Markdown Language

<!-- Headings -->

# Heading 1

## Heading 2

### Heading 3

#### Heading 4

##### Heading 5

###### Heading 6

# Heading 1

## Heading 2

### Heading 3

#### Heading 4

##### Heading 5

###### Heading 6

<!-- Italics -->

\*This text\* is italic

\_This text\_ is italic

This text is italic

This text is italic

<!-- Strong -->

\*\*This text\*\* is strong

\_\_This text\_\_ is strong

**This text** is strong

**This text** is strong

<!-- Strikethrough -->

~~This text~~ is strikethrough

<!-- Horizontal Rule -->

---

\_\_\_

<!-- Blockquote -->

> This is a quote

This is a quote

<!-- Links -->

[Traversy Media](http://www.traversymedia.com)

[Traversy Media](http://www.traversymedia.com "Traversy Media")

<!-- UL -->

\* Item 1

\* Item 2

\* Item 3

\* Nested Item 1

\* Nested Item 2

<!-- OL --> (Displays incremental numbers e.g. 1, 2 , 3)

1. Item 1

1. Item 2

1. Item 3

<!-- Inline Code Block -->

`<p>This is a paragraph</p>`

<!-- Images -->

![Markdown Logo](https://markdown-here.com/img/icon256.png)

<!-- Github Markdown -->

<!-- Code Blocks -->

```bash

npm install

npm start

```

```javascript

function add(num1, num2) {

return num1 + num2;

}

```

```python

def add(num1, num2):

return num1 + num2

```

<!-- Tables -->

| Name | Email |

| -------- | -------------- |

| John Doe | john@gmail.com |

| Jane Doe | jane@gmail.com |

<!-- Task List -->

\* [x] Task 1

\* [x] Task 2

\* [ ] Task 3

Basic syntax is very simple:

* Use different number of # symbols to create headings of different sizes
* Use - to create lists
* Use numbers with a dot to create numbered lists – 1.
* Use ![](%path\_to\_an\_image%) to insert an image (do not forget to upload it to the repository)
* Use - [ ] and - [x] to create checklists
* Use \*\*text\*\* to make text **bold**
* Use ext to make text italic

Linux Commands

Pwd present working directory

Touch create folder

Cat view content

Ls list

Clear clear the screen

Tail shows last part of the file

Head Opposite tail

Which where on the file system the command is stored

Info Describes command

Man similar to info but more info

Cd change directory

Mkdir make directory

Cd – go back to the last folder that I was in

Ls -a show all files

Ls -la All information about all files

Rmdir remove directory (only if empty)

Rm remove

Rm -r remove folders recursively e.g. with content

Cp e.g. cp DevOps Desktop/

Mv move and rename e.g. mv DevOps Desktop or mv Desktop/DevOps Desktop/Learning\_Linux

Echo Returns back what’s written

Variables e.g. Name = Hossam, echo $Name

Course=DevOps, echo $Course

Mkdir -p Create all folders in one go e.g.

mkdir -p first\_folder/second\_folder/third\_folder/fourth\_folder

apt-get install tree sudo apt-get install tree

Text

Description automatically generated

First\_folder doesn’t have a parent so it’s the root folder

Fourth\_folder is the child folder

/ root

Tree / -L 1

Bin contains binary files, linked to usr/bin: can’t read binary files

Which ls shows where folder is stored

Which cat usr/bin/cat

Control + R + few letters of command

BOOT Files related to the booting of the system e.g. grub, efi, vmLinuz, System.map

CDROM Similar to the /media and /mount directories, where attached compact disk to the computer is accessed. If attach a disc to the computer, there has to be a mount point.

DEV tree /dev -L 1 special device files. A device file is an interface to a device driver

ETC Where configuration files are kept. To tweek and control how program works. E.g. Nginx needs to know where to store log files.

