurrences in a line.

5) **sed '3 s/unix/linux/' geekfile.txt** : Replacing string on the 3rd line number

6) **sed -n 's/unix/linux/p' geekfile.txt** : The -n option along with the /p print flag to display only the replaced lines.

8) **sed '5d' filename.txt** : To delete 5th line.

9) **sed '$d' filename.txt** : To Delete a last line

10) **sed '3,6d' filename.txt** : To delete 3rd line to 6th line

11) **sed '12,$d' filename.txt** : To delete 12th to last line.

12) **sed '/abc/d' filename.txt** : To Delete line having 'abc'.

Understanding various Servers

1. **DHCP Server**:

A DHCP server is a network server that automatically provides and assigns IP addresses and other network configuration information to computers on a network. It allows a client to obtain its IP address from network server when it joins network. It relies on a standard protocol known as Dynamic Host Configuration Protocol (DHCP) to respond queries by clients. This is a standardized network protocol that is used by network devices to configure the IP settings of another device such as a computer, laptop or tablet.

For Example: When we connect our computer or mobile in a network, it need a unique ip address to be identified in a network. DHCP server program (now a days in our device) requests for ip address and gets assigned.

DHCP reduces the amount of time required to configure clients. It allows one to move a computer to various networks and be configured with the appropriate IP address, gateway and subnet mask. For ISP's it conserves the limited number of IP addresses it may use.

DHCP servers maintain a database of available IP addresses and configuration information, which is used to assign IP addresses to client devices. A client configured for DHCP will send out a broadcast request to the DHCP server requesting an address. DHCP servers typically grant IP addresses to clients only for a limited interval. The DHCP server will then issue a "lease" (for a period of time) and assign it to that client. The time period of a valid lease can be specified on the server. DHCP clients are responsible for renewing their IP address before that interval has expired, and must stop using the address once the interval has expired, if they have not been able to renew it.

DHCP assignment:

- Lease Request: Client broadcasts request to DHCP server with a source address of 0.0.0.0 and a destination address of 255.255.255.255. The request includes the MAC address which is used to direct the reply. (MAC – Media Access Control address is a unique identification number for network interface card and is assgned by the manufacturer.)
- IP lease offer: DHCP server replies with an IP address, subnet mask, network gateway, name of the domain name servers, duration of the lease and the IP address of the DHCP server. (subnet mask is a number that defines a range of IP addesses that can be used in a network)
- Lease Selection: Client receives offer and broadcasts to all DHCP servers that will accept given offer so that other DHCP server need not make an offer.
- The DHCP server then sends an ACK to the client. The client is configured to use TCP/IP.
- Lease Renewal: When half of the lease time has expired, the client will issue a new request to the DHCP server.

DHCP overview:

- host broadcasts “DHCP discover” msg
- DHCP server responds with “DHCP offer” msg
- host requests IP address: “DHCP request” msg
- DHCP server sends address: “DHCP ack” msg
-

The dhcp package and **isc-dhcp-server** contain DHCP server. To install the package as the super user:

```
# yum install dhcp (in RedHat & Fedora)
# apt-get install isc-dhcp-server (in Debian)
```

The DHCP

- Can renew its lease on address in use
- Allows reuse of addresses (only hold address while connected or “on”)
- Support for mobile users who want to join network.

Installing the dhcp package creates a file, `/etc/dhcp/dhcpd.conf`, which is an empty configuration file. The sample configuration file can be found at `/usr/share/doc/dhcp-`<*version*>`/dhcpd.conf.sample`. We should use this file to configure `/etc/dhcp/dhcpd.conf`.

2. DNS Server : –

Computers use IP addresses like 192.168.0.100 but people usually use host names like www.example.com. DNS (Domain Name System) was designed to handle that problem. DNS converts Domain Name into associated IP address and vice versa.

A DNS server is any computer registered to join the Domain Name System. A DNS server runs special-purpose networking software, features a public IP address, and contains a database of network names and addresses for other Internet hosts.

The name and IP address of one or more DNS servers (usually 2 or 3) are usually assigned to the DHCP client from the DHCP host and stored in your *etc/*resolv.conf file.

Eg: search example.com
nameserver 192.168.0.254
nameserver 192.168.0.253

When you request to connect to the host name example.com, the first nameserver is queried. If not found, each subsequent name server is quried. If not found, "Host Not Found" message will be shown.

