terminal INTERACTIONS

Directories:

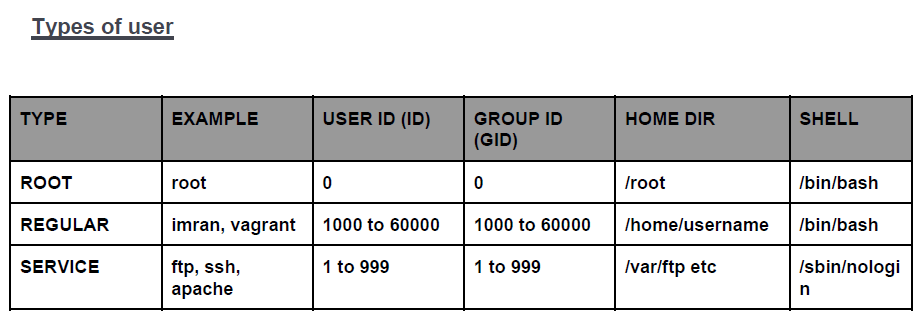
* Home directory:
  + /root – for root user
  + /home/ username- for specific user
* User executable:
  + /bin – root user
  + /usr/bin
  + /usr/local/bin
* System executable:
  + /sbin
  + /usr/sbin
* Configuration:
  + /etc
* Temporary files:
  + /tmp

Linux commands:

|  |  |  |
| --- | --- | --- |
| Command | Options | Description |
| pwd |  | **Present Working Directory** |
| mkdir |  | **Create a directory/folder** |
| cd |  | **Change Directory** |
| ls | -l  Long listing details  -a  Hidden files and directories  -g  List with group names  -n  List UID and GID  -r  In reverse order  -t  Sorted  -R  List subdirectories and files as well | **List directories and files** |
| touch |  | **Create empty files** |
| cp <source> <destination> | -r  recursively | **Copy from source to destination** |
| mv <source> <destination> | -i  Throw warning for overwriting | **Move from source to destination** |
| rm <file> | -rf <directory>  -R for directory | **Remove file** |
| cat <file> |  | **Read file** |
| ln -s <destination> <linkname> |  | **Link name will direct to file original location** |
| grep <word> <location> | -i  Ignore case | **Grep command is used to search for text in file** |
|  | -v  Reverse selection |  |
| **Filter commands** |  |  |
|  | less <file>  Enter to scroll down  d for next page  b for previous page  / search for a word in file | **Display file content page wise or line wise** |
|  | more <file> | **Same like less** |
|  | head <file> | **Display top 10 lines** |
|  | tail <file> | **Display last 10 lines** |
| cut -d -f <file> | d = delimiter  f = field |  |
| sed ‘s/searchfor/replacewith/g’ filename | Sed = stream editor | **Used to search a word in the file and replace in the output but original file remains unchanged** |
| >  >> | Copy a command output to any file | **I/O redirection using one or two >>** |
| | | ls | grep loggdir | **Piping, output of left program is sent as input to right program** |
| diff <file1> <file2> |  | **Compares two files line by line** |

**Users & Groups:**

* User and groups are used to access control
* User is associated to a group
* Every user has UID (Unique user id)
* User id and UID are stored in /etc/passwd
* User password is stored in /etc/shadow



**File Permission:**

**VIM editor:**

Vim filename – will launch the file in vim editor

* Hit I or insert key to start writing
* To save and close: Esc >> :wq

Modes:

1. Command mode
2. Insert mode
3. Extended command mode

In Command mode:

|  |  |
| --- | --- |
| **Key** | **Description** |
| gg | Beginning of page |
| G | End of page |
| u | Undo |
| VY | Copy a line |

In Extended mode:

|  |  |
| --- | --- |
| **Key** | **Description** |
| Esc + :w | Save the changes |
| Esc + :q | Quit without saving |
| Esc + :wq | Save and quit |
| Esc + :wl | Save forcefully |

Types of files:

|  |  |  |
| --- | --- | --- |
| **Symbol** | **Type** | **Description** |
| ~ | Regular file | Normal file text, executable files |
| d | Directory |  |
| l | Link | Shortcut that points to file location |
| s | Socket |  |
| p | pipe | Process communicate without network |

**File System:**

/bin : /bin is the directory that contains binaries, that is, some of the applications and programs you can run.

/boot : The /boot directory contains files required for starting your system

/dev: /dev contains device files. Many of these are generated at boot time or even on the fly.

