UNIT 5

SIMPLE FILTER COMMANDS

There are some Linux commands that accept input from standard input or files, perform some manipulation on it, and produces some output to the standard output. Since these commands perform some filtering operations on data, they are appropriately called as **filters**. These filters are used to display the contents of a file in sorted order, extract the lines of a specified file that contains a specified pattern, etc.

Commonly used filter commands in Linux are:

Command	Purpose
pr	Prints files to the standard output
head	Prints the first few lines of a file
tail	Displays the last few lines of a file
sort	Sorts and merges files
uniq	Reports repeated lines in a file
tr	Translates characters in a file
cut	Cuts out selected fields of each line of a file

pr

The *pr* command is used to **format a file** to make it look better when printed. This command displays the contents of the specified file adding with suitable headers and footers. The header part consists of the last modification date and time along with file name and page number. General format is:

pr [option] <filename>

Options	Meaning
-l <number>	Changes the page size to specified <number> of lines. (By default, the page size is 66 lines)
-d	Produces double-spaced output.
-h header	Uses the specified header string as the page header.
-t	Omit page headers and footers.
-w width	Set page width to PAGE_WIDTH (default 72) characters for multiple text-column output only, -s[CHAR] turns this off.
-n	Numbers lines
-<number>	Prepares the output in <number> lines

Examples:

pr sample

pr –n sample

pr -2 sample

```
[salmiya@localhost ~]$
[salmiya@localhost ~]$ pr -n Sample

2019-10-13 20:51                Sample                Page 1


     1   This is a file named sample
     2   Showing the result of pr command
     3   Used for Printing
```

head

The head command, as the name implies, **print the top N number of lines** in the specified file. By default, it prints the first 10 lines of the specified files. If more than one file name is provided then data from each file is preceded by its file name. General format is:

head [options]<filename>

Options	Meaning
-n <num>	Print first *num* lines
-c <num>	Print first *num* bytes
-v	Print header giving filenames

Example:

```
head –3 sample
```

tail

This command displays **the end of specified file**. By default, it prints the last 10 lines of the specified file.

tail [options]<filename>

Options	Meaning
-n <num>	Print last *num* lines
-c <num>	Print last *num* bytes
-v	Print header giving filenames

Example:

```
tail -3 sample
```

cut

This command is used to **cut the columns/fields** of a specified file. ie, used for cutting out the sections from each line of files and writing the result to standard output. It can be used to cut parts of a line by byte position, character and field. General format is:

cut option <filename>

Options	Meaning
-b	select only these bytes
-c	select only these characters
-f	select only these fields
-d DELIM	Use character DELIM instead of a tab for the field delimiter.

Note: Without any option specified, the cut command displays an error displays an error.

Examples:

1. To display the first, second and fifth bytes:

```
cut –b 1,2,5sample
```

2. To display bytes in the range:

```
cut –b 1-3,6-9 sample
```

3. To print the second, fifth and seventh character from each line of the file.

```
cut –c 2,5,7 sample
```

4. To print the first field of the file:

```
cut –d " " –f 1sample
```

sort

This command is use to **sort the contents** of a given file based on ASCII values of characters. General format is:

sort [options] <filename>

Options	Meaning
-m <filelist>	Merges sorted files specified in <filelist>
-o <filename>	Stores output in the specified <filename>
-r	Sorts the content in reverse order
-u	Removes duplicate lines and display the sorted content
-n	Numeric-sort
-c	Checks if the file is sorted or not. If the file is sorted, no message will be displayed. Otherwise it will indicate the unsortedness.
-f	Ignore case
-b	Ignore leading spaces
+pos	Starts sort after skipping the posth field
-pos	Stops sorting after the posth field
-t "char"	Uses the specified "char" as delimiter to identify fileds.

Examples:

1. To sort the content of the file use:

```
sort file1
```

2. To sort the content in reverse order:

```
sort –r file1
```

3. To store the output to a file use :

```
sort –o sortfile  file1
```

This command not displays anything, but stores the result in *sortfile*.

4. To sort the file with numeric data:

```
sort –n number.txt
```

uniq

With the help of *uniq* command you can form a sorted list in which every word will occur only once. ie, the *uniq* command reports or **filters out repeated lines** in a file.

uniq [option] <filename>

Option	Meaning
-d	Displays only the duplicate lines
-u	Displays only unique lines
-i	Normally, comparisons are case-sensitive. This option performs case-insensitive comparisons instead
-c	Displays line by eliminating duplicate lines and prefix lines with a number representing how many times they occurred.
-w	It only compares N characters in a line.

Example:-

```
uniq file1
```

To sore the output of *uniq* command, you should specify the destination filename as shown below:

```
uniq file1 outfile
```

tr

The tr (stands for *translate*) command line utility for **translating or deleting characters**. It supports a range of transformations including uppercase to lowercase, squeezing repeating characters, deleting specific characters and basic find and replace. It can be used with pipes to support more complex translation. General format is:

tr [option] Set1 [Set2]

Options	Meaning
-c	Use the complement of Set1
-d	Delete characters in Set1

We can specify the values in set in the following way.

Value	Meaning
[:alnum:]	All letters and digits
[:alpha:]	All letters
[:blank:]	All horizontal white space
[:digit:]	All digits
[:space:]	All horizontal or vertical space
[:lower:]	All lower case letters
[:upper:]	All upper case letters

Examples:

1. To convert lower case to upper case letters

   cat filename | tr "[a-z]" "[A-Z]"

   or

   cat filename | tr "[:lower:]" "[:upper:]"

2. to translate white-space to tabs

   cat filename | tr [:space:] '\t'

3. To delete specified character use –d option

   cat filename | tr –d '<character>'

4. To remove all the numbers

   cat filename | tr –d [0-9]

   cat filename | tr –d [:digit:]

