Linux Commands & Editors

Module-2
OSSD (21B12CS320)
B.Tech.(CSE-6th Sem)

JIIT, Noida

Where are these commands located?

- Directories:
  - /bin
  - /sbin
  - /usr/bin
  - /usr/sbin

Command Structure

- **command [-option(s)] [argument(s)]**
  - the command;
  - any options required by the command
  - the command's arguments (if required).
  - Options MUST come after the command and before any command arguments
  - Options SHOULD NOT appear after the main argument(s)
  - However, some options can have their own arguments!

More About Options

- All options are preceded by a hyphen (-)
- Options without arguments may be grouped after the hyphen
- The first option argument, following an option, must be preceded by white space. For example -o sfile is valid but -osfile is illegal.
- Option arguments are not optional

More About Options

- All options must precede other arguments on the command line
- A double hyphen -- may be used to indicate the end of the option list
- The order of the options are order independent
- The order of arguments may be important
- A single hyphen - is used to mean standard input

man

- Manual Pages
- Contains information about almost everything
  - Other Commands
  - System Calls
  - C Library Functions

Syntax:

```
man <command name>
```

Example:

```
$ man ls
```

which

- Displays a path name of a command
- Searches a path environmental variable for the command and displays the absolute path

Syntax:

```
which <command name/app>
```

Example:

```
$ which sh
```

(shows which sh is actually in use)

whereis

- Display all locations of a command (or some other binary, man page, or a source file).
- Searches all directories to find commands that match the argument

Syntax:

```
whereis <command name>
```

Example:

```
$ whereis sh
```

passwd

- Change your login password.

  Syntax:

  ```
  passwd
  passwd <username>
  ```

  Example:

  ```
  $ passwd user1
  ```

date

- Displays dates in various formats

  Example :

```
$ date              (in IST)
$ date -u           ( in GMT )
```

clear

- To clear the screen

  Syntax:

  ```
  $ clear  or ctrl + L
  ```

alias

- Defines a new name for a command

  Syntax:

  ```
  alias <newcommand>='<oldcommand>'
  ```

  Example:

  ```
  $ alias dt='date'
  $ dt
  ```

history

- Display a history of recently used commands

  Syntax:

  ```
  history <option>
  ```

  Example:

  ```
  $history 10
  ```

exit

- Exit from your login session.

- **Example :**
- $ exit

shutdown

- Causes system to shutdown or reboot
- May require super-user privileges

- **Example:**

```
$ shutdown -h now          ( stop )
$ shutdown -r now     ( reboot )
```

File Management Commands

ls

- Lists directory contents

  Syntax:

  ```
  ls <option>
  ```

  Examples:

  ```
  $ ls          (lists all files except those starting with a “.”)
  $ ls -a
  $ ls -l
  $ ls -al
  ```

cat

- Takes a copy of a file and sends it to the standard output

  Syntax:

  ```
  cat <filename>
  ```

  Example:

  ```
  $ cat link.txt
  ```

more

- Display contents of large files page by page or scroll line by line up and down.
- **Syntax:**

```
more <option> <filename>
```

Examples:

```
$ more   a.txt          (press enter to see next page content)
$ more -s a.txt          (squeeze multiple space line into single)
```

cp

- Copies files/directories

  Syntax:

  ```
  $ cp <options><source> <destination>
  ```

  Example:

  ```
  $ cp a.txt b.txt
  ```

(Useful option: `-i` to prevent overwriting existing files and prompt the user to confirm)

mv

- Moves or renames files/directories

  Syntax:

  ```
  % mv <source> <destination>
    (The <source> file gets removed)
  ```

  Example:

  ```
  % mv b.txt d.txt
  ```

rm

- Removes file(s) and/or directories.

Syntax:

```
$ rm <options> <filename>
```

Example:

```
$ rm d.txt
```

diff

- Compares file and, shows where they differ.

  Syntax:

  ```
  $ diff <filename1> <filename2>
  ```

  Example:

  ```
  $ diff  a.txt   b.txt
  ```

find

- Searching a file in a directory tree

  Syntax:

  ```
  $find <option> <filename>
  ```

  Example:

  ```
  $ find -name "a.txt"
  ```

cd

- Changes your current directory to a new one.

  Syntax:

  ```
  cd <dirname>
  ```

  Example:

  ```
  $ cd /usr/home/example
  ```

mkdir

- Creates a directory

Syntax:

```
$ mkdir <dirname>
```

Example:

```
$ mkdir etcs lab
```

rmdir

- Removes a directory

Syntax:

```
$ rmdir <dirname> (empty)
$ rm -r <dirname>
```

Example:

```
$ rm -r etcs
```

wc

- Tells you how many lines, words, and characters there are in a file

Syntax:

```
$ wc filename
```

Example:

```
$ wc a.txt                    (line words char)
```

pwd

- Displays the present working directory, i.e. your current directory.