/etc: contains system configuration files

/home: /home is where you will find your users’ personal directories

/lib: /lib is where libraries live. Libraries are files containing code that your applications can use

/media: The /media directory is where external storage will be automatically mounted when you plug it in and try to access it

/mnt: The /mnt directory, however, is a bit of remnant from days gone by. This is where you would manually mount storage devices or partitions.

/opt: The /opt directory is often where software you compile

/root: /root is the home directory of the superuser

/run: /run is another new directory. System processes use it to store temporary data for their own nefarious reasons

/srv: The /srv directory contains data for servers. If you are running a web server from your Linux box, your HTML files for your sites would go into /srv/http (or /srv/www). If you were running an FTP server, your files would go into /srv/ftp

/tmp: */tmp* contains temporary files, usually placed there by applications that you are running.

**File Permission:**

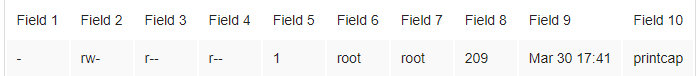
To check permission use **ls-l**



|  |  |
| --- | --- |
| r | Permission to read |
| w | Permission to write |
| x | Permission to execute |
| - | No permission |

|  |  |
| --- | --- |
| **Field** | **Description** |
| 1 byte | File type (-, d, l) |
| 3 byte | Owner’s permission |
| 3 byte | Group’s permission |
| 3 byte | Other user permission |
| 1 byte | Number of links or directories inside that directory |
| 1 byte | User who owns that file |
| 1 byte | Group that belongs to |





Changing Permission:

* Only root can change file’s owner
* Only root or owner can change file’s group

Ownership is changed using **chown**

**Chown [-R] user\_name file/directory**

Group ownership is changed using chgrp

**Chgrp [-R] group\_name file/directory**

**Changing permission:**

**Chmod [-option] mode file/directory**

* Option includes:
  + -R = recursive
  + -v= verbose
  + –reference = reference other file for mode
* Mode includes:
  + u – user, g – group, o – other, a - All
  + + (grant), - (remove) , = (set)
  + R,w x

Chmod ugo+r file = grant read access to all for a file

chmod u=rw,og=r new\_file.txt

Using the “=” operator means we wipe out any existing permissions and then set the ones specified.

$PATH = system env. Variable which holds all executables

cd ./ = in current directory

cd ../ = in previous directory

tr i.e. translate commands are used to replace things like

tr a-z A-Z will replace all lower case to upper case

**$ tr [OPTION] SET1 [SET2]**

**Wildcards:**

\*= represents zero or more characters

?=represents a single character

[ ]= range of characters

ACL (Access Control List): Used to define more fine-grained access rights for files or directories- commands used are

Setfacl- will set permissions

Getfacl- gives permission to see existing permissions

Example:

For a user:

setfacl -m u:user:rwx /path/to/file

For a group:

setfacl -m g:group:rwx /path/to/file

To allow files and directories to inherit ACL, use option -Rm ( recursive)

To remove

Setfacl -x u:user /path/to/file

**Input Output Redirects:**

1. Standard Input (stdin)- file description number -0
2. Standard output (stdout) - file description number -1
3. Standard error (stderr) - file description number -2

***Stdout- used to redirect console output***

Redirect to a file / overwritten >

Use the same file or append >>

***Stdin- used to get file contents of a file***

< is used as input redirect

Mail -s “office memo” < memoletter

X < 1+4

***Stderr- used to redirect errors***

Using file descriptor i.e. 2

Ls -l /root 2> errorfile

telnet localhost 2> errorfile

**Tee command**: used to store and view the output at the same time

Command | tee <filename>

Use option -a to append

Command | tee -a <filename>

**Filter /Text Processors Commands:**

1. Cut- cut some text

Cut -c1 filename == get first character of every line of file

Cut -c1, 2,4 filename == get the first, second and fourth character from every line of file

Cut -c1-3 filename == get the range of characters from 1-3

Cut -c1-3, 6-8 filename

Ls -l | c2-4 >> newfile.txt

1. Awk - used to extract fields from a file or output

awk '{action}' your\_file\_name.txt

When you want to search for text that has a specific pattern or you're looking for a specific word in the text, the command would look something like this:

awk '/regex pattern/{action}' your\_file\_name.txt

* **$0**: Represents the entire line of text.
* **$1**: Represents the first field.
* **$2**: Represents the second field.
* **$7**: Represents the seventh field.
* **$45**: Represents the 45th field.
* **$NF**: Stands for “number of fields,” and represents the last field.
* **$NR**: will give line number

awk ‘{print NR, $1}’