HOME Is where users keep specific files and folders

e.g. tree /home -L 1

LIB (library) Collection of resources used by the computer. Functions are grouped together to form a library

.so extension is used by the kernel

.py python libraries

Lost+found System recovery e.g. system shuts down abruptly to recover corrupt files

Media & Mnt (mount)

Opt Optional or option: Custom applications not tied to specific system

Proc An instance of an application. E.g. vim & will give ID number

Ls -l /proc | grep Proc\_ID

Ls /proc

Proc is used to store specific info about specific instances of programs running in the system

Root e.g. ls -l /root: no permission

Sudo ls -la /root: .profile, .cache…

/\_run The run directory is the mountpoint for tempfs filesystem in the computer’s memory. Temporary data used by processes are kept there.

df disk free – information on free space on the disk (showing in blocks)

A picture containing text

Description automatically generated

df -h human readable disk

Text

Description automatically generated

free memory usage & swap

free -h human readable format

ls /var/run & ls /run

.pid temporary processes running in the background

tempfs Temporary file system in memory which is mounted in the ROM directory.

id give more information about the logged in user

ls /run/user e.g. 1000

ls /run/sudo sudo ls/run/sudo

/sbin system binaries

/snap package management, installing software e.g. apt install

e.g. snap install tree

/srv service, server: where service files are kept

e.g. if ftp is installed in server

/swapfile

/sys system, kernel uses this: mounted on a virtual file system

e.g. if df -h /sys

/tmp Where temp files and directories are kept on disk. Files and folders get deleted.

/usr Where programs and libraries are being stored

/var System uses to write data

e.g. ls -ltr /var/log shows .log files

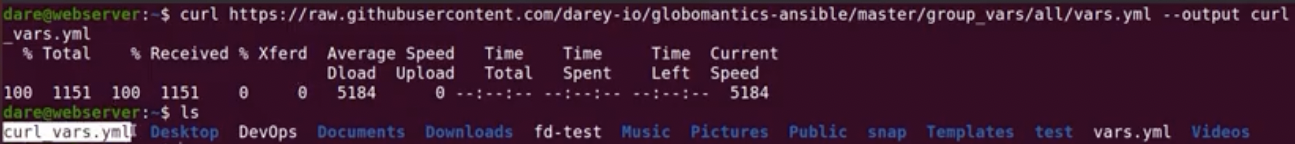
ls -ltr /var/lib contains dynamic data and libraries used by OS

ls -ltr /var/spool where data relating to email, news, printers stored

wget download file from the internet

less view file contents vs cat (dumps all in screen)

curl apt install curl



Text

Description automatically generated with medium confidence

vim e.g. nano editor,

grep sudo grep peter /etc/passwd

man grep

Text

Description automatically generated

Shows every line of the file that has es

History Shows all the commands that we have been using

diff get the difference between 2 files

Text

Description automatically generated

In vi editor, escape then :set number to show numbers

useradd e.g. sudo useradd -m peter (creates without password)

ls /home

id peter

sudo passwd peter

A screenshot of a computer

Description automatically generated with medium confidence

Sudo less /etc/passwd

CNTRL+R (reverse-i-search) Recursive search

passwd change password

Linux inode Every file has metadata, file table stores the file name and the inode number

inode entries: size, Device ID, UID, GID, Mode, Time stamp, pointer

/df -i Text

Description automatically generated

We can have 622592 inodes depending on how the partition was formatted

/ls -i 

/stat how to access inode of file to see metadata

Text

Description automatically generated

!s run the last command that has s in it

/chmod change mod, permissions on the file

e.g. chmod 777 file

ln link e.g. ln file1 file2

stat shows 2 links to same inode

File Descriptors and Redirections

All programs (commands) internally use libraries to function e.g. modules from the kernel, libraries from the program e.g. python import. Kernel keeps track of every open file: File Descriptors. The entries in these tables that are the file descriptors. Only the linux kernel is able to update this table because that entry only happens when there is an open file.

Why need file descriptors?

