📇

Launching a Virtual Machine

⬇ Type	📙 Lecture	
🗓 Date	@December 22, 2021	
≡ Lecture #	1	
⊘ Lecture URL	https://youtu.be/PHrN7yp1AJw	
⊘ Notion URL	https://21f1003586.notion.site/Launching-a-Virtual-Machine-e493cecce6a743a9b37516d196c07c1d	
# Week #	1	

Why do we need a Virtual Machine (VM)?

- Laptops usually come pre-installed with Windows
    - Unless, of course, you are an 🍎 person
- We wish to try Linux almost natively, without removing the existing OS
    - Also considering the fact that we do not wish to dual boot

Requirements

- An .iso image of the operating system we want
    - Ubuntu 20.04 is recommended, or just use arch btw
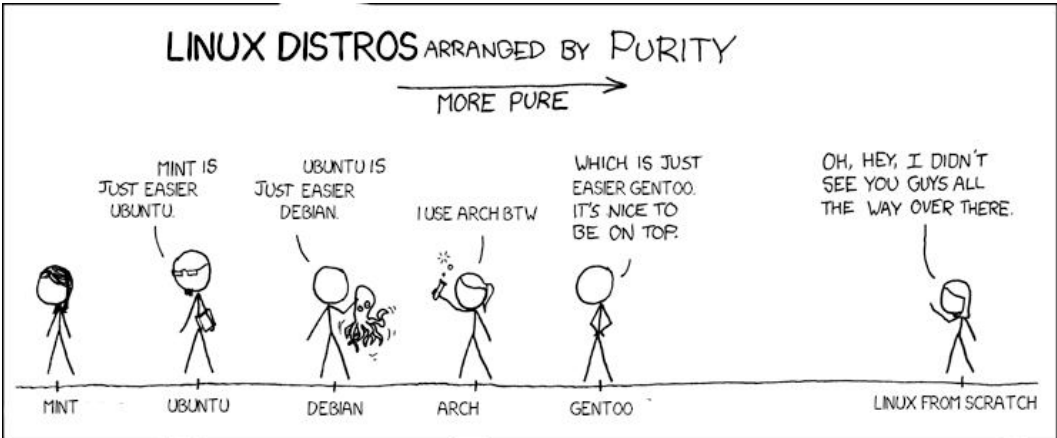
    - Can be downloaded here
- A Hypervisor
    - A **hypervisor**, also known as a virtual machine monitor or VMM, is software that creates and runs virtual machines (VMs). A hypervisor

allows one host computer to support multiple guest VMs by virtually sharing its resources, such as memory and processing.

Source: https://www.vmware.com/topics/glossary/content/hypervisor.html

- Oracle VirtualBox

- VMWare Workstation Player

- or just use Windows Subsystem for Linux

- Atleast 20GB free space for the VM

- Some RAM ¯_(ツ)_/¯

    - 8GB+ recommended

Steps
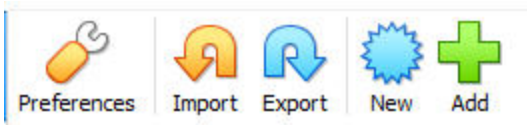
- Download the Ubuntu .iso file from https://ubuntu.com/download/desktop

- Download either VirtualBox or the VMWare Workstation Player
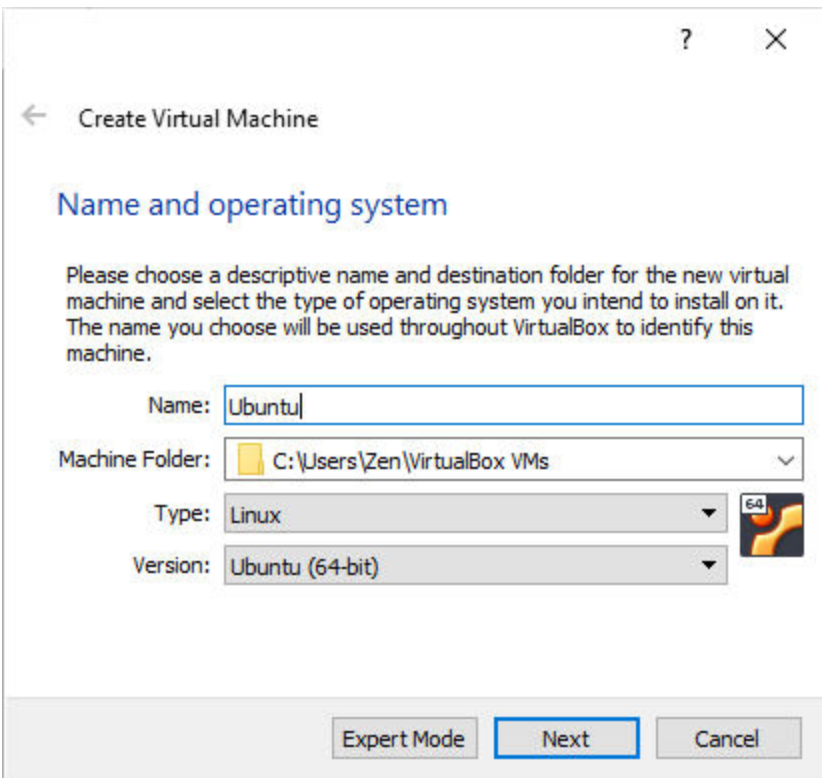
    - Install them