5. To remove all characters except digits
   cat filename | tr –cd [0-9]
6. To remove all spaces
   cat filename | tr –d " "
   cat filename | tr –d ' '
7. To store the result in another file
   cat filename | tr "[:lower:]" "[:upper:]" > result

```
[salmiya@localhost ~]$ cat file1
This is a file for
Showing the Concept of tr command
[salmiya@localhost ~]$
[salmiya@localhost ~]$ cat file1 | tr [:lower:] [:upper:]
THIS IS A FILE FOR
SHOWING THE CONCEPT OF TR COMMAND
[salmiya@localhost ~]$
[salmiya@localhost ~]$ cat file1 | tr [" "] [*]
This*is*a*file*for
Showing*the*Concept*of*tr*command
[salmiya@localhost ~]$
```

Filters Using Regular Expressions: grep, egrep, sed

grep

Grep is used to search for a specified pattern from a specified file and display all lines containing the pattern. *grep* is an acronym for '**globally search a regular expression and print it**'. The command searches the specified input fully (globally) for a match with the supplied pattern and displays it. The pattern that is searched in the file is referred to as the *regular expression*. General format is:

grep [options] pattern <filename>

Options	Meaning
-i	Ignore uppercase and lowercase when comparing.
-v	Print only lines that do not match the pattern.
-c	Print only a count of the matching lines.
-n	Display the line number before each matching line.
<number>	Display the matching lines along with <number> of lines above and below
-w	Match Whole word
-o	Print only the matched part of a matched line
-l	Display list of a filenames only

The most common types of regular expressions are:

Regular Expression	Meaning
abc	Match lines containing the string "abc" anywhere.
^abc	Matches the lines that are starting with "abc."
abc $	Matches the lines that are ending with "abc."
a…c	Match lines containing "a" and "c" separated by any three characters
a.*c	Match lines containing "a" and "c" separated by any number of Characters
*	Match zero or more characters
.	Matches a single character

Example:

1. To find match

   grep "Linux" file1

2. Displaying the count of number of matches

   grep –c "Linux" file1

3. To display the file names that contains the given pattern

   grep –l "Linux" *

4. Checking for the whole words in a file, by default, *grep* matches the given string/pattern even if it found as a substring in a file.

   grep -w "Linux" file1

5. Matching the lines that start with a given string

   grep "^Linux" file1

```
[salmiya@localhost ~]$ cat file1
linux is an open Source OS
It is developed by linus Torvalds
Linux Supports multi tasking
[salmiya@localhost ~]$
[salmiya@localhost ~]$ grep "Linux" file1
      Supports multi tasking
[salmiya@localhost ~]$
[salmiya@localhost ~]$ grep -i "Linux" file1
      is an open Source OS
      Supports multi tasking
[salmiya@localhost ~]$
[salmiya@localhost ~]$ grep -i ^lin file1
   ux is an open Source OS
   ux Supports multi tasking
[salmiya@localhost ~]$
```

egrep

egrep- extended global search for regular expression- offers additional features than *gerp*. **Multiple patterns can be searched** by using pipe symbol (|).

Example:

egrep "linux|unix" file1

```
[salmiya@localhost ~]$ cat file1
Linux is an open source OS
It supports multitasking
Unix is the mother of linux
[salmiya@localhost ~]$
[salmiya@localhost ~]$ egrep "Linux|Unix" file1
Linux is an open source OS
Unix is the mother of linux
[salmiya@localhost ~]$
```

sed

sed stands for **stream editor** and it can perform lot's of function on file like, searching, find and replace, insertion or deletion. The most common use of *sed* is for substitution or for find and replace. By using *sed* you can edit files even without opening it. General format is:

sed [options] 'editcommands' <filename>

Where the '*editcommand's* can be,

Command	Meaning
i	Inserts before line
a	Append after line
d	Deletes line
p	Prints line
q	Quits
s/string1/string2	Substitute String1 by String2

Options used in sed:

Options	Meaning
-n	Suppress automatic printing of pattern space

Example:

1. To delete second line, use:

   sed '2d' file1

2. To inserts the specified star line before each line of the file:

   sed 'i **********' file1

3. To change the third line of the file

   sed '3c \This third line is changed' file1

4. To display the line 2 to 5:

   sed –n '2,5 p' file1

```
[salmiya@localhost ~]$ cat file1
Linux is an open source OS
It supports multitasking
Unix is the mother of linux
[salmiya@localhost ~]$ sed 'i \**********' file1
**********
Linux is an open source OS
**********
It supports multitasking
**********
Unix is the mother of linux
[salmiya@localhost ~]$ sed '2d' file1
Linux is an open source OS
Unix is the mother of linux
[salmiya@localhost ~]$ sed -n '2p' file1
It supports multitasking
[salmiya@localhost ~]$
```

Understanding various servers

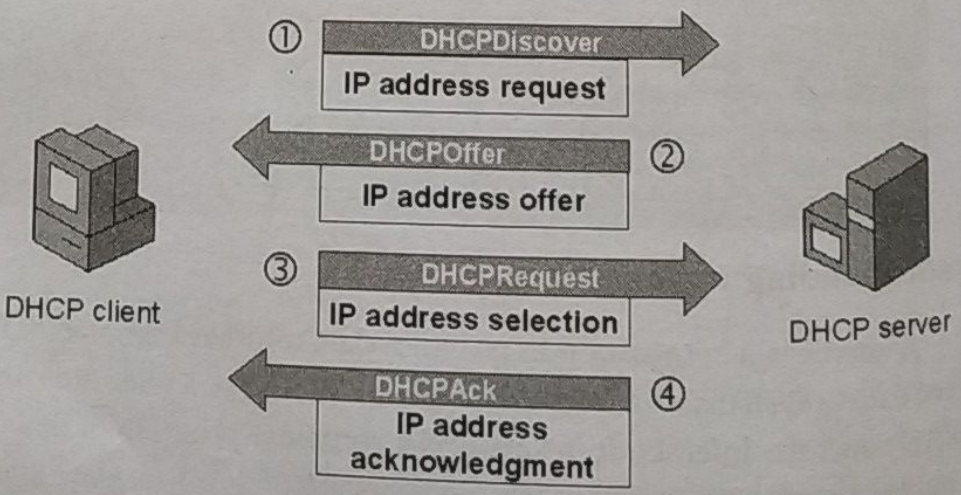
A server is a computer that provides data to other computers. It may serve data to systems on a local area network (LAN) or a wide area network (WAN) over the Internet. ie, a server is a computer program or device that provides a service to another computer program and its user, also known as the client. In a data center, the physical computer that a server program runs on is also frequently referred to as a server.