DNS networking is based on the client / server architecture. A web browser functions as a DNS client (also called DNS resolver) and issues requests to the Internet provider's DNS servers when accessing Web sites. When a DNS server receives a request not in its database, it automatically passes that request to another DNS server or up to the next higher level in the DNS hierarchy as needed. Eventually the request arrives at a server that has the matching name and IP address in its database (all the way to the root level if necessary), and the response flows back through the chain of DNS servers to the requesting client.

All DNS servers are organized in a hierarchy. At the top level of the hierarchy, there are root servers. A root name server is a name server for the Domain Name System's root zone. It directly answers requests for records in the root zone and answers other requests by returning a list of the authoritative name servers for the appropriate top-level domain (TLD). Because of the limits in the DNS and certain protocols, the number of root servers is limited to 13 logical servers. Their names are:

a.root-servers.net,
b.root-servers.net,
c.root-servers.net,
d.root-servers.net,
e.root-servers.net : used by NASA,
f.root-servers.net (192.5.5.241) : used by Internet System Consortium, Inc,
…… , m.root-servers.net

To serve the needs of the public Internet worldwide, the number of root server instances is 376 as of 22 August 2013 and 937 as on Sep 2018. All other DNS servers are installed at lower levels of the hierarchy. ISPs also maintains DNS servers as part of providing Internet connection setup.

BIND is the de facto standard DNS server. It is a free software product and is distributed with most Unix and Linux platforms, where it is most often also referred to as named (name daemon). It is the most widely deployed DNS server.

Example: In browser when we give google.com, first searching is in computer's cache and then search in higher levels in the heirarchy.

3. Apache Server (Web Server) :

The primary function of a web server is to deliver web pages to clients. The term web server means the hardware (the computer) or the software (the computer application) that helps to deliver web content that can be accessed through the Internet. For example, if we install apache in our computer, then our computer becomes a web server.

The communication between client and server takes place using the Hypertext Transfer Protocol (HTTP). Pages delivered are most frequently HTML documents, (which may include images, style sheets and scripts in addition to text content.)

A user agent, commonly a web browser or web crawler (web crawler systematically browses the www for web indexing), initiates communication by making a request for a specific resource using HTTP and the server responds with the content of that resource or an error message if unable to do so. The resource is typically a real file on the server's secondary storage.

While the primary function is to serve content, a full implementation of HTTP also includes ways of receiving content from clients. This feature is used for submitting web forms, including uploading of files.

Web servers supported by GNU/Linux are Apache (httpd), lighttpd, nginx. The most popular of them is Apache. It runs on most available operating systems, and is licensed under the GNU GPL.

Apache supports Virtual hosting, to serve many web sites using one IP address

Apache support server-side scripting, to generate dynamic web pages, using PHP, Python, Perl, Ruby.

To install apache use any of the super user commands:

yum install httpd (Red Hat)

apt-get install apache2 (Debian / Ubuntu)

Some useful modules of Apache that can add functionality are:

1) PageSpeed Module : The mod_pagespeed module is an Apache enhancement that optimizes your content automatically. It can compress data, implement caching, resize files, and remove unnecessary whitespace from configuration files.

2) Security Module : The mod_security module provides a configurable security layer that can accept or deny traffic based upon rules set by the administrator. It is an application firewall that can prevent exposing vulnerabilities to the internet

3) Status Module : The mod_status module provides an overview of your server load and requests.

4) Spamhaus Module : The Spamhaus module enables you to block attackers by denying request from a blacklist of IP addresses that are known to be bad.

5) Rewrite Module : One of the most useful modules for Apache is mod_rewrite. This module allows you to generate unique and easily readable urls for content requested on the server.

6) Proxy Module: mod_proxy is an optional module for the Apache HTTP Server. This module implements a proxy, gateway or cache for Apache. It is possible to set various web framework-based applications as virtual hosts. It is possible to set multiple virtual hosts on a single server. This is a convenient way to partition separate websites and applications.

