UNIX Commands

Created By: Prof. Mital Goswami

Login Commands

passwd

logout

who

who am I

uname

clear

Login Commands

Passwd

The **passwd** command changes passwords for user accounts. A normal user may only change the password for his/her own account, while the superuser may change the password for any account.

OPTIONS

-d, --delete

Delete a user's password (make it empty).

-e, --expire

Immediately expire an account's password.

-l, --lock

Lock the password of the named account.

Login Commands

-u, --unlock
Unlock the password of the named account.

Example :
Change your own password:
$ passwd

output:
$ passwd
Changing password for ubuntu.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully

Login Commands

logout

logout command allows you to programmatically logout from your session. causes the session manager to take the requested action immediately.

For logout `exit` OR `logout` OR `bye` command is used

EXAMPLES

Example:

To logout from current user session:

$ logout

output:

no output on screen, current user session will be logged out.

Login Commands

who

who - show who is logged on

OPTIONS

-l, --login

print system login processes

-q, --count

all login names and number of users logged on

-u, --users

list users logged in

-a Display All Users

Login Commands

whoami

whoami prints the effective user ID.

This command prints the username associated with the current effective user ID.

syntax

whoami [OPTION]

OPTIONS

--help

Display a help message, and exit.

--version

Display version information, and exit.

Login Commands

clear

clear the terminal screen.

EXAMPLE

Clear the terminal

$ clear

Login Commands

uname:

- • Display about the system.

SYNTAX: uname OPTION DETAILS

- -a Kernel name, network node hostname, kernel release date, kernel version, machine hardware name, hardware platform, operating system
- -s Kernel name
- -n hostname of the network node(current computer).
- -r Kernal release name
- -v Version of kernel
- -m Machine hardware name
- -p Types of processor
- -i Platform of hardware
- -o Operating system name

File & Directory Commands

ls - list directory contents.
ls List information about the FILEs (the current directory by default).

Options
-l
use a long listing format
--author
with –l, print the author of each file.
-C
list entries by columns
-g
like -l, but do not list owner
-G, --no-group
in a long listing, don't print group names

File & Directory Commands

-m

fill width with a comma separated list of entries

-S

sort by file size

-r, --reverse

reverse order while sorting

-t

sort by modification time

-x

list entries by lines instead of by columns

File & Directory Commands

Example:

To list all files of current directory:

$ ls

output:

ls

File & Directory Commands

cat

- The cat (short for "concatenate") command is one of the most frequently used command in Linux/Unix like operating systems. cat **command allows us to** create single or multiple files, view contain of file, concatenate files and redirect output in terminal or files.

SYNTAX

cat [Options] [File]…

Example :

Create two sample files

#sample.txt

This is a sample text file

#sample1.txt

This is a another sample text file

File & Directory Commands

To display content of a file.
$ cat sample.txt
This is a sample text file

To display content of all txt files.
$ cat *.txt
This is a another sample text file This is a sample text file

To concatenate two files.
$ cat sample.txt sample1.txt > sample2.txt
$ cat sample2.txt
This is a sample text file
This is a another sample text file

File & Directory Commands

To display content of a file.
$ cat sample.txt
This is a sample text file

To display content of all txt files.
$ cat *.txt
This is a another sample text file This is a sample text file

To concatenate two files.
$ cat sample.txt sample1.txt > sample2.txt
$ cat sample2.txt
This is a sample text file
This is a another sample text file

File & Directory Commands

cd

To change directory - change the current working directory to a specific Folder.

If directory is given, changes the shell's working directory todirectory.

EXAMPLES

Move to the sample folder

$ cd /usr/local/sample

Move up one folder

$ cd ..

Get back to original location

$ cd /

#root directory

$ cd ~

#home directory

File & Directory Commands

pwd

print name of current/working directory.

The current directory is nothing but the directory in which you are currently operating while using bash or ksh.

