

Experiment Number: 1

TITLE: Study of UNIX Shell Commands

OBJECTIVE:

- 1.To study Unix basic.
- 2.To study Unix basic shell commands.

UNIX Basic:

UNIX is an operating system which was first developed in the 1960s, and has been under constant development ever since. By operating system, we mean the suite of programs which make the computer work. It is a stable, multi-user, multi-tasking system for servers, desktops and laptops. UNIX systems also have a graphical user interface (GUI) similar to Microsoft Windows which provides an easy to use environment. There are many different versions of UNIX, although they share common similarities. The most popular varieties of UNIX are Sun Solaris, GNU/Linux, and MacOS X. The UNIX operating system is made up of three parts; the kernel, the shell and the programs.

The kernel

The kernel is the core of the UNIX operating system. Basically, the kernel is a large program that is loaded into memory when the machine is turned on, and it controls the allocation of hardware resources from that point forward. The kernel knows what hardware resources are available (like the processor(s), the on-board memory, the disk drives, network interfaces, etc.), and it has the necessary programs to talk to all the devices connected to it. As an illustration of the way that the shell and the kernel work together, suppose a user types `rm myfile` (which has the effect of removing the file `myfile`). The shell searches the file store for the file containing the program `rm`, and then requests the kernel, through system calls, to execute the program `rm` on `myfile`. When the process `rm myfile` has finished running, the shell then returns the UNIX prompt to the user, indicating that it is waiting for further commands.

The shell

The shell acts as an interface between the user and the kernel. The shell is a command line interpreter (CLI). It interprets the commands the user types in and arranges for them to be carried out. The commands are themselves programs: when they terminate, the shell gives the user another prompt. The adept user can customize his /her own shell and users can use different shells on the same machine. UNIX system offers variety of shells like 1) Bourne shell 2) c shell 3) Korn shell 4) Bash shell (very powerful & recommended for use, Linux default shell)

History - The shell keeps a list of the commands you have typed in. If you need to repeat a command, use the cursor keys to scroll up and down the list or type `history` for a list of previous commands.

The UNIX file system

All the stored information on a UNIX computer is kept in a file system. Any time we interact with the UNIX shell, the shell considers us to be located somewhere within a file system. The place in the file system tree where we are located is called the current working directory.

The UNIX file system is hierarchical (resembling a tree structure). The tree is anchored at a place called the root, designated by a slash "/". Every item in the UNIX file system tree is either a file, or a directory. A directory is like a file folder. A directory can contain files, and other directories. A directory contained within another is called the child of the other. A directory in the file system tree may have many children, but it can only have one parent. A file can hold information, but cannot contain other files, or directories.

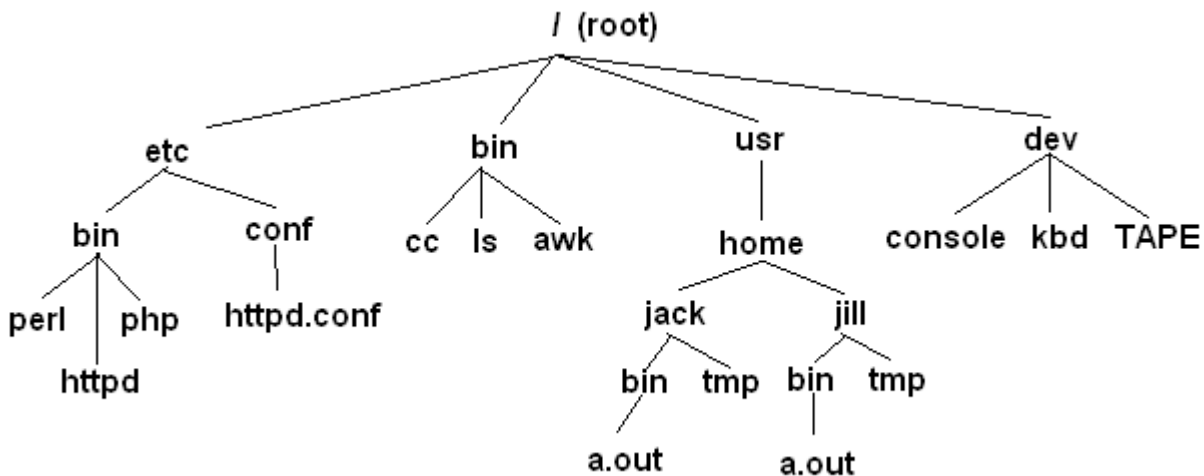


Fig. UNIX/Linux directory structure.