Many types of servers exist, including web servers, mail servers, and file servers. Each type runs software specific to the purpose of the server. For example, a Web server may run Apache Server, which provides access to websites over the Internet. A mail server provides services for sending and receiving email. A file server is used to share files over a network.

DHCP

DHCP stands for "Dynamic Host Configuration Protocol". DHCP is a protocol that automatically assigns a unique IP address to each device that connects to a network. With DHCP, there is no need to manually assign IP addresses to new devices. Therefore, no user configuration is necessary to connect to a DCHP-based network. Because of its ease of use and widespread support, DHCP is the default protocol used by most routers and networking equipment.

When you connect to a network, your device is considered a client and the router is the server. DHCP provides an automated way to distribute and update IP addresses and other configuration information on a network. A DHCP server provides this information to a DHCP client through the exchange of a series of messages, known as the DHCP conversation or the DHCP transaction. In order to successfully connect to a network via DHCP, the following steps must take place.



1. When a client detects it has connected to a DHCP server, it sends a DHCPDISCOVER request.

2. The router either receives the request or redirects it to the appropriate DHCP server.
3. If the server accepts the new device, it will send a DHCPOFFER message back to the client, which contains the client device's MAC address and the IP address being offered.
4. The client returns a DHCPREQUEST message to the server, confirming it will use the IP address.
5. Finally, the server responds with a DHCPACK acknowled-gement message that confirms the client has been given access for a certain amount of time.

Depending on implementation DHCP server may have three methods of allocating IP-addresses:

Static allocation: The DHCP server allocates an IP address based on a table with MAC address/IP address pairs, which are manually filled. Only requesting clients with a MAC address listed in this table will be allocated an IP address.

Dynamic allocation: A network administrator assigns a range of IP addresses to DHCP, and each client computer on the LAN is configured to request an IP address from the DHCP server during network initialization.

Automatic allocation: The DHCP server permanently assigns a free IP address to a requesting client from the range defined by the administrator. This is like dynamic allocation, but the DHCP server keeps a table of past IP address assignments, so that it can preferentially assign to a client the same IP address that the client previously had.

DNS

The Domain Name Systems (DNS) is the phonebook of the Internet. Each device connected to the Internet has a unique IP address which other machines use to find the device. DNS servers eliminate the need for humans to memorize IP addresses.

Humans access information online through domain names, like example.com. Web browsers interact through Internet Protocol (IP)

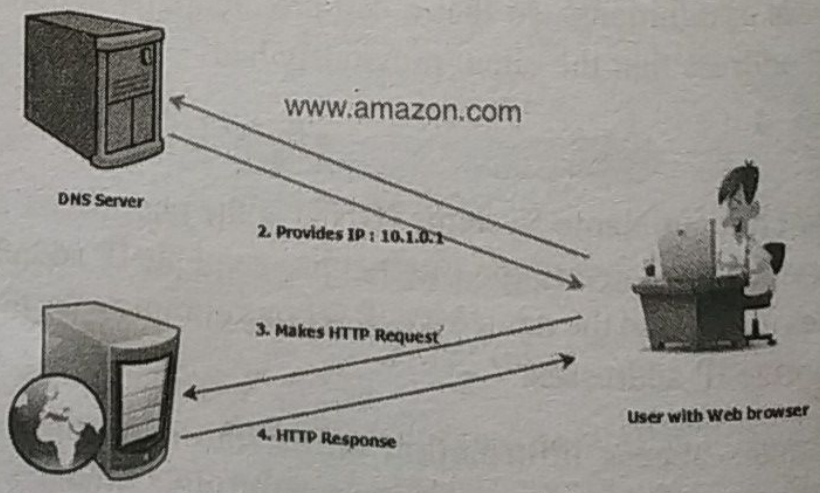
addresses. DNS translates domain names to IP addresses so browsers can load Internet resources.

DNS is an Internet service that translates domain names into IP addresses. Because domain names are alphabetic, they're easier to remember. The Internet however, is really based on IP addresses. Every time you use a domain name, therefore, a DNS service must translate the name into the corresponding IP address. For example, the domain name www.example.com might translate to 198.105.232.4.

A DNS server is a type of name server that manages, maintains and processes Internet domain names and their associated records. In other words, a DNS server is the primary component that implements the DNS (Domain Name System) protocol and provisions domain name resolution services to Web hosts and clients on an IP-based network.

It is primarily designed to locate and deliver websites to end users over the Internet or a private network, a DNS server is developed on typical hardware but runs specialized DNS software. It is always connected to the Internet or a network.

DNS server software comes in dozens, if not hundreds of flavors. The best known version is BIND, which is free and distributed with Linux/ Unix systems. On Microsoft systems, the Microsoft DNS is bundled as part of many Windows Server releases.



SQUID

Squid is a widely-used proxy cache server for Linux and UNIX platforms. It is widely used to improve the speed of the web server by caching frequently requested web pages as well as media files. It is mostly used for HTTP and FTP. However, it is compatible with other protocols such as HTTPS, SSL, Dopher and likewise. But it does not support SOCKS protocol.

Squid acts as a proxy cache. It redirects object requests from clients to the server. When the requested objects arrive from the server, it delivers the objects to the client and keeps a copy of them in the hard disk cache. One of the advantages of caching is that several clients requesting the same object can be served from the hard disk cache. This enables clients to receive the data much faster than from the Internet. This procedure also reduces the network traffic.

The Squid proxy server is installed on a separate server other than the web server. The efficiency of the proxy server is determined whether it is able to recognize the repeated requests and keep the corresponding objects in it or not.

Apart from the actual caching, Squid offers a wide range of features such as distributing the load over intercommunicating hierarchies of proxy servers, defining strict access control lists for all clients accessing the proxy, allowing or denying access to specific web pages with the help of other applications, and generating statistics about frequently-visited web pages for the assessment of the users' surfing habits. Squid is not a generic proxy.

