Commands

Listing Files & Directories

`ls`

is used to list files and directories.

Using Additional Options

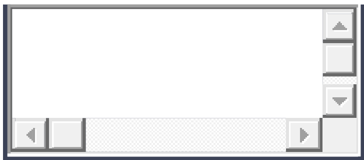
- **-l -h** option provides details in human readable format.
- Some options require values arguments/values to be passed.

  - **--block-size** option rounds the file size to nearest values.
  - Inputs: **KB** , **MB**
  -

Get Options for Commands

`help`

displays a list of options that you can use with the command.



```
ls --help
```

This Command will be helpful when you don’t know about its parameters and return type etc.

Clear the Screen

- `clear`

command clears the terminal.

- Shortcut:

Ctrl+L

.

```
clear
```

Get User Manual for Commands

- man

displays the user manual of a command.

- Here we pass the command as an argument.

Syntax:

```
man <command>
```

Example:

```
man ls
```

- Type **q** to exit the manual

Get System Date & Time

date

displays the system date and time.

```
date
```

Get Current User

`whoami`

displays the current logged in user.

```
whoami
```

Previous Commands

- Shell keeps track of the commands you have typed in.
- Use up (⍰) and down (⍰) arrows to access the commands.

History

- `history`

  display the history of the commands you have typed in so far.

- By default,It shows the last 500 recent commands.

```
history
```

Bash History

Bash maintains the history to *. bash_history* file.

```
cat .bash_history
```

exit

exit

to close/end a shell session.

```
exit
```

Summary

Command	Description
ls	List files & directories
date	Displays the system date and tin
whoami	Displays current logged in user
--help	Displays list of options for a given co
man	Displays user manual for a given com
clear	Clears the shell
history	It will show you the last five hundred commands
exit	Ends the shell

Working with File System

Working With Files

Creating a File

`touch`

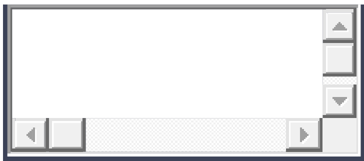
creates an empty file.

touch filename

Viewing File Content

`cat`

reads contents of file and prints it.



```
cat filename
```

Echo

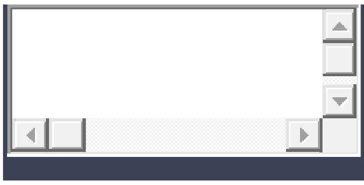
`echo`

output/prints a string in the terminal.

echo "content"

Writing to Files Using echo Command

Using `>` operator we can redirect the output of echo command to a file.



```
echo "Hello World!" > filename
```

Renaming a File

- mv

renames the file names.

- *destination* can be a new or existing file.

Syntax

mv source destination

Copying Files

cp

copies src_file to dest_file.

Syntax

cp src_file dest_file

Deleting a File

rm

removes (delete) files.

rm filename

Hidden Files

- Linux, by default, hides many of the sensitive system files, in order to avoid accidental changes.

- Hidden files starts with "."
- ls -a

shows the hidden files.

- ls -a

also shows the current and parent directories:

  - . represents **Current directory**
  - .. represents **parent directory**

```
ls -a
```

Summary

Command	Description
touch	Creates an empt
cat	Reads contents of file a
echo	Writes text to standa
mv	Renames the file r
cp	Copies content fro
rm	Removes (delete)
ls -a	Shows the hidden

Working with Directories

Creating A Directory

`mkdir`

creates a directory.

```
mkdir directory_path
```

Current Working Directory

`pwd`

prints name of current working directory.

Changing the Current Working Directory

`cd`

changes the current working directory.

```
cd directory_path
```

Note :

`cd /`

changes your current directory to root folder.

Creating a Directory in Directory

`mkdir`

creates a directory.

Switching to Parent Directory

- `cd ..`

  move to parent directory.

- Here `..` is relative path to parent directory.

```
cd ..
```

File Paths

There are two notations for file paths: 1. Absolute Path 2. Relative Path

Absolute Path:

Representing the complete path of a file or folder from the root.

Relative Path:

Representing the path of a file or folder wrt. current working directory.

In relative path conventions:

- `.` refers to the current working directory.
- `..` refers to the parent directory.

Home Directory

Each user in the computer is given a separate directory to work with - called home directory.