To describe a specific location in the file system hierarchy, you must specify a "path." The path to a location can be defined as an absolute path from the root anchor point, or as a relative path, starting from the current location. When specifying a path, you simply trace a route through the file system tree, listing the sequence of directories you pass through as you go from one point to another. Each directory listed in the sequence is separated by a slash. UNIX provides the shorthand notation of "." to refer to the current location, and ".." to refer to the parent directory.

Linux

Linux is not UNIX, as UNIX is a copyrighted piece of software that demands license fees when any part of its source code is used. Linux was written from scratch to avoid license fees entirely, although the operation of the Linux operating system is based entirely on UNIX. It shares UNIX's command set and look-and-feel, so if you know either UNIX or Linux, you know the other, too. Linux is a freely distributable version of UNIX developed primarily by Linus Torvalds at the University of Helsinki in Finland.

Text editors in Linux

Linux is just as well suited for word processing as any other operating system. There are several excellent word processing programs for Linux like AbiWord, KWord, part of the KOffice suite and the OpenOffice.org as well as StarOfficesuite's word processor.

Why use a text editor?

A text editor is just like a word processor without a lot of features. All operating systems come with a basic text editor. Linux comes with several. The main use of a text editor is for writing something in plain text with no formatting so that another program can read it. Based on the information it gets from that file, the program will run one way or another.

The text editor "vi"

The most popular text editor for Linux is called 'vi'. This is a program that comes from UNIX. There is a more recent version called 'vim' which means 'vi improved'. The problem with 'vi' or 'vim' is that a lot of people don't like it. You have to remember a lot of key combinations to do stuff that other text editors will do for you more easily.

Working with 'vi'

Let's make a text file. Type:

```
vi osdl
```

We'll see a line of tildes down the left side and the name 'osdl' at the bottom and [new file].

To write something, we have to press ESC and the 'i' key (i for insert). We can always erase our mistakes with the backspace key.

To save this file, we would press ESC then the colon key ':' then 'w' (write)

To save the file and quit vi, you would press ESC, ESC the colon key ':' then `wq` (write, quit)

To quit without saving, press ESC, ':' then 'q'. vi may protest if we've written something and we don't want to save it. If we press ESC ':' 'q!' with an exclamation point, vi will accept it and not save our changes.

'pico' and 'joe'

Actually, these are two other text editors like 'vi' in that we use them to create and edit non-formatted text, but they are a little more user-friendly. We won't see the tildes like vi. It looks a little friendlier. The majority of joe's commands are based on the CTRL-K keys and a third key. The most important of these is CTRL-K-H which gets you 'help'. Help shows us the key combinations to use with 'joe'.

Shells in Linux

Frequently Used Shell Commands

1) **cd**

'cd' means 'change directory'.

Typing: `cd [/directory name]` will get us into one of the main directories.

Typing `cd ..` will get us out of it. (move to parent directory.)

Typing `cd` without the / and a sub-directory name will get into that subdirectory.

If we type just: `cd` we'll go back to our home directory.

2) **ls**

Typing 'ls' will list the contents of a directory with just information about file names.

The syntax for the ls command is:

```
ls [options] [names]
```

Options:

- a Displays all files.
- b Displays nonprinting characters in octal.
- c Displays files by file timestamp.
- C Displays files in a columnar format (default)
- d Displays only directories.
- l Displays the long format listing.
- L Displays the file or directory referenced by a symbolic link.
- m Displays the names as a comma-separated list.
- n Displays the long format listing, with GID and UID numbers.
- o Displays the long format listing, but excludes group name.
- p Displays directories with /
- q Displays all nonprinting characters as ?
- r Displays files in reverse order.
- R Displays subdirectories as well.

- t Displays newest files first. (based on timestamp)
- u Displays files by the file access time.
- x Displays files as rows across the screen.
- 1 Displays each entry on a line.

Example:

```
ls -la
```

3) **cp**

'cp' is used for copying files from one place to another, or for making a duplicate of one file under a different name.

Example: `cp first.txt second.txt`

The `first.txt` file will be copied into `second.txt`

4) **mv**

'mv' is used for moving files from one place to another. It cuts the file from one place and pastes it to another.

Options:

- f Forces the move.
 - i Prompt for a confirmation before overwriting any files.
- syntax: `mv [options] sources target`

Examples:

```
mv -f sitrc /usr
```

It will move the file `sitrc.txt` to the directory `usr`

5) **mkdir**

This command is used for making or creating directories. The syntax for the `mkdir` command is:

```
mkdir [options] directories
```

options:

- m Sets the access mode for the new directory.
- p If the parent directories don't exist, this command creates them.