A Squid proxy server is generally installed on a separate server than the Web server with the original files. Squid works by tracking object use over the network. Squid will initially act as an intermediary, simply passing the client's request on to the server and saving a copy of the requested object. If the same client or multiple clients request the same object before it expires from Squid's cache, Squid can then immediately serve it. Squid is provided as free, open source software and can be used under the GNU General Public License (GPL) of the Free Software Foundation. Squid was

originally designed to run on Unix-based systems but can also be run on Windows machines.

Apache

Apache is the most widely used web server software. The official name is Apache HTTP Server, and it's maintained and developed by the Apache Software Foundation. It allows website owners to serve content on the web — hence the name "web server". It's one of the oldest and most reliable web servers, with the first version released in 1995.

Apache is an open source software available for free. It runs on 67% of all webservers in the world. It is fast, reliable, and secure. It can be highly customized to meet the needs of many different environments by using extensions and modules.

Apache is the most widely used Web Server application in Unix-like operating systems.It was designed for Unix environments, more than 90% of installations run on Linux. It's also available for Windows and other operating systems.

It is a modular, process-based web server application that creates a new thread with each simultaneous connection. It supports a number of features; many of them are compiled as separate modules and extend its core functionality, and can provide everything from server side programming language support to authentication mechanism. Virtual hosting is one such feature that allows a single Apache Web Server to serve a number of different websites.

Although we call Apache a web server, it is not a physical server, but rather a software that runs on a server. Its job is to establish a connection between a server and the browsers of website visitors (Firefox, Google Chrome, Safari, etc.) while delivering files back and forth between them (client-server structure). Apache is a cross-platform software, therefore it works on both Unix and Windows servers.

When a visitor wants to load a page on your website, for instance, the homepage or your "About Us" page, their browser sends a request to your

server and Apache returns a response with all the requested files (text, images, etc.). The server and the client communicate through the HTTP protocol and Apache is responsible for the smooth and secure communication between the two machines.

Apache is highly customizable, as it has a module-based structure. Modules allow server administrators to turn additional functionalities on and off. Apache has modules for security, caching, URL rewriting, password authentication, and more.

Advantages

1. Open-source and free, even for commercial use.
2. Reliable, stable software.
3. Frequently updated, regular security patches.
4. Flexible due to its module-based structure.
5. Easy to configure, beginner-friendly.
6. Cross-platform (works on both Unix and Windows servers).
7. Huge community and easily available support in case of any problem.

Disadvantages

1. Performance problems on extremely traffic-heavy websites.
2. Too many configuration options can lead to security vulnerabilities.
3. Requires a strict updating policy that needs to be done regularly without fail.

Telnet

Telnet (**TEL**ecommunication **NET**work) is a networking protocol and software program used to access remote computers and terminals over the Internet or a TCP/IP computer network. Telnet was developed in 1969 and standardized as one of the first Internet standards by the Internet Engineering Task Force (IETF).

Telnet is a computer protocol that was built for interacting with remote computers.The word "Telnet" also refers to the command-line utility

"telnet", available under Windows OS and Unix-like systems, including Mac, Linux, and others.

Telnet utility allows users to test connectivity to remote machines and issue commands through the use of a keyboard. It is a network protocol used on the Internet or local area networks to provide a bidirectional interactive communications facility. Typically, telnet provides access to a command-line interface on a remote host via a virtual terminal connection. The user's computer, which initiates the connection, is referred to as the local computer.

The computer being connected to, which accepts the connection, is referred to as the remote computer. The remote computer can be physically located in the next room, the next town or in another country.

The network terminal protocol (TELNET) allows a user to log in on any other computer on the network. We can start a remote session by specifying a computer to connect to. From that time until we finish the session, anything we type is sent to the other computer.

The Telnet program runs on the computer and connects your PC to a server on the network. We can then enter commands through the Telnet program and they will be executed as if we were entering them directly on the server console. This enables us to control the server and communicate with other servers on the network. To start a Telnet session, we must log in to a server by entering a valid username and password. Telnet is a common way to remotely control Web servers.

The term telnet also refers to software which implements the client part of the protocol. TELNET clients have been available on most Unix systems for many years and are available virtually for all platforms. Most network equipment and OSs with a TCP/IP stack support some kind of TELNET service server for their remote configuration including ones based on Windows NT.

Through Telnet, an administrator or another user can access someone else's computer remotely. On the Web, HTTP and FTP protocols allow you

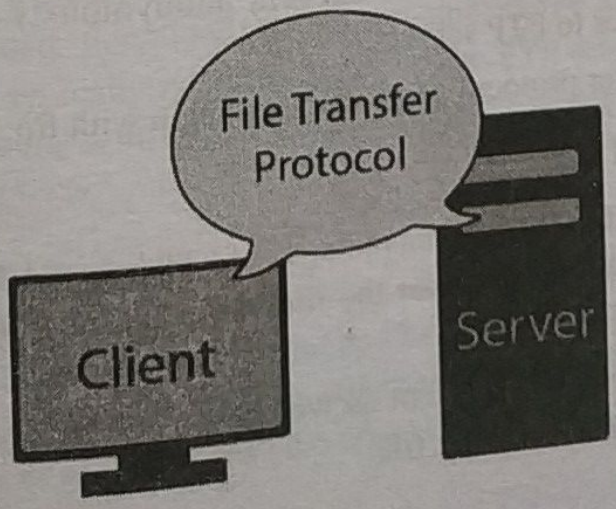
to request specific files from remote computers, but not to actually be logged on as a user of that computer. With Telnet, you log on as a regular user with whatever privileges you may have been granted to the specific application and data on that computer. Telnet is most likely to be used by program developers and anyone who has a need to use specific applications or data located at a particular host computer.

FTP

An FTP Server is a piece of software that is running on a computer and uses the File Transfer Protocol to store and share files. Remote computers can connect anonymously, if allowed, or with a user name and password in order to download files from this server using a piece of software called a FTP Client.

In the simplest of definitions, an FTP Server (which stands for File Transfer Protocol Server) is a software application which enables the transfer of files from one computer to another. It is a way to transfer files to any computer in the world that is connected to the internet.