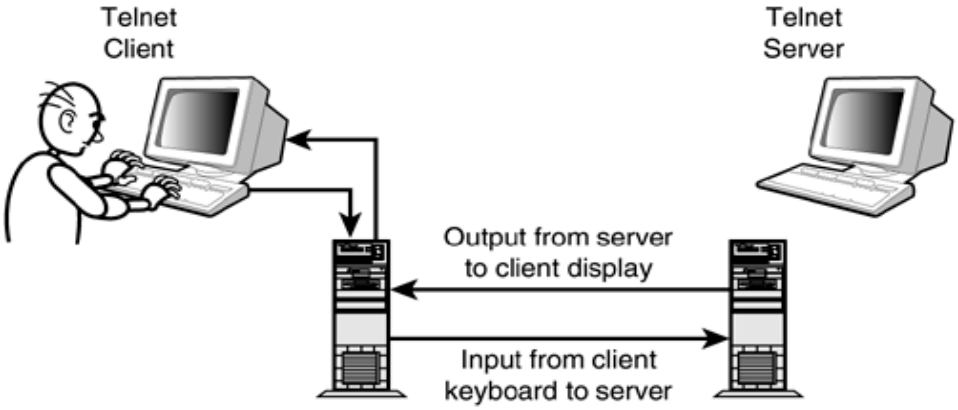
7) Reverse Proxy Module : A reverse proxy is a type of proxy server that takes HTTP(S) requests and transparently distributes them to one or more backend servers. Reverse proxies are useful because many modern web applications process incoming HTTP requests using backend application servers. Reverse proxy can be used to prevent these underlying application servers from being directly accessed. .. They can also be used to distribute the load from incoming requests to several different application servers, increasing performance at scale and providing fail-safeness.

8) Authentication Module : Apache provides the basic framework and directives to perform authentication and access control. The authentication modules provide support for validating passwords against a specific backend. Users can optionally be organized in groups, easing management of access control rules. It provides

- **Backend storage:** Provide text or database files containing the username and groups information
- **User management:** Supply tools for creating and managing users and groups in the backend storage
- **Authoritative information:** Specify whether the results of the module are authoritative

4. **Telnet Service**: Telnet is famous for being the original Internet when the Net first launched in 1969. Telnet stands for 'telecommunications network', and was built to be form of remote control to manage mainframe computers from distant terminals. In those original days of large mainframe computers, telnet enabled research students and professors to 'log in' to the university mainframe from any terminal in the building.

Telnet is a protocol that allows you to connect to remote computers (called hosts) over a TCP/IP network (such as the internet). Using telnet client software on your computer, you can make a connection to a telnet server (that is, the remote host). Telnet allows us to log in to a computer, just as if we were sitting at the terminal.



The telnet command is the client program that you use to do the remote login. Once your username and password are verified, you are given a shell prompt. From there, you can do anything requiring a text console, compose email, read newsgroups, move files around, and so on.

Syntax: telnet [OPTION...] [HOST [PORT]]

The most common way to use telnet is with a hostname. To login to a remote machine, use this syntax:

telnet <hostname>

The following is an example telnet session:

```
$ telnet maple
  Trying 10.0.0.11 ...
  Connected to maple (10.0.0.11).
  login: mike
  Password:
[mike@maple mike]$
```

Here the telnet command is used to log in from a computer named pine to a computer named maple. The computer tries to connect to the telnet port on maple (IP address 10.0.0.11). Because maple is also a Red Hat Linux computer, once the connection is established, we can see the standard login: prompt. Then type the user name (mike) and the password. When the login and password are accepted, we can see the shell prompt for the user named mike.

The telnet service is available by default on Red Hat Linux systems. Some useful options with telnet are:

−a : Automatic login.
−l user : User name.same as −a option, but choose user name.

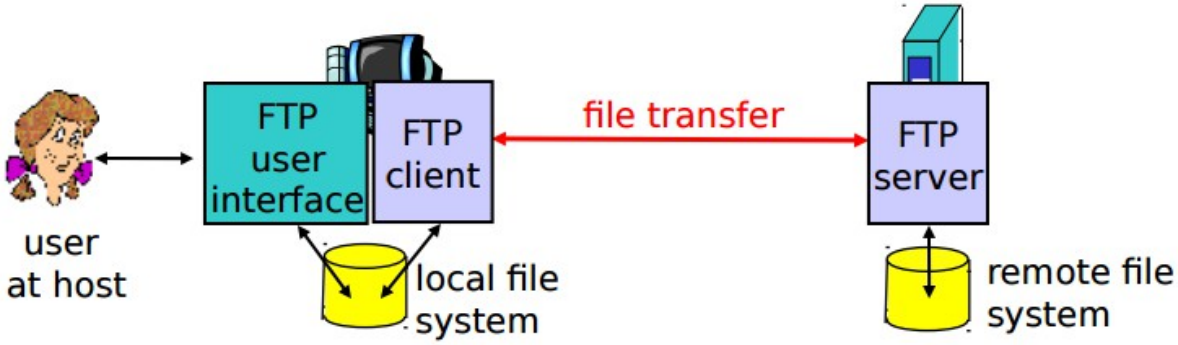
After connecting to a remote computer, we can return by typing Ctrl+]. Some other options used during telnet session are:

? : Print help information.
! : Escape to the shell.
close : If you have an open connection, type close to close it.
display : Shows the operating parameters that are in effect.
logout : Logs off any remote connection in this session and closes the connection.
mode : Tries to enter line mode or character mode.
quit : Close telnet and exit.
z : Suspend the current telnet session.

