

LINUX COMMANDS

By

Muhammad Wasim

UNIX Commands

- ❏ A command is a program which interacts with the kernel to provide the environment and perform the functions called for by the user.
- ❏ A command can be: a built-in shell command; an executable shell file, known as a shell script; or a source compiled, object code file.
- ❏ The shell is a command line interpreter. The user interacts with the kernel through the shell. You can write ASCII (text) scripts to be acted upon by a shell.

UNIX Shell

- ❏ The shell sits between you and the operating system, acting as a command interpreter.
- ❏ It reads your terminal input and translates the commands into actions taken by the system. The shell is analogous to `command.com` in DOS.
- ❏ When you log into the system you are given a default shell.
- ❏ When the shell starts up it reads its startup files and may set environment variables, command search paths, and command aliases, and executes any commands specified in these files.

UNIX Shell



- ❑ The original shell was the Bourne shell, sh.
- ❑ Every Unix platform will either have the Bourne shell, or a Bourne compatible shell available.
- ❑ The default prompt for the Bourne shell is \$ (or #, for the root user).
- ❑ Another popular shell is C Shell. The default prompt for the C shell is %.

UNIX Shell

- Numerous other shells are available from the network. Almost all of them are based on either sh or csh with extensions to provide job control to sh, allow in-line editing of commands, page through previously executed commands, provide command name completion and custom prompt, etc.
- Some of the more well known of these may be on your favorite Unix system: the Korn shell, ksh, by David Korn and the Bourne Again Shell, bash, from the Free Software Foundations GNU project, both based on sh, the T-C shell, tcsh, and the extended C shell, cshe, both based on csh.

Shell Programming

- ❏ You can write shell programs by creating scripts containing a series of shell commands.
- ❏ The first line of the script should start with `#!` which indicates to the kernel that the script is directly executable.
- ❏ You immediately follow this with the name of the shell, or program (spaces are allowed), to execute, using the full path name. So to set up a Bourne shell script the first line would be: `#! /bin/sh`

Shell Programming

- ❏ The first line is followed by commands
- ❏ Within the scripts # indicates a comment from that point until the end of the line, with #! being a special case if found as the first characters of the file.

```
#!/bin/bash
```

```
cd /tmp
```

```
mkdir t
```

- ❏ You also need to specify that the script is executable by setting the proper bits on the file with chmod, e.g.:

```
$ chmod +x shell_script
```

LINUX COMMANDS



- ▣ **File Management and Viewing**
- ▣ **Filesystem Mangement**
- ▣ **Help, Job and Process Management**
- ▣ **Network Management**
- ▣ **System Management**
- ▣ **User Management**
- ▣ **Printing and Programming**
- ▣ **Document Preparation**
- ▣ **Miscellaneous**


Command Structure

- ❏ **Command <Options> <Arguments>**
- ❏ **Multiple commands separated by ; can be executed one after the other**

Help Facilities for Commands

- ❏ To understand the working of the command and possible options use (man command)
- ❏ Using the GNU Info System (info, info command)
- ❏ Listing a Description of a Program (whatis command)
- ❏ Many tools have a long-style option, '--help', that outputs usage information about the tool, including the options and arguments the tool takes. Ex:
whoami --help

Pipes

- 
-
- An important early development in Unix was the invention of "pipes," a way to pass the output of one tool to the input of another.

eg. `$ who | wc -l`

By combining these two tools, giving the `wc` command the output of `who`, you can build a new command to list the number of users currently on the system

Linux File Management and Viewing

File and Directory management

 **cd** Change the current directory. With no arguments "cd" changes to the users home directory. (cd <directory path>)

 **chmod** Change the file permissions.

Ex: chmod 751 myfile : change the file permissions to rwx for owner, rx for group and x for others

Ex: chmod go+=r myfile : Add read permission for the group and others (character meanings u-user, g-group, o-other, + add permission,-remove,r-read,w-write,x-exe)

Ex: chmod +s myfile - Setuid bit on the file which allows the program to run with user or group privileges of the file.