FTP is a standard network protocol used for the transfer of files between a client and server on a computer network. FTP is a very well-established protocol, developed in the 1970s to allow two computers to transfer data over the internet. One computer acts as the server to store information and the other acts as the client to send or request files from the server.



FTP servers perform 2 basic tasks: "Put" and "Get." You can put files on the FTP Server or get files from the FTP Server. If security is not a concern, FTP Server software is an easy and inexpensive way to accomplish this.

Features of an FTP server:

- In order for the client to establish connection to the FTP server, the username and password are sent over using USER and PASS commands. Once accepted by the FTP server, an acknowledgement is sent across to the client and the session can start.
- In the case of an FTP connection, it is possible to resume the download if it was not successfully completed earlier. In other words, checkpoint restart support is provided.
- The FTP server allows the downloading and uploading files. There could be access restrictions as determined by the FTP server administrator for downloading different files and from different folders residing in the FTP server.
- The FTP server can provide connection to users without need of login credentials; however, the FTP server can authorize these to have only limited access.
- Files residing in FTP servers can be retrieved by common web browsers, but they may not be supporting protocol extensions like FTPS.
- FTP servers can provide anonymous access. This access allows users to download files from the servers anonymously, but prohibits uploading files to FTP servers.
- All file transfer protocol site addresses begin with ftp://.

Advantages of FTP:

- **Speed:** One of the biggest advantages of FTP is speed. The FTP is one of the fastest way to transfer the files from one computer to another computer.
- **Efficient:** It is more efficient as we do not need to complete all the operations to get the entire file.

- **Security:** To access the FTP server, we need to login with the username and password. Therefore, we can say that FTP is more secure.
- **Back & forth movement:** FTP allows us to transfer the files back and forth. Suppose you are a manager of the company, you send some information to all the employees, and they all send information back on the same server.

Disadvantages of FTP:

- The standard requirement of the industry is that all the FTP transmissions should be encrypted. However, not all the FTP providers are equal and not all the providers offer encryption. So, we will have to look out for the FTP providers that provides encryption.
- FTP serves two operations, i.e., to send and receive large files on a network. However, the size limit of the file is 2GB that can be sent. It also doesn't allow you to run simultaneous transfers to multiple receivers.
- Passwords and file contents are sent in clear text that allows unwanted eavesdropping. So, it is quite possible that attackers can carry out the brute force attack by trying to guess the FTP password.
- It is not compatible with every system.

FTP is a client-server protocol that relies on two communications channels between client and server: a *command channel* for controlling the conversation and a *data channel* for transmitting file content. Clients initiate conversations with servers by requesting to download a file. Using FTP, a client can upload, download, delete, rename, move and copy files on a server. A user typically needs to log on to the FTP server, although some servers make some or all of their content available without login, also known as anonymous FTP.

FTP sessions work in *passive* or *active* modes. In active mode, after a client initiates a session via a command channel request, the server initiates a data connection back to the client and begins transferring data. In passive mode, the server instead uses the command channel to send the client the information it needs to open a data channel. Because passive mode has the

client initiating all connections, it works well across firewalls and Network Address Translation (NAT) gateways.

Samba

Linux Samba Server is one of the powerful servers that helps you to share files and printers with Windows-based and other operating systems. It is an open-source implementation of the *Server Message Block/Common Internet File System* (SMB/CIFS) protocols. Samba was originally developed by Andrew Tridgell in 1991.

Samba is an open-source software suite that runs on Unix/Linux based platforms but is able to communicate with Windows clients like a native application. This means you can use a Linux server to provide file sharing, printing, and other services to other non-native Linux clients such as Microsoft Windows. So Samba is able to provide this service by employing the Common Internet File System (CIFS).

At the heart of this CIFS is the Server Message Block (SMB) protocol. Samba does this by performing these 4 key things –

- File & print services
- Authentication and Authorization
- Name resolution
- Service announcement (browsing)

Samba is often referred to as a *network file system* and Samba can be run on many different platforms including Linux, Unix, OpenVMS and operating systems other than Windows and allows the user to interact with a Windows client or server natively. It can basically be described as the Standard Windows interoperability suite of programs for Linux and Unix.

Samba is based on the common client/server protocol of Server Message Block (SMB) and Common Internet File System (CIFS). Using client software that also supports SMB/CIFS (for example, most Microsoft Windows products), an end user sends a series of client requests to the Samba server on another computer in order to open that computer's files,

access a shared printer, or access other resources. The Samba server on the other computer responds to each client request, either granting or denying access to its shared files and resources.The Samba SMB/CIFS client is called *smbclient*.

REVIEW QUESTIONS

Part A

1. What is the use of filter commands?
2. What is grep?
3. What is the difference between head and tail filters?
4. What is the use of sed command?
5. What is tr command?
6. What is pr command?
7. What is samba?
8. What is Apache?
9. What is DHCP?
10. What is DNS?
11. What is squid?

Part B

12. Differentiate grep and egrep.
13. What is Telnet? What are its advantages and disadvantages?
14. What is DNS Server?
15. Write short note on FTP?
16. Write a note on Apache server.

Part C

17. What is the use of filter commands? Explain the commands used for filtering?
18. Explain about different servers?

Software Lab IV (Core)

Sl.No	Topic and Details
1	Getting started –Commands
2	The Linux Architecture and command usage – Commands, General-purpose utilities
3	The File system –Commands
4	Process related commands
5	Handling ordinary files, Basic file attributes
6	The vi editor
7	Simple Filters, Filters using regular expressions-use of grep command
8	Introduction to shell concept and writing shell script, Essential Shell programming
9	User management, monitoring system performance, disk usage etc.