- `cd ~`

  can be used to switch to home directory.

```
cd ~
```

- cd

(cd and space) command can also be used to switch to home directory.

```
cd
```

Renaming a directory

mv

renames the directory name

Syntax

```
mv source destination
```

Example

```
mv tutorial commands
```

Moving a directory

mv

moves files or directories from source to destination paths.

mv source destination

Copying Files to Another Directory

`cp`

can be used to copy files between directories.

SyntaxCopying Files to Another Directory

`cp`

can be used to copy

Copying Directory

`cp -r`

can be used to copy a directory.

Syntax

```
cp -r source_path destination_path
```

cp -r source_path destination_path

Deleting a Directory

`rm -r`

removes(deletes) directories.

Syntax

```
rm -r directory_name
```

Example:

```
rm -r commands
```

Summary

We can use folder/file paths for cp, mv, rm commands.

Command	Description
mkdir	Creates a directory
pwd	Prints name of current working
cd	Changes the current working
rm -r	Deletes a directory

Working with Files

Text Editor

- A text editor is used for editing text files.

- Various text editors are:
    - Notepad++
    - Sublime Text
    - gEdit
    - Visual Studio Code etc..

Nano

Nano is an easy to use command line text editor for Unix and Linux-based operating systems.

Open file

To open a file with nano, pass the filename as an argument.

```
nano filename
```

Updating File

Add the text of the file in the middle of the editor.

Saving File

To save a file,

PRESS Ctrl+O

and

Enter()

.

Exit Nano

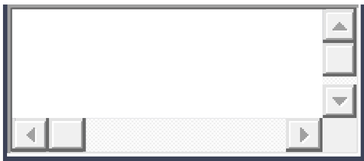
To exit from nano editor,

PRESS Ctrl+X

.

Viewing File Contents

To view file Contents:



```
cat filename
```

Filtering & Output Redirection

Filtering

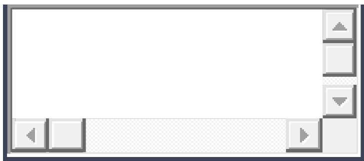
We can filter the contents of a file using the following filter commands.

- head
- tail
- grep

head

- Used to print top N lines of a file.
- By default, it will print the first 10 lines.

Syntax:



```
head [-N] filename
```

Example:

```
head -2 sentences.txt
```

tail

- Used to print last N lines of a file.
- By default it will print the last 10 lines.

Syntax:

```
tail [-N] filename
```

Example:

```
tail -2 sentences.txt
```

Counting

Word Count

`wc`

is used to find out number of lines, word count and characters count in the files.

Piping

- Pipe is used to combine two or more commands

- Output of one command is passed as an input to the command followed and so on.

Using '|'

Syntax:

```
command_1 | command_2 | command N
```

Example:

```
cat sentences.txt | head -2
```

Grep

Searches a file or files for lines that have a certain pattern.

Syntax:

```
cat filename | grep <pattern>
```

Example:

Output Redirection

`">"` takes the standard output of the command and redirects it to the file.

Syntax:

```
command > filename
```

Example:

```
cat sentences.txt | head -2 > learnings.txt
```

Compressing & Uncompressing Files

- File compression is a reduction in the number of bits needed to store the data of a file.
- Files are stored in such a way that, it uses less disk space than all the individual files and directories combined
- Advantages of compressing files are:
    - taking less disk space
    - easier and faster transmission
- Commonly used file formats for the compressed files:
    - gzip
    - zip
    - tar

tar

We can use tar to compress files & directories

Compression

Syntax:

```
tar -czvf file-name.tar.gz path1 path2 ..
```

Example:

```
tar -czvf my_collection.tar.gz videos report.txt
```

Extract/ Uncompress

Syntax:

```
tar -xzvf filename.tar.gz -C path
```

Example:

```
tar -xzvf my_collection.tar.gz -C collections
```

zip

It is used to package all the files into one file with .zip extension

Syntax:

```
zip -r zipfile.zip file1 folder1 file2 ...
```

Example:

```
zip -r collections.zip videos report.txt
```

unzip

The unzip command extracts all files from the specified ZIP archive

Syntax:

The unzip command extracts all files from the specified ZIP archive

Syntax:

```
unzip filename.zip -d path
```

Example:

```
unzip collections.zip -d new-folder
```

Summary

Command	Description
head	Used to print top N lines of a file
tail	Used to print last N lines of a file
wc	Used to find out number of lines, word count and characte
grep	Searches a file or files for lines that have a certa
tar	Used to compress files & directories
zip	Used to package all the files into one file with .zip
unzip	It extracts all files from the specified ZIP ar

Super User & File Permissions

Linux Users

The **root user**, also known as the **superuser** or **administrator**, has access to all commands and files.