A BEGIN rule is executed once before any text processing starts. In fact, it’s executed before awk even reads any text. An END rule is executed after all processing has completed. You can have multiple BEGIN and END rules, and they’ll execute in order.

awk 'BEGIN {print "Dennis Ritchie"} {print $0}' dennis\_ritchie.txt

who | awk 'BEGIN {print "Active Sessions"} {print $1,$4}'

Input separator: If you want awk to work with text that doesn’t use whitespace to separate fields, you have to tell it which character the text uses as the field separator. For example, the /etc/passwd file uses a colon (:) to separate fields.

We’ll use that file and the -F (separator string) option to tell awk to use the colon (:) as the separator

awk -F: '{print $1,$6}' /etc/passwd

awk ‘/Jerry/ {print }’ file == will return lines where Jerry is matched

replace a word and print

awk ‘{$2=”Kumar”; print $0}’ file – will replace second column values with Kumar

awk ‘length{$0} >15’ file == give all lines of length greater than 15

1. Grep and egrep – search for a keyword

Grep <keyword> file

Grep -c <keyword> file == will give count of word

Grep -i <keyword> file == ignore case sensitivity

Grep -n <keyword> file == display the matched lines and their line numbers

Grep -v <keyword> file == display everything but keyword

Egrep is used to search two words in a file

egrep -I “keyword1|keyword2” file

1. Sort / Uniq - Text processor commands

Sort command sorts in alphabetical order

Uniq command filters out repeated or duplicate lines

Sort file == sort the file in alphabetical order

Sort -r file == sort in reverse order

Sort -k<filednumber> file == sorting apply to field number by default it takes first field

Uniq file == removes duplicate

Sort file | uniq == sorting with unique

Sort file | uniq -c === sorting with unique but with count of repeats

Sort file | uniq -d == only show repeated lines

1. Wc – text processor commands

This command gives newline count, word count, byte count

Wc file == output in the format of <no of lines> <no of words> <no of bytes> <filename>

Wc -l file == only word count

Wc -c file == only byte count

1. Diff and cmp

Diff- compares line by line

Cmp- compares byte by byte

**Compress and Un-compress files:**

tar – takes multiple files and put them together in a file like zip

gzip- compress a file

gzip -d or gunzip – uncompress (unzip) file

**Truncate file size: shrink or extend file size**

truncate -s 10 filename ==

**Combining and splitting a file:**

Cat file1 file2 file3 … > file4 === combining files

Split -l 300 file1.txt childfile === split file1.txt into 300 lines per file and output to childfile

**Admin part:**

sed command:

substitute command, it replaces a string

find and delete a line

removes empty line

**Shell Programming**

Echo $0 == current shell

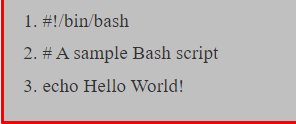
Cat /etc/shells == available shells

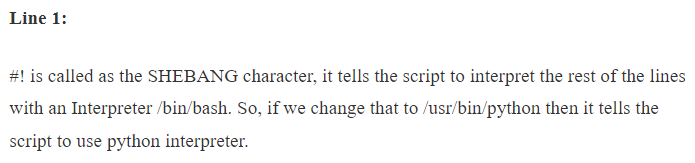
Types of shell:

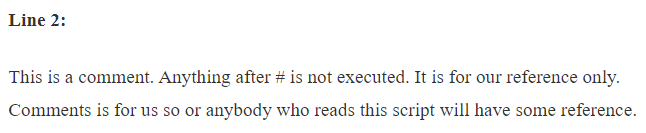
1. Gnome- GUI
2. KDE - GUI
3. Sh – CLI
4. Bash -CLI
5. Csh and tcsh
6. Ksh

**Scripting:**

* Shell commands are executed sequentially
* The file can contain
  + Shell ( #! /bin/bash)
  + Comments (#)
  + Commands
  + Statements ( if, while, for, do ..)
* Shell script should have executable permission
* Shell script has to be called from absolute path (e.g- /home/userdir/script.sh)







<https://visualpath.in/devopstutorials/devops>

Variables:

Types of operations:

Setting a value for a variable

VarName = value

Getting a value form a variable

$varName