Basic Commands

1. **Command to show date and time in different formats**

   date

   date '+%d/%m/%Y'

   date '+%A,%B,%d'

   date '+%A,%d/%m/%y'

   date '+%I:%M:%s'

   date '+%I:%M %P'

2. **To get the calendar of current month and specified month**

   cal

   cal 7 1989

3. Perform the following operations

   a) Create three directories *Computer, Commerce,* and *Mathematics* under your home directory

   b) Move to the directory *Commerce*

   c) Create two directories *Finance* and *Marketing* under the directory *Commerce*

   d) Move to the directory *Mathematics* using single command

   e) Create directory *Linux* under the directory *Computer* without moving from *Mathematics*

   f) Move to *Linux* and check your current directory

   g) Create a file *command.txt* with a brief report on various commands

h) Print the content of *command.txt* file from your home directory
i) Make a copy of *command.txt* file and store it under the same directory where the file *command.txt* is stored
j) Use *ls* command to list the hierarchy in various formats
k) Remove the file *commnd.txt*
l) Remove all created directories recursively from your home director y

a) mkdir Computer Commerce Mathematics
b) cd Commerce
c) mkdir Finance Marketing
d) cd ../Mathematics
e) mkdir ../Computer/Linux
f) cd ../Computer/Linux
pwd
g) cat > Command.txt
h) cd ~
cat Computer/Linux/Command.txt
i) cat Computer/Linux/Command.txt > Computer/Linux/Copy.txt
j) ls
ls –R
ls –l
k) rm Computer/Linux/Command.txt
l) rm –r Computer Commerce Mathematics

4. Create a file *text.txt*, copy *text.txt* to *sample.txt* , Count the number of lines, words and characters, and sort the file and display the original and sorted file.

```
cat > text.txt
cat text.txt > sample.txt
wc –l text.txt
wc –w text.txt
wc –c text.txt
sort text.txt > textsort.txt
cat text.txt
cat textsort.txt
```

5. Create a file named sample.txt. Using the grep command perform the following:
a) Select the line from a file that has exactly 5 characters
b) Select the line from a file that has atleast 5 characters

c) Select the line from a file that start with uppercase letters
d) Select the line from a file that has one or more blank spaces

a) grep "^.....$" sample.txt
b) grep "^....." sample.txt
c) grep "[A-Z]" sample.txt
d) grep " " sample.txt

6. Command to change all lower case letters in a file to upper case letters
cat filelower | tr "[a-z]" "[A-Z]" > fileupper
cat filename | tr "[:lower:]" "[:upper:]"

7. Copy the first 5 and last 5 line of a file into another files.
head -5 sample > samplehead
tail -5 sample > sampletail

8. Create file student which contains rno, name, and mark. Display the total number of students, sort the file based on mark and show the student who got highest mark.
cat > student
wc –l student
sort –nrk 3 student > sortfile
head -1 sortfile

9. Replace all characters in a file to # except vowels and digits.
tr – c "[0-9 aeiou]" '#' < sample

10. Translate each word in the input file to separate line
tr " " "\n" < sample

Basic Shell Programs

1. **Shell program to read your name using read statement and print it.**
#!/bin/bash
echo –n "Enter your name : "
read name
echo -n "Your name is : " $name

2. **Program to read an integer and display it**
#!/bin/bash
echo "Enter a number"
read num
echo "Entered number is $num"

3. **Shell program to add two numbers**

```
#!/bin/bash
echo "Enter first number"
read num1
echo "Enter second number"
read num2
sum=`expr $num1 + $num2`
echo "Sum of the entered numbers: $sum"
```

4. **Program to perform basic arithmetic operations**

```
#!/bin/bash
echo "enter two numbers"
read a
read b
sum=`expr $a + $b`
diff=$((a-b))
pro=$(($a*$b))
let div=$a/$b
echo "Sum is $sum"
echo "Difference is $diff"
echo "Product is $pro"
echo "Quotient is $div"
```

5. **Program to accept the name of the file from the standard input and then performs the following operation: Enter 5 values in a file, Sort the file, and list unsorted and sorted file.**

```
#!/bin/bash
echo -n "Enter the file name: "
read fname
echo "Enter 5 values in the file $fname and press ^d at the end"
cat > $fname
sort $fname > sortfile
echo
echo "UNSORTED LIST"
cat $fname
echo
echo "SORTED LIST"
cat sortfile
```

Programs using Branching statements

1. Program to find the largest among two numbers

```
#!/bin/bash
echo "Enter two numbers"
read a b
if [ $a -gt $b ]
then
   echo "Numbers are $a and $b"
   echo "Largest number is $a"
elif [ $a -lt $b ]
then
   echo "Numbers are $a and $b"
   echo "Largest number is $b"
else
   echo "Numbers are $a and $b and are equal"
fi
```

2. Program to find the largest among three numbers.

```
#!/bin/bash
echo "Enter three numbers"
read a b c
if [ $a -gt $b ]
then
  if [ $a -gt $c ]
  then
        echo "Largest number is $a"
  else
        echo "largest number is $c"
  fi
else
  if [ $b -gt $c ]
  then
        echo "Largest number is $b"
  else
        echo "largest number is $c"
  fi
fi
```

3. **Program to read a student register number, name, and four subjects' marks and print whether he is passed or fail.**

```
#!/bin/bash
echo -n "Enter the register number:"
read reg
echo -n "Enter the name:"
read name
echo -n "Enter four marks one by one:"
read m1 m2 m3 m4
if [ $m1 -ge 40 -a $m2 -ge 40 -a $m3 -ge 40 -a $m4 -ge 40 ]
then
   total=`expr $m1 + $m2 + $m3 + $m4`
   echo "Totla mark is : $total"
   echo "Result : PASS"
else
   echo "Result : FAIL"
fi
```

4. **Program to test whether a file is existing. If the file exists check whether it is ordinary file, directory, readable, writable, and executable and empty.**

```
#!/bin/bash
echo -n "Enter the file name: "
read fname
echo
if [ -e $fname ]
then
   echo "File $fname EXISTS"
   if [ -f $fname ]
   then
         echo "$fname is an ordinary file"
   fi
   if [ -d $fname ]
   then
         echo "$fname is a directory"
   else
         echo "$fname is not a directory"
   fi
```

```
if [ -r $fname ]
then
        echo "$fname is readable"
else
        echo "$fname is not readable"
fi
if [ -w $fname ]
then
        echo "$fname is writeable"
else
        echo "$fname is not writeable"
fi
if [ -x $fname ]
then
        echo "$fname is executable"
else
        echo "$fname is not executable"
fi
if [ -s $fname ]
then
        echo "$fname is not empty"
else
        echo "$fname is empty"
fi
else
   echo "File $fname NOT EXISTS"
fi
```