  Example:

  ```
  $ pwd
  ```

chown

- To change the owner and owning group of files

  Syntax:

  - chown <owner/user> <filename>
  - chown <owner-user:owner-group > <filename>

  Example:

  ```
  $ chown abc link.txt
  ```

chmod

- To change permissions of files or directories

Syntax:

```
$ chmod <option> <permission> <filename>
```

Example:

```
$ chmod 777 link.txt
```

grep

- To print lines of input matching a specified pattern

  Syntax:

  ```
  $ grep <option> <pattern> <file>
  ```

  Example:

  ```
  $ grep include link.txt
  ```

User/Group Management Commands

useradd

- To add a new user

  Syntax:

  ```
  useradd <username>
  ```

  Example:

  ```
  $ useradd xyz
  ```

userdel

- To delete a user

  Syntax:

```
$ userdel <username>
```

Some Other Commands

zip

- Compresses files, so that they take up much less space

Syntax:

```
$ zip -r <filenames.zip> <file1> <file2>
```

Example:

```
$ zip -r                foo.zip a.txt b.txt etcs
```

unzip

- Uncompress the files compressed by gzip

Syntax:

```
% unzip <options> filename
```

(zipfile name without extension)

Example:

```
% unzip foo
```

who

- Tells you who's logged on, and where they're coming from.

  Example:

  ```
  $ who
  ```

whoami

- Displays the same information as who, but only for the terminal session from where the command was issued.

  Example:

  ```
  $ whoami
  ```

last

- Tells you when the user last logged on and off and from where.

  Syntax:

  ```
  $ last -1 username
  ```

(Without any options, **last** will give you a list of everyone's logins)

echo

- Displays a line of text

  Syntax:

  ```
  echo <option> <string>
  ```

  Example:

  ```
  $ echo Hello, World!
  $ x=10
  $ echo The value of x is $x.
  ```

ps

- Displays information about a selection of the active processes.
- Contains lots of information about them including the process ID
- **Syntax:**

```
$ ps <options>
```

Example:

```
$ ps -a
```

ifconfig

To see the IP Address

Syntax:

```
$ ifconfig <option>
```

Example:

```
$ ifconfig -a
```

telnet

- To connect to a remote host

  Syntax:

  ```
  $ telnet <hostname/ipaddress>
  ```

  Example:

  ```
  $ telnet myhost.com
  ```

ftp

- To download/upload files from/to a remote host which is set up as an ftp-server
- **Syntax:**

```
$ ftp <hostname/ipaddress>
```

Example:

```
$ ftp 172.31.128.116
```

Important Commands for OpenSource Development Support

System Monitoring

- Display and manage the running processes
  - **$ top**
- Display processor related statistics
  - **$ mpstat 1**
- Display virtual memory statistics
  - **$ vmstat 1**
- Display disk I/O statistics
  - **$ iostat 1**

System Monitoring

- List all open files on the system
  - **$ lsof**
  - **$ lsof -u USER** [file open by specific user]
- Display disk space occupied by current directory
  - **$ du -sh**
- Execute periodically:
  - **$ watch**

Files Related

- Creating empty files:
  - **$ touch**
- List directory tree
  - **$ tree**
- Create symbolic link (shortcut/pointer)
  - **$ ln -s file1 file1-link**
- Display first few/ last few lines of a file
  - **$ head -n <num> file**
  - **$ tail -n <num> file**

Process Related

- Display your currently running processes
  - **$ ps**
- Display every process on the system.
  - **$ ps auxf**
- Display process information for the process name
  - **$ ps uf -C processname**
- Display interactive real-time view of running processes
  - **$ top**
  - **$ htop**

Process Related

- Look-up process ID based on a name
  - **$ pgrep <processname>**
- Kill a process with a given process ID. By default TERM signal is sent
  - **$ kill PID**
- Kill a process based on a name
  - **$ kill <processname>**
- Run a command as a background job
  - **$ <command> &**

Process Related

- List background jobs
  - **$ jobs**
- Display stopped or background jobs
  - **$ bg**
- Brings the most recent background job to the foreground
  - **$ fg**