To track open files in the system

A way for the kernel to help the program that is requesting the file to give it more information such as:

* How to access the file
* Where is the file in the file system

File Descriptor points to the file table and the file table points to the inode table

Each time there is a file opening, there’s an entry that points to the file table specific to that file and in the file table is the inode entry that gives more information (metadata)

Redirection

e.g. echo hello > somefile

Text

Description automatically generated

0 Standard in (stdin) - Data that you input into the system from your keyboard

Data stream from keyboard into the system

1 Standard out (stdout)- Data stream that prints onto the screen

2 Standard error (stderr)-

Find e.g. find / -name darey.io

find / -name darey.io > stdout.txt (doesn’t store errors in file)

find / -name darey.io 2> error.txt

less error.txt (shows file with errors)

find / -name darey.io 2> /dev/null (discards errors)

vim error.txt & shows process id

ps -ef | grep proc\_id PS (process status),



lsof | grep proc\_id shows all files open because vim uses many libraries to do its job

Users Groups Permissions

ls -l /home

id root

sudo su switch user

Text

Description automatically generated

sudo groupadd jnrdev

sudo grep jnrdev /etc/group

sudo usermod -a -G jnrdev peter (G add to existing group, g remove from current group and add to new group)

id peter

sudo userdel peter

mkdir devops\_dir

ls -l

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| d= directory  - = file | user | group | others |  | User | Group |
| - | rwx | rwx | rwx | 2 | dare | Dare |
| d | rwx | rwx | r-x | 2 | dare | Dare |

Read, write, execute

ls -latr /user/bin/ls

/usr/bin/ls absolute path vs relative path

chmod change mode

chmod u+r file

chmod u-w file

chmod u+x file

chmod g+x file

chmod o+x file

Permissions can be assigned to:

u user

g group

o others

Permissions can be of

Read 4

Write 2

Execute 1

No permission 0

Permissions can be a combination

Read + Write + Execute 7

Read + Write 6

Read + Execute 5

Write + Execute 3

e.g. chmod 732 file

Client – Server

Network Protocol – any established set of rules that determines how data is transmitted between different devices is considered a network protocol

Algorithm – a set of rules or a process which needs to be followed to solve a specific problem commonly used in calculations

e.g. an algorithm used to compress a file – which set of rules are being used to decide how this file is being compressed

Packet – Fragmentation of data to be sent across the network

Payload – The actual data within the packet excluding metadata

Remote Server

SSH – It is a cryptographic network protocol which means a standard set of rules that is being used to transfer encoded packets over the network.

Introduced in 1995 by Tatu Ylonen

Developed to solve security problems faced by tools such as: Telnet, Rlogin, RSH, Std username/password

Raw packets could be extracted, put together by hackers to get your credentials

What is SSH solving?

1. Being able to connect to a remote server in a very secure manner

SSH uses asymmetric cipher for encryption and decryption of packets

Symmetric cipher uses the same key to encrypt/decrypt data

Asymmetric cipher uses a key pair. Requires both private and public key to encrypt and decrypt data. How the encryption and decryption happens depends on the algorithm that is being used to do the encryption

With symmetric cipher, cannot send key across the internet

There are many encryption algorithms such as: RSA, DSA, ED25519

How SSH uses asymmetric cipher to

1. Establish connectivity between client and server
2. Send encrypted data across the channel
3. A key pair has to exist

The client shares the public key with the remote server

Public key can be shared with anyone

Client has the private key

Remote server has to be sure that the client is who he actually claims that he is

Once server receives connection request from the client, it uses the public key that it has to encrypt data.

Client receives the data and can decrypt data and sends decrypted text to server

Client and server establish trust

SSH creates a tunnel between client and remote server for data exchange

Use tools such as Putty, Seguin, Git bash or powershell to use ssh

Disadvantages or SSH

1. Any misconfiguration of SSH can affect you from ever being able to login to the system
2. If you private key gets lost