Syntax

pwd [options]

Example

To print current working directory, enter:

$ pwd

output:

/home/user

File & Directory Commands

mv

The ***mv*** command allows you to move and rename files.

Syntax

The syntax for the ***mv*** command is:

mv [options] sources target

Options

Option	Description
-f	Forces the move.
-i	Prompt for a confirmation before overwriting any files.

Example

mv -f tech /usr

File & Directory Commands

cp

To copy one or more files to another location.

SYNTAX

cp [options]... Source Dest

EXAMPLES

Copy sample.txt to sample.bak.

$ cat sample.txt

This is a sample file

$ cp sample.txt sample.bak

$ cat sample.bak

This is a sample file

File & Directory Commands

ln

creates links between files. **ln** creates a link to file *TARGET* with the name *LINKNAME*.

Syntax

$ ln file1 file2

Example

$ cat target.txt
target file

$ ln target.txt link.txt
$ cat link.txt
target file

File & Directory Commands

rm

rm removes each specified file. By default, it does not remove directories.

SYNTAX

rm [OPTION]... FILE...

Example-1:

Remove the file myfile.txt. If the file is write-protected, you will be prompted to confirm that you really want to delete it:

$ rm myfile.txt

$rm -r folder_name

Example-2:

Remove the file myfile.txt. You will not be prompted, if the file is write-protected .

$ rm -f myfile.txt

File & Directory Commands

rmdir

Remove the DIRECTORY(ies), if they are empty.

Syntax :

rmdir [OPTION]... DIRECTORY...

Example :

rmdir command will delete the empty directories. i.e directory without any sub-directories or files:

$ rmdir test

File & Directory Commands

mkdir

Create the DIRECTORY(ies), if they do not already exist.

Syntax :

mkdir [OPTION]... DIRECTORY...

Example :

Creates a new directory called mydir whose parent is the current directory.

Also we can make multiple directory using following command:

$mkdir bca bscit

$ mkdir mydir

output:

$ls

File & Directory Commands

umask

- a new file's permissions may be *restricted* in a specific way by applying a permissions "mask" called the **umask**. The **umask** command is used to set this mask, or to show you its current value.

Syntax

$ umask [-S] [*mask*]

Options

-S Accept a symbolic representation of a *mask*, or return one.

To view your system's current **umask** value, enter the command:

$ umask

File & Directory Commands

- which will return your system's umask as a four-digit octal number, for example:

0002
0002 is the same as **002**.

To view this as a symbolic representation, use the **-S** flag:

$ umask -S

Which will return the same value symbolically, for example:
u=rwx,g=rwx,o=rx

File & Directory Commands

chmod

chmod changes the permissions of each given file according to mode, where mode describes the permissions to modify. Mode can be specified with octal numbers or with letters.

chmod [Options]... Mode

Numeric mode

A numeric mode is from one to four octal digits (0-7), derived by adding up the bits with values 4, 2, and 1. Permissions for the user who owns the file: read (4), write (2), and execute (1).

File & Directory Commands

EXAMPLES

- Read by owner only
  $ chmod 400 sample.txt

Read by group only
$ chmod 040 sample.txt

Read by anyone
$ chmod 004 sample.txt

Write by owner only
$ chmod 200 sample.txt

Write by group only
$ chmod 020 sample.txt

File & Directory Commands

Write by anyone
$ chmod 002 sample.txt

Execute by owner only
$ chmod 100 sample.txt

Execute by group only
$ chmod 010 sample.txt
Execute by anyone
$ chmod 001 sample.txt

Allow read permission to owner and group and anyone.
$ chmod 444 sample.txt
Allow everyone to read, write, and execute file.
$ chmod 777 sample.txt

- Reference for u = owner , g = group , o = other , a = All (Owner , Group , Other). • Add Read , Write , Execute permission to owner.

$chmod u+rwx text.txt

- Remove write permission for the group and other.

$chmod go-w text.txt

- Read and Write for owner , for other and group read only.