If telnet session is suspended or escaped to the shell, we can return to our telnet session by typing fg.

Everything is sent in plain text, even passwords. It is not advisable to use telnet over the Internet. Instead, consider the Secure Shell. It encrypts all traffic.

5. **FTP Server**: FTP means **file transfer protocol**. It has been the standard method for sharing files over the internet for many years. It provides tools for sharing files and protecting the computer system. It transfer file to/from remote host.



- FTP host stores files
- Client logs into host
- Client program sends command to get a file
- FTP host downloads the file with error correction

It has client/server model. The client side initiates file transfer (either to/from remote). The server is the remote host. It is the most preferred protocol for data transfer among computers. FTP can be to

- Logging in and establishing a connection with a remote host
- Upload and download files
- Navigating through directories
- Browsing contents of the directories
-

The FTP Server delivered with Red Hat Linux is the Very Secure FTP server (vsFTPd). It was designed to secure the computer system against most common attacks. To connect to those servers from Red Hat Linux, we can either type the URL of that server into a Web browser or we can use the ftp command or graphical ftp windows such as gFTP.

Using the ftp command :

The ftp command is available on every Linux and UNIX systems, for copying files to and from FTP servers. Like telnet, FTP has a command mode to connect directly to a remote computer. For example:

```
$ ftp maple
  Connected to maple.
  220 maple FTP server (Version wu−2.6.1−18) ready.
  530 Please login with USER and PASS.
  Name (maple:mike): jake
  331 Password required for jake.
  Password: *********
  230 User jake logged in.
  Remote system type is LINUX.
  Using binary mode to transfer files.
ftp>
```

In this example, ftp connects to a computer called maple (ftp maple). login name is assumed (maple:mike). We can press Enter key to use the name mike. But here,we logged in as jake and typed the password. The password was accepted and, some information was printed. Then the ftp> prompt was displayed. This command mode includes a lot of commands for moving around the remote file system and for copying files.

FTP directory commands:

The commands from the ftp> prompt that are used to work with both the remote and local directories associated with the FTP connection are:

pwd : Shows the name of the current directory on the remote system.
ls : Lists the contents of the current remote directory. We can use any valid ls options with this command.
Dir : Same as ls.
cd : Use the cd command to move to the directory named.
cdup : Moves up one directory in the file system.
lcd : Lists the name of the current local directory.

If you want to make changes to any of the remote files or directories, use the following commands:

mkdir : Creates a directory on the remote system.
rename : Renames a file or directory on the remote system.
rmdir : Removes a remote directory.
delete : Removes a remote file.
mdelete : Removes multiple remote files.

FTP file copying commands : Before copying files between the remote and local systems, consider the type of transfer. The two types of transfer modes are:

binary : For transferring binary files (such as data files and executable commands). This is also referred to as an image transfer. binany is the default type.
Ascii : For transferring plain−text files.

Most file copying is done with the get and put commands. The mget and mput commands are used to transfer multiple files at once.

get file: Copies a file from the current directory on the remote file system and copies it to the current directory on the local file system.

Eg: 1)get file1 : takes the file file1 from the current remote directory and copies it to the current local directory.
2) get /tmp/options /home/mike/op : opies the remote file /tmp/options and copies it to the local file /home/mike/op
3) get /tmp/sting : copies from the local to the remote system.

mget file . . . : This command lets us to get multiple files at once.
mput file . . . : This command lets us to put multiple files on the remote computer.

The mget, mput commands prompts us before transferring each file.

FTP Exiting Commands :

! : This temporarily exits to the local shell. Type exit to return to the FTP session
close : Closes the current connection.
bye : Closes the connection and exits the ftp command.

When we exit FTP, we can see information about how many files were transferred and how much data was sent during those transfers.

6. **Samba Server:**

Samba is a software package that comes with Fedora and Red Hat Enterprise Linux systems and many other Linux systems. The Samba software package can be obtained from www.samba.org if it is not included in the distribution.

Samba offers a number of ways to share files among Windows, Linux, and Mac OS/X systems. Samba enables to share file systems and printers on a network. File sharing is done using the Common Internet File System (CIFS).