Download files from a remote HTTP server

- **wget**
- **curl**

Text/File Search

- Search for a pattern in a text file
  - **$ grep pattern file**
- Find files within a directory with a matching filename
  - **$ find directory -iname 'pattern'**
- Find files based on filesize
  - **$ find <directory> -size <+1M>**
  - **$ find <directory> -size <+1M>**

Redirection

- Redirect normal output (stdout) from a command to a file
  - **$ echo "hello" > a.txt**
- Append normal output (stdout) from a command to a file unlike > which overwrites the file
  - **$ echo "world" >> a.txt**

Communication between Commands (Pipes)

- The shell pipe (|) is a way to communicate between commands.
- Basically it passes output of first command as input to second and so on.
- Examples:

```
$ cat a.txt | sort -n
$ cat a.txt | sort -nr
$ cat a.txt | sort -n | head -n 5  # show the first 5 lines
$ cat a.txt | sort -nr | head -n 5  # show the first 5 lines
```

Linux Editors

Editor Concepts

- Editing a file is to modify the content of a file
- Text editor:
  - Enter and modify text in a text file
- Word processor:
  - Enter, modify and <u>format</u> text in a document
- Line editor:
  - Edit file one line at a time
  - Unix examples: ex, ed and sed
- Full screen editor
  - Shows a whole screen of text at a time

Editor Features

- enter text
- search and replace
- copy, cut and paste
- undo and redo
- importing and exporting text
- save and cancel

Text Files

- Linux file name does not require file extension
- Linux file system does not consider the extension when treating files
- However, some extensions are commonly used
- Program source code: .c .cc .cpp .f .f77 .f95
- Compiled object code: .o .a .so .sa
- Compressed files: .z .gz .zip
- Archive files: .tar .tz
- Web site source code: .html .shtml .php
- Executable files typically have no extension
- Text files that will be moved to Windows: .txt