$chmod u+rw,go+r text.txt

File & Directory Commands

chown

- To change owner, change the user and/or group ownership of each given File to a new Owner.

chown [Options]... NewOwner File...

EXAMPLES

Change the owner of file.

$ chown user sample.txt

Change the group of file.

$ chown :mygroup file.txt

Change both the owner and group of file in single command.

$ chown user:mygroup file.txt

File & Directory Commands

chgrp

- To change group ownership. 'chgrp' command changes the group ownership of each given File to Group (which can be either a group name or a numeric group id).

chgrp [Options]... {Group} File...

EXAMPLES

To Make oracleadmin the owner of the database directory

$ chgrp oracleadmin /usr/database

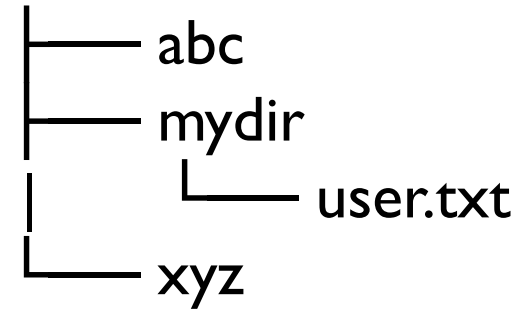
$sudo chgrp group2 file1.txt

File & Directory Commands

find

- The find command in UNIX is a command line utility for walking a file hierarchy. It can be used to find files and directories and perform subsequent operations on them. It supports searching by file, folder, name, creation date, modification date, owner and permissions.
To find a single file by name pass the -name option to find along with the name of the file you are looking for.

Suppose the following directory structure exists shown here as the output of the tree command.



File & Directory Commands

The file user.txt can be located with the find by using the -name option.

$find ./user -name user.txt

./user/mydir/user.txt
Find file with entered permission (read write /read write / read)

$find ./ -perm 664

How to find and delete a file

To find and delete a file pass the --delete option to find. This will delete the file with no undo so be careful.

$find ./user -name user.txt --delete

$find . -type d

How to find a directory

To find a directory specify the option -type d with find.

find ./user -type d -name mydir
./user/mydir

Example

- find . -name a.txt

- #Find Read-Only Files
- find / -perm /u=r

- find sss -name c.txt

- find -type d -name sss

- find ~ -name "a.txt"

- #Find all Empty Directories
- find /tmp -type d -empty

- #. File all Hidden Files
- find /tmp -type f -name ".*"

- find / -size 50M

File & Directory Commands

pg

- Pg is a pager: it allows you to view text files one page at a time. pg displays a text file on a CRT one screenful at once. After each page, a prompt is displayed. The user may then either press the newline key to view the next page or one of the keys described below.

pg [options] [*file*...]

Options

-number : The number of lines per page. Usually, this is the number of CRT lines minus one.

-c : Clear the screen before a page is displayed, if the terminfo entry for the terminal provides this capability.

-e : Do not pause and display "(EOF)" at the end of a file.

-f : Do not split long lines.

File & Directory Commands

Example

$ pg myfile.txt

Displays the first screenful of the contents of text file myfile.txt, and a prompt (":"). Pressing the Return key displays the next page.

File & Directory Commands

more

- **more** is a filter for paging through text one screen at a time. The RETURN key displays the next line of the file. The spacebar displays the next screen of the file.

Syntax

more [options] [files]

Options

-c	:	Page through the file by clearing the window. (not scrolling).
-d	:	Displays "*Press space to continue, 'q' to quit*"
-w	:	Waits for a user to press a key before exiting.
-n	:	Displays n lines per window.

File & Directory Commands

- **examples**
  more +3 myfile.txt
  Display the contents of file **myfile.txt**, beginning at line 3.

  more +/"hope" myfile.txt
  Display the contents of file **myfile.txt**, beginning at the first line containing the string "**hope**".

  ls | more
  List the contents of the current directory with **ls**, using **more** to display the list one screen at a time.