Samba software is secure and robust. The Samba software package contains a variety of daemon processes, administrative tools, user tools, and configuration files. To do basic Samba configuration, start with the Samba Server Configuration window, which provides a graphical interface for configuring the server and setting directories to share. To access features that are not available through the Samba Server Configuration window, we can edit /etc/samba/smb.conf, to configure Samba.

To set up Samba to start automatically when the Linux system starts, type the following:

```
# chkconfig smb on
```

To install all the packages of Samba type the following as root:

yum install samba samba-client samba-doc samba-swat samba-winbind

samba-client package : Contains command-line tools for connecting to Samba or Windows shares, for looking up host addresses to find SMB hosts on the networ etc.

samba-doc package : Contains Samba documentation in HTML and PDF formats.

samba-swat package : Contains a web-based interface for configuring a Samba server.

samba-winbind package : Includes components that allow your Samba server in Linux to become a complete member of a Windows domain, including using Windows user and group accounts in Linux.

Two main services associated with a Samba are:

- smb—This service provides the file and print sharing services that can be accessed by Windows clients.
- nmb—This service can map requests from Windows clients for NetBIOS names so they can be resolved into IP addresses.

To share files and printers with other Linux systems with Samba, only the smb service is required.

Starting the Samba (smb) service:

Starting smb service makes files and printers available from the local system to other computers on the network. In Fedora, to start Samba, type the following from the command line as root:

systemctl enable smb.service
systemctl start smb.service
systemctl status smb.service

To stop the smb, type

systemctl stop smb.service

In RHEL , start the Samba service as follows:

service smb start

To stop, type

service smb stop

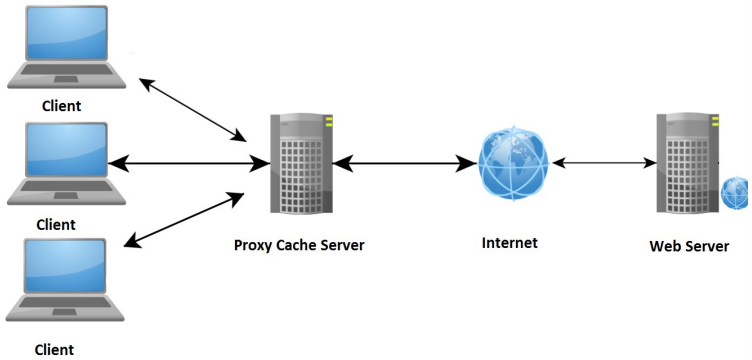
Securing Samba:

- Firewalls—The default firewall for Fedora, RHEL, and other Linux systems prevents any access to local services from outside systems. So, to allow users from other computers to access your Samba service, you must create firewall rules that open one or more ports for selected protocols (TCP in particular).
- While sharing user home directories, set least access required
- Host and user restrictions—Within the Samba configuration files themselves, you can indicate which hosts and users can have access to the Samba server as a whole or to particular shared directories.

7. **Squid Server (Proxy Server)**

In computer networks, a proxy server is a server (a computer system or an application) that acts as an intermediary for requests from clients seeking resources from other servers.

A common proxy application is a caching Web proxy. This provides a nearby cache of Web pages and files available on remote Web servers, allowing local network clients to access them more quickly or reliably. When it receives a request for a Web resource (specified by a URL), a caching proxy looks for the resulting URL in its local cache. If found, it returns the document immediately. Otherwise it fetches it from the remote server, returns it to the requester and saves a copy in the cache.



The cache usually uses an expiry algorithm to remove documents from the cache, according to their age, size, and access history. Two simple cache algorithms are Least Recently Used (LRU) and Least Frequently Used (LFU).

Squid is a full-featured web proxy cache server application which provides proxy and cache services for Hyper Text Transport Protocol (HTTP), File Transfer Protocol (FTP), and other popular network protocols. Squid can implement caching and proxying of Secure Sockets Layer (SSL) requests and caching of Domain Name Server (DNS) lookups, and perform transparent caching. Squid also supports a wide variety of caching protocols, such as Internet Cache Protocol, (ICP) the Hyper Text Caching Protocol, (HTCP) etc. It reduces bandwidth and improves response times by caching and reusing frequently requested web pages. Squid has extensive access controls and makes a great server accelerator. It runs on most available operating systems, and is licensed under the GNU GPL (General Public Licence). Caching frequently requested Web pages, media files and other content accelerates response time and reduces bandwidth congestion.

****