Root User

```
sudo
```

command temporarily elevates the privileges allowing users to complete sensitive tasks without logging in as the root user.

Linux Commands

Executable Path

```
which
```

command is used to identify the location of a given executable path.

Syntax :

```
which command
```

Example :

```
$  which sudo
/usr/bin/sudo
```

Create New Users

`useradd`

is used to create a new user with the given username.

Syntax :

```
sudo useradd username
```

Example :

Set/Change User Password

`passwd`

is used to set or change password of a given user.

Syntax :

```
sudo passwd username
```

Execute Command as Another User

`su`

is used to execute command as another user.

```
su user
```

File Permissions

Authorization Levels

Multi-user operating systems like linux provide two levels of authorization in securing the files

- Ownership
- Permission

User Ownership

Users accessing a file/ directory can be categorized into 3 types

Type	Description
Owner	User who is considered as own
Group	A group of users, who are assigned a specif

Type	Description
Others	Any other users who doesn’t belong to the al

Ownership and Permissions

Permissions	Descripti
Read	Open and rea
Write	Modify the conte
Execute	Authority to run a

Changing Permissions

Syntax :

```
chmod permissions filename
```

Number	Binary	Symbol	
0	000	---	
1	001	--x	
2	010	-w-	
3	011	-wx	
4	100	r--	
5	101	r-x	
6	110	rw-	
7	111	rwx	

Changing Ownership

- For changing the user of a file/directory

Syntax :

```
sudo chown user filename
```

Example :

```
sudo chown root sample.txt
```

- For changing the user and group of a file/directory

Syntax :

```
sudo chown user:group filename
```

Example :

```
sudo chown root:root sample.txt
```

Packaging

Package & Repository

- A package file is a compressed collection of files that comprise the software package
- Packages are stored in repositories to make them accessible to users

Package Managers

apt

- APT stands for the Advanced Packaging Tool
- It is used to install, upgrade or remove software packages in linux

Syntax :

```
sudo apt
```

Installing Package

Installing a package from a repository

Syntax :

```
sudo apt install package_name
```

brew

- It is a Package Manager for macOS.
  - With

`brew` command you can install the packages in macOS.

Installing Package

Syntax :

```
brew install package_name
```

yum

`yum` is a graphical based package management tool for RPM (RedHat Package Manager) based Linux systems.

Installing Package

Syntax :

```
yum package_name
```

Downloading Files From Web

wget

`wget` is a command-line utility for downloading files from the web.

To install

`wget`

:

- apt

```
sudo apt install wget
```

- brew :

```
sudo brew install wget
```

- yum:

```
sudo yum install wget
```

`wget`

will download the resource specified in the url to the current directory.

Syntax :

```
wget "URL"
```

Example :

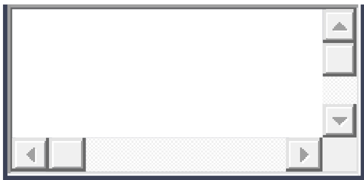
```
wget "https://www.lifewire.com/uses-of-command-wget-2201085"
```

curl

`curl` is a command-line utility for transferring data from or to a server designed to work without user interaction.

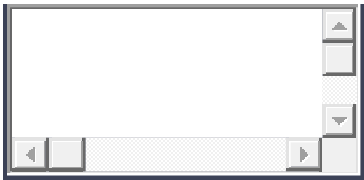
To install `curl`:

- apt:



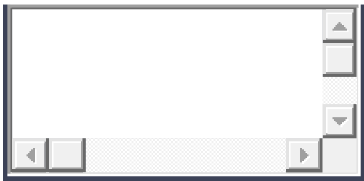
```
sudo apt install curl
```

- brew :



```
sudo brew install curl
```

- yum :



```
sudo yum install curl
```

`curl` prints the contents of the URL to the output.

Syntax :

```
curl "URL"
```

Example :

```
curl "wttr.in"
```

Searching a Package

apt-cache

command is used to searches for a particular package.

Syntax :

```
sudo apt-cache search package_name
```

Example :

```
apt-cache search google
```

Updating Packages

- System checks against the repositories.
- If newer version available of the program it will update the information about the existing packages and their versions available.