5. Program for menu driven calculator

```
#!/bin/bash
echo "Enter two numbers"
read a b
echo "Enter an Operator"
read op
case $op in
+)
   res=`expr $a + $b`
```

```
    echo "Sum is $res"
    ;;
-)
    res='expr $a - $b'
    echo "Differene is $res"
    ;;
\*)
    res='expr $a \* $b'
    echo "Product is $res"
    ;;
/)
    res='expr $a / $b'
    echo "Quotient is $res"
    ;;
*)
    echo "Invalid Choice"
esac
```

6. **Program to check whether two strings are Equal or not, length is 0 or not and concatenating two strings.**

```
#!/bin/bash
echo -n "Enter first string: "
read first
echo -n "Enter second string: "
read second
echo -e "Enter your option \n 1: Equal or not \n 2: Find length \n 3: Concatenate"
read op
case $op in
1)
   if [ $first = $second ]
   then
            echo "Strings are equal"
   else
            echo "Strings are not equal"
   fi
   ;;
```

```
2)
   if [ -z $first ]
   then
            echo "Length of first string is zero"
   else
            echo "Length of first string is not zero"
   fi
   if [ -z $second ]
   then
            echo "Length of second string is zero"
   else
            echo "Length of second string is not zero"
   fi
   ;;
3)
   first+=$second
   echo "concatenated string is : $first"
   ;;
*)
   echo "invalid choice!"
   ;;
esac
```

Programs using loops

1. Program to print first n numbers using while loop

```
#!/bin/bash
#first n using while
echo "Enter a number"
read n
echo "First $n numbers are:"
i=1
while [ $i -le $n ]
do
   echo $i
   i=`expr $i + 1`
done
```

2. **Program to print the first n numbers using until loop**

```
#!/bin/bash
#first n using UNTIL
echo "Enter a number"
read n
echo "First $n numbers are:"
i=1
until [ $i -gt $n ]
do
   echo $i
   i=`expr $i + 1`
done
```

3. **Program to print first n numbers using for loop**

```
#!/bin/bash
#first n using UNTIL
echo "Enter a number"
read n
echo "First $n numbers are:"
for ((i=1;i<=n;i++))
do
   echo $i
done
```

4. **Program to enter n numbers and find the sum**

```
#!/bin/bash
sum=0
i=1
echo "Enter the number of elements"
read n
echo "Enter the elemets"
while [ $i -le $n ]
do
   read num
   sum=`expr $sum + $num`
   i=`expr $i + 1`
done
echo "Sum of above $n numbers are : $sum"
```

5. **Program to find the sum of digits of a number.**

```
#!/bin/bash
sum=0
echo "Enter a number"
read num
while [ $num -ne 0 ]
do
   r=`expr $num % 10`
   sum=`expr $sum + $r`
   num=`expr $num / 10`
done
echo "Sum of digits is : $sum"
```

6. **Program to find the sum of even digits and an average of odd digits of a given number.**

```
#!/bin/bash
sumeven=0
sumodd=0
echo -n "Enter a number: "
read num
while [ $num -ne 0 ]
do
   rem=`expr $num % 10`
   div=`expr $rem % 2`
   if [ $div -eq 0 ]
   then
         sumeven=`expr $sumeven + $rem`
   else
         sumodd=`expr $sumodd + $rem`
   fi
   num=`expr $num / 10`
done
echo "Sum of even numbers are: $sumeven"
echo "Average of odd numbers are: $sumodd"
```

7. **Program to find the reverse of a number.**

```
#!/bin/bash
rev=0
echo "Enter a number"
```

```
read num
while [ $num -ne 0 ]
do
   r=‘expr $num % 10‘
   rev=‘expr $rev \* 10 + $r‘
   num=‘expr $num / 10‘
done
echo “Reverse is : $rev”
```

8. Program to check whether the entered number is palindrome or not.

```
#!/bin/bash
sum=0
rev=0
echo “Enter a number”
read num
n=$num
while [ $num -ne 0 ]
do
   r=‘expr $num % 10‘
   rev=‘expr $rev \* 10 + $r‘
   num=‘expr $num / 10‘
done
if [ $n -eq $rev ]
then
   echo “Entered number is Palindrome”
else
   echo “Entered number is Not Palindrome”
fi
```

9. Program to find the factorial of a number.

```
#!/bin/bash
fact=1
echo “Enter a number”
read num
for((i=1;i<=num;i++))
do
   fact=‘expr $fact \* $i‘
done
echo “factorial of $num is $fact”
```

10. Program to check whether the entered number is prime or not.

```
#!/bin/bash
flag=0
echo "Enter a number"
read num
for((i=2;i<=num/2;i++))
do
   if [ $((num%i)) -eq 0 ]
   then
            flag=1
            break
   fi
done
if [ $flag -eq 0 ]
then
   echo "Entered number is Prime"
else
   echo "Entered number is Not Prime"
fi
```

11. Program to print all prime numbers between two ranges.

```
#!/bin/bash
echo "Enter the starting number"
read st
echo "Enter the ending number"
read end
echo "Prime numbers between $st and $end are:"
for((num=st;num<=end;num++))
do
flag=0
for((i=2;i<=num/2;i++))
do
   if [ $((num%i)) -eq 0 ]
   then
            flag=1
            break
   fi
```

```
done
if [ $flag -eq 0 ]
then
   echo $num
fi
done
```

12. **Program to check whether the entered number is Armstrong or not**

```
#!/bin/bash
a=0
sum=0
echo "Enter a number"
read num
temp=$num
while [ $temp -ne 0 ]
do
   a=`expr $temp % 10`
   sum=`expr $sum + $a \* $a \* $a`
   temp=`expr $temp / 10`
done
if [ $num -eq $sum ]
then
   echo "$num is an armstrong number"
else
   echo "$num is not an armstrong number"
fi
```