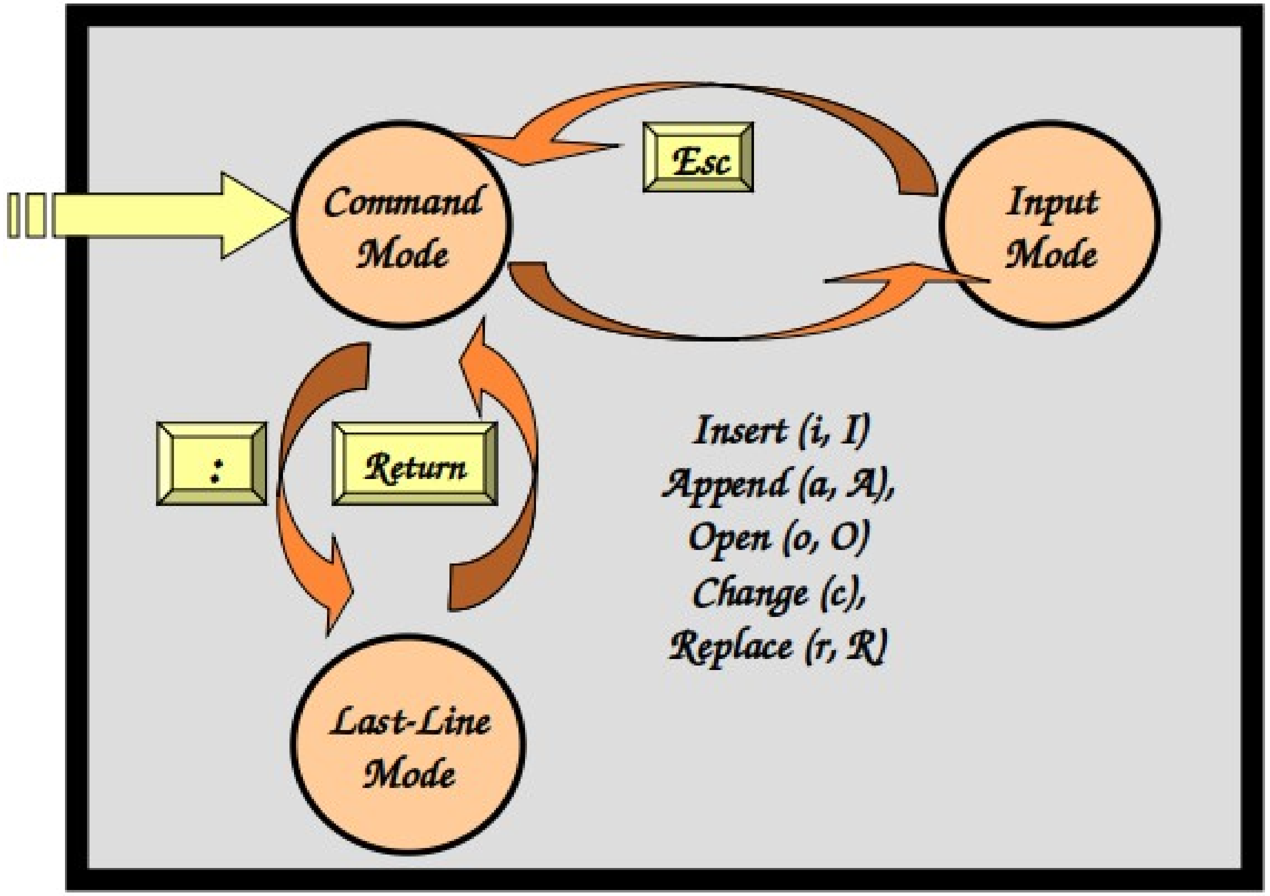
Unix Text Editors

- Console Based
  - vi
  - emacs
  - nano
- GUI editors
  - Gedit
  - Xedit

vi Editor

- short for: visual editor
- available on all Linux systems
  - original vi part of BSD Unix
    - written by Bill Joy in 1976
  - many derived, improved versions available
  - open source vim (vi improved)
    - is part of GNU/Linux
- vi has multiple modes of operation:
  - input mode, command mode, last-line mode

vi Editor Editing Modes



vi Editor

- To end vi tutorial in the middle of the session, execute the command :q!
  - :q! = quit without saving
  - :wq = write out (save) and quit

- F1 = help
  - or :help
  - :help <command>
  - :q to exit help window

vi Editor Commands

- Delete characters
  - x deletes character under the cursor

- Insert characters
  - i converts to insert mode
  - then type characters
  - <esc> to exit insert mode

vi Editor Commands

- Insert lines
  - o = open line below cursor
  - O = open line above cursor
  - <esc> to exit insert mode

- Append characters
  - A converts to insert mode at end of a line
  - then type characters
  - <esc> to exit insert mode

vi Editor Commands

- Deletion
  - d$ deletes to end of line
  - dw deletes to beginning of next word
  - de deletes to end of current word
  - d + motion
- Using motions for movement
  - Use any of the motions above
  - Use count for repetition
  - 2w = move cursor two words forward
  - 0 = start of line

vi Editor Commands

- Using repetition as part of deletion
  - 2dw deletes next two words
- Deleting a line
  - dd = delete line
  - 2dd = delete two lines
- Undo
  - u = undo one command
  - U = restore a line
  - cntl-R = redo a command

vi Editor Commands

- p = put back the deleted text (in new place)
  - one of the delete command above + put = cut-and-paste

- More general cut-and-paste
  - v = start visual mode (start block)
  - move cursor to end of block
  - y = yank (copy to buffer)
  - then p = put in new place

vi Editor Commands

- Location
  - ctrl-g = show position in file
  - G = go to bottom of file
  - gg = go to top of file
  - <number>G = go to line <number>

vi Editor Commands

- Search
  - /<phrase> = search
  - /<phrase>\c = ignore case
  - ?<phrase> = search backwards
  - n = repeat search
  - N = repeat search in the other direction
  - cntl-o = move backward one instance
  - cntl-i = move forward one instance
- Search for matching parentheses
  - Put cursor on (, [or {
  - % = go to matching one
  - % = go to first one again

vi Editor Commands

- Files
  - :w filename = write a file (save)
  - :!ls = list directory
  - :!xx = any command

- Substitute (replace)
  - :s/thee/the = changes first one
  - :s/thee/the/g = changes all (global change)
  - :s/thee/the/gc = change all with query
  - :#,#/thee/the/g = only change within that line range

Emacs Editor

- originally started as editor macros in 1976
- Gosling Emacs available for Unix in 1981
- GNU Emacs created by Richard Stallman in 1984
  - very popular editor on Unix until recently
  - history: editor war: emacs vs. vi
- uses lisp-like macro language for powerful features and extensions:
  - programming language sensitive editing
  - email client
  - news reader
- has built-in tutorial: ^h-t

THE PICO AND NANO EDITORS

- part of the popular pine mail utility on UNIX
- developed by the University of Washington

- pico = <u>pi</u>ne email <u>co</u>mposer

- nano is *improved* open source of pico available for GNU/Linux
  - very intuitive operation
  - on-screen guide and help

GUI Editors

- use onscreen direct manipulation via mouse and menus
  - gedit
  - xedit
- require to run X11 window server

What’s Next?

- Shell, AWK, SED

- Some Linux Utilities