Updating a package from a repository:

upgrade

upgrade

will upgrade all the applications to latest version.

```
sudo apt upgrade
```

update

update

will simply update the information about the existing packages and their versions available

```
sudo apt update
```

Listing Installed Packages

dpkg -l

: It is used to list all installed packages

```
sudo dpkg -l
```

Adding Repository

PPA (Personal Package Archive) is an application repository that can be used to upgrade and install packages from third parties.

`add-apt-repository`

: It is used to add repository

Syntax :

```
sudo add-apt-repository repository_link
```

Adding the Private PPA's security key

Your Linux device can use that signature to check the authenticity of the packages.

`apt-key`

: Add PPA security key

Syntax :

```
sudo apt-key add - KEY_ID
```

Removing a Package

`remove`

: removes the installed packages

Syntax :

```
sudo apt remove package_name
```

Summary

Command	Descriptio
sudo	Temporarily elevates t
which	Used to identify the location of a
useradd	Used to create a new user with
passwd	Used to set or change passw
su	Used to execute command
chmod	Used to change the access p
chown	Changing the user of a f
apt	Used to install, upgrade or remove s
wget	Used for downloading file
curl	Used for transferring data fr
apt-cache	Searches for a particu
add-apt-repository	Used to add repo
apt-key	Used to add PPA sec

Warning

Under the installation of an

apt

package from a PPA, there is a change in one of the MongoDB installation steps, due to the changes made by the MongoDB website.

Update command from

```
curl https://www.mongodb.org/static/pgp/server-4.4.asc  | sudo apt-key add -
```

to

```
curl https://pgp.mongodb.com/server-4.4.asc | sudo apt-key add -
```

for a seamless monogodb installation.

Networking Commands

Network Connectivity

ping

checks the network connectivity between host and server/host.

Syntax :

```
ping hostname/IP address
```

Example :

```
ping google.com
```

Route Path

traceroute

prints the route that a packet takes to reach the host.

Syntax :

```
traceroute host_address
```

Example :

```
traceroute google.com
```

Network Interface

A network interface is a software interface to networking hardware.

Different types of network interfaces are:

- Ethernet
- Loopback etc.

Information About Network Interfaces

ifconfig

ifconfig

gives you the information of various network interfaces.

Environment Setup

Environment Variables

- Linux environment variables act as placeholders for information stored within the system.
- They will be available to the programs launched from the shell.

Listing

env

command can be used to print all the environment variables.

```
env
```

Creating & Updating

export

command is used to define/update value for a variable.

Syntax :

```
export VARIABLE_NAME=variable_value
```

Example :

```
export CUSTOM_ENV_VARIABLE=10
```

Accessing Environment Variables

- Use $ to access environment variables.
- When redefining variables, do not use the dollar sign.

Syntax :

```
echo $VARIABLE_NAME
```

Example :

```
$  echo $CUSTOM_ENV_VARIABLE
```

Delete Environment Variable

`unset`

command removes an environment variable.

```
unset VARIABLE_NAME
```

Commands Folder Path

PATH variable holds the colon-separated list of folder paths where executables are located.

```
echo $PATH
```

Aliasing Command

alias

is a (usually short) name that the shell translates into another name or command.

Syntax :

```
alias name=value
```

Example :

```
alias t=traceroute

t google.com
```

Un-aliasing Command

unalias

removing an existing alias is known as unaliasing.

```
unalias alias_name
```

Persistent Variables

Environment variables defined in a shell are deleted as soon as you exit the terminal.

To *persistent* environment variables we write them in .bashrc or ~/.bash_profile configuration files.

Editing .bashrc

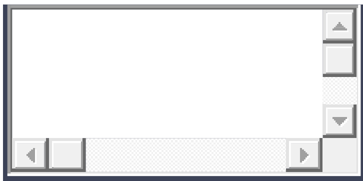
Syntax :

```
export VARIABLE_NAME=variable_value
```

Example :

```
# ~/.bashrc: executed by bash(1) for non-login shells.

# see /usr/share/doc/bash/Examples/startup-files (in the package bash-doc)

# for Examples

case $- in

    *i*) ;

...

export CUSTOM_ENV_VARIABLE=5
```

- To immediately apply all changes in .bashrc, use the `source` command



```
source ~/.bashrc
```