File & Directory Commands

- **less**

This command is used to see the information of the document page wise. This command work just like more command and it is older version.

Syntax

$less File Name **Press q to exit**

Example

$less abc.txt

File & Directory Commands

Navigation of less Command

f or spacebar	To move page forward
b	To move back one page
Enter or j	One line forward
k	One line back
p	Beginning of file
G	End of file
q	Quit

File & Directory Commands

head

head command help in viewing lines at the beginning of the file respectively.

Syntax:

$head –LineNumber File Name

Unless otherwise specified the head command assumes that you want to display first 10 lines in the file. Should you decide to view first fifteen lines you simply have to say.

$head -5 myfile

Here with head command we can specify the number of lines if we decide to override this default value.

File & Directory Commands

tail

tail command help in viewing lines at the ending of the file respectively.

Syntax:

$tail –LineNumber FileName

Unless otherwise specified the tail command assumes that you want to display last 10 lines in the file. Should you decide to view last fifteen lines you simply have to say.

$tail -5 myfile

Here with tail command we can specify the number of lines if we decide to override this default value.

File & Directory Commands

wc (Word Count)

This command is used to count number of characters, words and lines in an given input file. If file is not given, it takes input from standard input.

Syntax

$wc [Option] File Name

Options

-l	Counts number of lines
-w	Counts number of words
-c	Counts number of characters

File & Directory Commands

Example

A file abc.txt having following data

1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9 10

$ wc abc.txt

Output

2 20 42 abc.txt

Note: Spacebar and Enter also known as on character

File & Directory Commands

touch (Changing the time stamp)

This command is used to change the time stamp.

Syntax

$touch File Name

EXAMPLE:

$touch -d '1 MAY 2023 11:11' std1.txt

Timestamp	**Description**		
Access time	The last time the file was read.	-	**atime**
Modification	The last time the contents of the file were modified.	-	**mtime**
Change time	The last time the file's Status, was changed.	-	**ctime**

File & Directory Commands

With no options, **touch** will change the atime, mtime, and ctime of file to the current system time.
-a:-This option changes the access time only
-m:-This option changes the modification time only.
-r:-Uses the access and modification times from the reference file.

Examples

$ touch file.txt
$ touch -c file.txt
$touch a
$touch b
$stat a
$touch -m a b
$stat a
$touch -r a.txt b.txt
$touch m.txt a.txt -m
#set the time of m replace a

touch -c -d "16 sep" doc2
#This is used to update access[-c] and modification[-d: date string] time
touch -d "2 hours ago" filename
touch test{1..10}

stat:

• stat is command which is gives information about the file and filesystem. • Gives information such as the size of the file, access permission and user ID and group ID.

EXAMPLE: stat std1.txt

alias:

• alias command instructs the shell to replace one String with another string while executing the commands.

EXAMPLE: alias d = "cd Desktop"

$$alias ll='ls –l'

$alias –p

Type:

type command is used to get type of the command means it's return type of the provided command.

type command_name

type ls

Type	Meaning
alias	A shortcut command
builtin	Built into the shell
file	External executable
function	Shell function
keyword	Shell reserved word

Redirection & Piping

Redirection is a feature in Linux such that when executing a command, you can change the standard input/output devices. The basic workflow of any Linux command is that it takes an input and give an output.

The standard input (stdin) device is the keyboard.
The standard output (stdout) device is the screen.

Output Redirection
The **'>'** symbol is used for output (STDOUT) redirection.

Example:
ls -l > listings

Here the output of command ls -l is re-directed to file "listings" instead of your screen.

Redirection & Piping

>> : Does the same as >, except that if the target file exists, the new data are appended.

command >> out.txt

If out.txt exists, the output of command will be appended to it, after whatever is already in it. If it does not exist it will be created.

Input redirection
The '**<**' symbol is used for input(STDIN) redirection.

Example: The mail program in Linux can help you send emails from the Terminal.