Examples:

```
mkdir SITRC
```

6) **rmdir**

This is the opposite of 'mkdir'- which is used to delete the directories. It should be pointed out that in order to use it, the directory has to be empty.

So, we have to use your 'rm' command for removing files in the directory.

7) **rm**

This command is used for removing or deleting files.

Syntax:

```
rm [options] filename
```

Options:

- r removes directories, removing the contents recursively beforehand (so as not to leave files without a directory to reside in) ("recursive")
- i asks for every deletion to be confirmed ("interactive")
- f ignores non-existent files and overrides any confirmation prompts ("force"), although it will not remove files from a directory if the directory is write protected.

8) **more** and **less**

'more' is a command that we can use to read, for example, what's written in a file. We would type 'more xyz' to see the file completely. Then, we can press the 'q' key to stop viewing the file. We can scroll back up to see the whole text if we want.

9) Plumbing with "**pipes**" in Linux

To use the pipe command, we don't type: pipe. We press the '|' key. This is used with other commands. That means that the pipe will separate two commands so that they will be done one after the other. Let's try some plumbing.

For example, list the numbers of users currently login in the system and then sort it.

```
ls who | sort
```

10) **grep**

The grep command allows us to search one file or multiple files for lines that contain a pattern. Exit status is 0 if matches were found, 1 if no matches were found, and 2 if errors occurred.

The syntax for the grep command is:

```
grep [options] pattern [files]
```

options:

- b Display the block number at the beginning of each line.
- c Display the number of matched lines.
- h Display the matched lines, but do not display the filenames.
- i Ignore case sensitivity.
- l Display the filenames, but do not display the matched lines.
- n Display the matched lines and their line numbers.
- s Silent mode.

- v Display all lines that do NOT match.
- w Match whole word.

Examples:

```
grep -c tech file1
```

11) **who**

This is used to find out who's working on our system. As we now know, Linux is a multi-user system. Even if we're using one computer at our home, we may be working as more than one person. For example, if we logged in as 'root' but are working as 'nitin'. We may see something like this:

```
root      tty1 May 20 09:48
nitin     tty2 May 20 10:05
```

This is just Linux's way of saying that 'root' started working on terminal 1 on May 20 at 9:48 in the morning and *nitin* started working on terminal 2 at 10:05. This is mainly used in networked situations so the system administrator knows who's working.

12) **whoami**

It is a little program that tells us who we are, just in case we didn't know already. Actually it tells us who we are in terms of how Linux understands who you are, that is to say, our user name.

Syntax: `whoami`

13) **pwd**

(print working directory)

The `pwd` command displays the full pathname of the current directory. The syntax for the `pwd` command is:

```
pwd
```

14) **cat**

The `cat` command reads one or more files and prints them to standard output. The operator `>` can be used to combine multiple files into one. The operator `>>` can be used to append to an existing file.

The syntax for the `cat` command is:

```
cat [options] [files]
```

options:

- e \$ is printed at the end of each line. This option must be used with -v.
- s Suppress messages pertaining to files that do not exist.
- t Each tab will display as ^I and each form feed will display as ^L. This option must be used with -v.
- u Output is printed as unbuffered.
- v Display control characters and nonprinting characters

Examples:

```
cat file1
```

```
cat file1 file2 > all
cat file1 >> file2
```

15) wc

This command will give us the number of lines, words and letters (characters) in a file and in that order.

Usage:

```
wc -l <filename> print the line count
wc -c <filename> print the byte count
wc -m <filename> print the character count
wc -L <filename> print the length of longest line
wc -w <filename> print the word count
```

16) ps

It gives us a list of the processes running on our system. Just typing ps will give us the processes we're running as a user.

17) chmod

The chmod command changes the access mode of one file or multiple files. The syntax for the chmod command is:

```
chmod [option] mode files
```

options:

```
-R    Descend directory arguments recursively while setting modes.
-f    Suppress error messages if command fails.
```

mode:

Who u=user, g=group, o=other, a=all (default)

Opcode + means add permission

 — means remove permission

 = means assign permission and remove the permission of unspecified fields

Permission r=Read, w=write, x=Execute

Examples:

```
chmod ug+rw mydir
chmod a-w myfile
```

Conclusion: The shell acts as an interface between the user and the kernel. Thus shell use to execute commands.

References:

1. Unix Concepts and Applications By Sumitabha Das, Tata McGraw Hill
2. UNIX Shell Programming by Yashwant Kanetkar, BPB Publications.

Tushar B Kute
(Subject Teacher)