13. **Program to print the first n Fibonacci series**

```
#!/bin/bash
a=0
b=1
i=3
echo "Enter the value on n"
read n
echo "First $n Fibonacci Series"
if [ $n -eq 1 ]
then
   echo $a
```

```
elif [ $n -eq 2 ]
then
   echo $a
   echo $b
else
   echo $a
   echo $b
   while [ $i -le $n ]
   do
         c=‘expr $a + $b‘
         echo $c
         a=$b
         b=$c
         i=‘expr $i + 1‘
   done
fi
```

14. Program to find the GCD of two numbers

```
#!/bin/bash
echo “Enter two numbers”
read a b
if [ $a -gt $b ]
then
   m=$a
else
   m=$b
fi
for ((i=1;i<=m;i++))
do
   x=‘expr $a % $i‘
   y=‘expr $b % $i‘
   if [ $x -eq 0 -a $y -eq 0 ]
   then
         gcd=$i
   fi
done
echo “GCD of $a and $b is $gcd”
```

15. **Program to find the value of nCr**

```
#!/bin/bash
echo "Enter the values of n and r"
read n r
nfact=1
rfact=1
nrfact=1
#Finding n!
for((i=1;i<=n;i++))
do
   nfact=`expr $nfact \* $i`
done
#Finding r!
for((i=1;i<=r;i++))
do
   rfact=`expr $rfact \* $i`
done
#Finding (n-r)!
nr=`expr $n - $r`
for((i=1;i<=nr;i++))
do
   nrfact=`expr $nrfact \* $i`
done
i=`expr $rfact \* $nrfact`
res=`expr $nfact / $i`
echo "Result is : $res"
```

Programs using Array

1. **Program to read n numbers in an array**

```
#!/bin/bash
declare -a ar
echo "Enter the number of elements"
read n
echo "Enter the elements"
for((i=0;i<n;i++))
do
   read a[i]
```

```
done
echo "Array elemts are:"
echo ${a[@]}
```

2. Program to find the sum of elements in array

```
#!/bin/bash
declare -a ar
sum=0
echo "Enter the number of elements"
read n
echo "Enter the elements"
for((i=0;i<n;i++))
do
   read a[i]
   sum=`expr $sum + ${a[i]}`
done
echo "Array elemts are:"
echo ${a[@]}
echo "Sum is : $sum"
```

3. Program to sort elements in array

```
#!/bin/bash
echo "Enter the number of elements"
read n
echo "Enter the numbers"
for ((i=0;i<n;i++))
do
   read a[i]
done
echo "Numbers before sorting:"
echo ${a[@]}
for((i=0;i<n;i++))
do
   for((j=i+1;j<n;j++))
   do
        if [ ${a[i]} -gt ${a[j]} ]
        then
             t=${a[i]}
             a[i]=${a[j]}
             a[j]=$t
```

BCA Degree (C.B.C.S.S) Model Examination

Fourth Semester

Linux Administration

Time: 3hrs Total Marks: 80

Part A

Answer any **Ten** *questions. Each question carries* **2** *marks.*

1. Define Shell.
2. What is kernel?
3. How to create and delete directory in Linux.
4. What is the use of *sort* command?
5. What is process?
6. How to connect a process with pipe.
7. What is the use of *nohup* command?
8. What is a shell variable?
9. What are the different types of users in Linux?
10. What is the use of *su* command?
11. What are the different methods to temporary disable a user account?
12. What is the use of *grep* command? (10 × 2= 20)

Part B

Answer any **Six** *Questions. Each question carries* **5** *marks.*

13. Explain Linux system architecture.
14. How to compare files using *cmp* and *diff* commands.
15. Define process and what are the different types of processes?
16. Explain how to create and edit files using *vi* editor.
17. Write note on creating and mounting file system.
18. Explain how to insert and remove packages with *rpm* command.
19. What are the different configuration and log files in Linux OS.
20. Explain how to change permission and ownership of file using *chmod*?
21. What is filtering and write any four filtering commands. (6×5=30)

Part C

Answer any **Two** *Questions. Each question carries* **15** *marks.*

22. a) Explain Linux file system with neat diagram. (10 marks)
    b) Explain how to create and view file using *cat* command. (5 marks)
23. a) Explain how to redirect an input output. (10 marks)
    b) How to schedule processes using *at* and *batch* command. (5 marks)
24. Explain different branching and looping statements in shell programming with example.
25. Explain adding, deleting and modifying user and group accounts with example. (2 × 15 =30)

BCA Degree (CBCS) Examination, May 2019

Fourth Semester

Core Course- Linux Administration

2017 Admission Onwards

Time: 3 Hours

Maximum Marks: 80

Part A

Answer any **ten** *questions. Each question carries* **2** *marks.*

1. How Linux differs from other operating systems like MS Windows?
2. What is super block?
3. What is file command?
4. What is meant by background processing in Linux?
5. Which command is used to count the total number of words, lines and characters in a file?
6. What are the different modes of vi editor?
7. Give syntax of case statement.
8. Write commands to display first command line argument and total number of command line arguments in a shell script.
9. Define groupadd command in Linux.
10. Define the term super user.
11. What is the difference between head and tail filters?
12. What is samba?

(10 × 2 = 20)

Part B

Answer any **Six** *Questions. Each question carries* **5** *marks.*

13. Explain Linux file system in detail.
14. Write short note on Linux standard directories.
15. What is Linux redirection? Explain the different types of redirection with suitable examples.
16. List out the usages of find command in locating files and directories.
17. Describe the use of conditional statement in shell scripts with suitable example.
18. Explain different types of variables in shell script.
19. Discuss how a system administrator can manage its user account.
20. What is DNS Server?
21. What is Telnet? What all are the advantages and disadvantages of using Telent.

(6 × 5 = 30)

Part C

Answer any **Two** *Questions. Each question carries* **15** *marks.*

22. Explain the following Linux concepts:
    a. Connecting process using pipes
    b. Explain different mathematical commands.
23. What is shell in Linux? Compare the different shells available in Linux.
24. Explain the common administrative tasks in Linux.
25. Write a note on Apache Server.

(2 × 15 = 30)

———