Redirection & Piping

You can type the contents of the email using the standard device keyboard. But if you want to attach a File to email you can use the input re-direction operator in the following format.

Mail -s "Subject" to-address < Filename

<< : A here document. It is often used to print multi-line strings.

command << WORD Text WORD Here, command will take everything until it finds the next occurrence of WORD, Text in the example above, as input .

Redirection & Piping

Pipe operator

|

The pipe operator, it passes the output of one command as input to another. A command built from the pipe operator is called a pipeline.

command1 | command2

Any output printed by command1 is passed as input to command2.

Finding Pattern in Files

grep, fgrep, egrep

grep (globally search a regular expression and print)
This command can be used to search a pattern in one or more files directly from the command line.

SYNTAX
grep [OPTIONS] PATTERN [FILE...]

EXAMPLE
$ grep "Linux" input.txt
Welcome to Linux.

In the output above, the line in the file input.txt containing the pattern or string "Linux" was displayed as output.

Finding Pattern in Files

The pattern matching done by grep command is case sensitive. For example, if the argument to grep command is "LINUX" (instead of "Linux") then grep will not match the lines containing string "Linux".

```
$ grep "LINUX" input.txt
$
```

If it is desired that grep command should ignore the case sensitiveness then the option -i can be used.

```
$ grep -i "LINUX" input.txt
Welcome to Linux.
```

So we see that this time the string "LINUX" matched with the line containing the string "Linux".

Finding Pattern in Files

If more than one file is supplied in argument list then grep searches for the pattern or string in all the files.

For example :
$ grep "Linux" input.txt output.txt

input.txt:Welcome to Linux.
output.txt:I hope you enjoyed working on Linux.

The grep command also allows the usage of regular expressions in pattern matching.
grep -r ".*Linux" output.txt output1.txt
output.txt: I hope you enjoyed working on Linux.
output1.txt:Welcome to Linux.
output1.txt:I hope you will have fun with Linux.

Finding Pattern in Files

fgrep

fgrep searches for fixed-character strings in a file or files. **fgrep** is useful when you need to search for strings which contain lots of regular expression metacharacters, such as "**$**", "**^**", etc.

Syntex:

$fgrep [options] string [files]

Running **fgrep** is the same as running **grep** with the **-F** option.

example

$ fgrep "support" myfile.txt

Search for "**support**" in the file **myfile.txt**.

Finding Pattern in Files

egrep

Search for a pattern using extended regular expressions. A regular expression is a pattern that describes a set of strings.

egrep is essentially the same as running **grep** with the **-E** option.

syntax

egrep [*options*] *PATTERN* [*FILE...*]

Example 1

$ egrep "support|help|windows" myfile.txt

Search for patterns of support help and windows in the file **myfile.txt**.

Finding Pattern in Files

Example 2

$ egrep '^[a-zA-Z]+$' myfile.txt

Match any lines in **myfile.txt** which begin a line with an alphabetic word which also ends the line.

Example 3

$ egrep -c '^begin|end$' myfile.txt

Count the number of lines in **myfile.txt** which begin with the word '**begin**' or end with the word '**end**'.

Working with columns and fields

Cut – Removes sections from each line of files.
Prints selected parts of lines from each file to standard output.

Options

-f Extract a set of specified fields.

-d Used with the -f option. Use a specified delimiter rather than default tab.

Imagine we've got a file numbers.txt, which consist of three columns:

col_1	col_2	col_3
one	two	three
four	five	six

Working with columns and fields

In the following example cut will return only column (col_2) :

```
$ cut -f2 numbers.txt
col_2
two
Five
```

in the next example cut will return columns 2 and 3:

```
$ cut -f1,3 numbers.txt
col_1   col_3
one     three
four    six
```

Working with columns and fields

We've got the following file: fileone.txt

01234:567;89

ABCD:EFGH;IJ

display everything in first column before delimiter ";"

```
$ cut -d ";" -f1 fileone.txt
01234:567
ABCDE:FGH
```