Linux File Management and Viewing

There are three such special permissions within Linux. They are:

- ❏ **setuid** — used only for applications, this permission indicates that the application is to run as the owner of the file and not as the user executing the application. It is indicated by the character *s* in place of the *x* in the owner category. If the owner of the file does not have execute permissions, the *S* is capitalized to reflect this fact.
- ❏ **setgid** — used primarily for applications, this permission indicates that the application is to run as the group owning the file and not as the group of the user executing the application. The *setgid* permission is indicated by the character *s* in place of the *x* in the group category. If the group owner of the file or directory does not have execute permissions, the *S* is capitalized to reflect this fact.
- ❏ **sticky bit** — used primarily on directories, this bit dictates that a file created in the directory can be removed only by the user that created the file. It is indicated by the character *t* in place of the *x* in the everyone category. If the everyone category does not have execute permissions, the *T* is capitalized to reflect this fact.

Linux File Management and Viewing

 **chown** Change owner.

Ex: `chown <owner1> <filename>` : Change ownership of a file to owner1.

 **chgrp** Change group.

Ex: `chgrp <group1> <filename>` : Change group of a file to group1.

 **cp** Copy a file from one location to another.

Ex: `cp file1 file2` : Copy file1 to file2

Ex: `cp -R dir1 dir2` : Copy dir1 to dir2

 **md5sum** Prints the MD5 Checksum

Linux File Management and Viewing

 **ls** List contents of a directory.

Ex: `ls`, `ls -l`, `ls -al`, `ls -ld`, `ls -R`

(`-rwxrwxr-x 1 juan juan 0 Sep 26 12:25 foo`)

|more will list page wise

 **mkdir** Make a directory.

Ex: `mkdir <directory name>` : Makes a directory

Ex `mkdir -p /www/cache/var/log` will create all the directories starting from `www`.

 **mv** Move or rename a file or directory.

Ex: `mv <source> <destination>`

Linux File Management and Viewing

 **find** Find files (find <start directory> -name <file name> -print)

Ex: *find /home -name readme -print*

(Search for readme starting at home and output full path.)

"/home" = Search starting at the home directory and proceed through all its subdirectories

"-name readme" = Search for a file named readme

"-print" = Output the full path to that file

 **locate** File locating program that uses the slocate database.

Ex: locate -u to create the database,

locate <file/directory> to find file/directory

Linux File Management and Viewing

- ❏ **pwd** Print or list the present working directory with full path.
- ❏ **rm** Delete files (Remove files). (rm -rf <directory/file>)
- ❏ **rmdir** Remove a directory. The directory must be empty. (rmdir <directory>)
- ❏ **touch** Change file timestamps to the current time. Make the file if it doesn't exist. (touch <filename>)
- ❏ **whereis** Locate the binary and man page files for a command. (whereis <program/command>)
- ❏ **which** Show full path of commands where given commands reside. (which <command>)

Linux File Management and Viewing

File viewing and editing

📄 **emacs** Full screen editor.

📄 **pico** Simple text editor.

📄 **vi** Editor with a command mode and text mode. Starts in command mode.

📄 **gedit** GUI Text Editor

📄 **tail** Look at the last 10 lines of a file.

Ex: `tail -f <filename>` ,

Ex: `tail -100 <filename>`

📄 **head** Look at the first 10 lines of a file. (`head <filename>`)

Linux File Management and Viewing

File compression, backing up and restoring

- ▣ **compress** Compress data.
- ▣ **uncompress** Expand data.
- ▣ **cpio** Can store files on tapes. to/from archives.
- ▣ **gzip** - zip a file to a gz file.
- ▣ **gunzip** - unzip a gz file.
- ▣ **tar** Archives files and directories. Can store files and directories on tapes.

Ex: `tar -zcvf <destination> <files/directories>` - Archive copy groups of files. `tar -zxvf <compressed file>` to uncompress

- ▣ **zip** – Compresses a file to a .zip file.
- ▣ **unzip** – Uncompresses a file with .zip extension.

Linux File Management and Viewing

 **cat** View a file

Ex: cat filename

 **cmp** Compare two files.

 **cut** Remove sections from each line of files.

 **diff** Show the differences between files.

Ex: diff file1 file2 : Find differences between file1 & file2.

 **echo** Display a line of text.

Linux File Management and Viewing

❏ **grep** List all files with the specified expression.
(*grep pattern <filename/directorypath>*)

Ex: `ls -l | grep sidbi` : List all lines with a sidbi in them.

Ex: `grep " R "` : Search for R with a space on each side

❏ **sleep** Delay for a specified amount of time.

❏ **sort** Sort a file alphabetically.

❏ **uniq** Remove duplicate lines from a sorted file.

❏ **wc** Count lines, words, characters in a file. (`wc -c/w/l <filename>`).