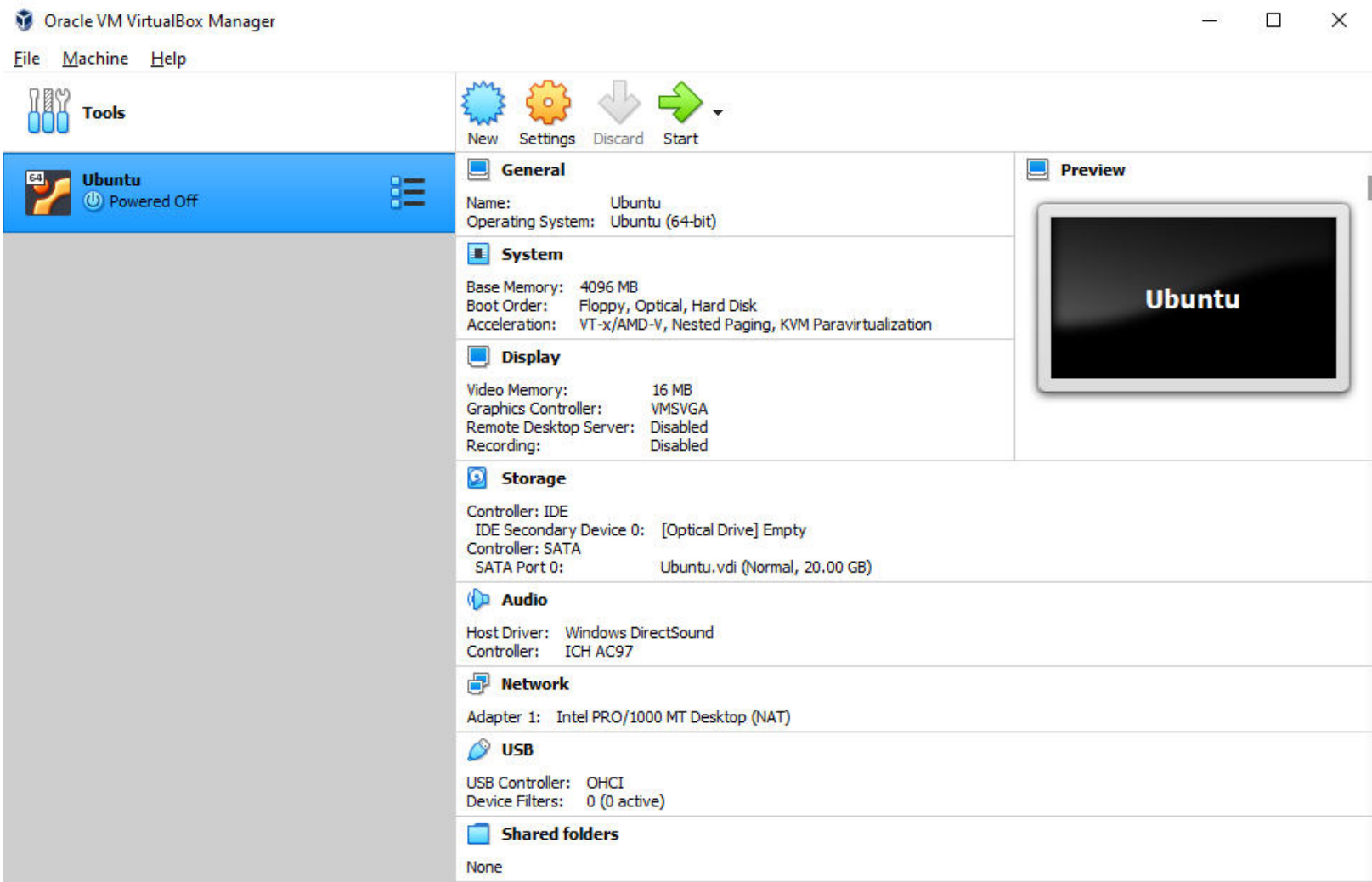- Open VirtualBox / VMWare Workstation Player

  (I will be using VirtualBox)

    - Click on the "New Button to create a new VM"
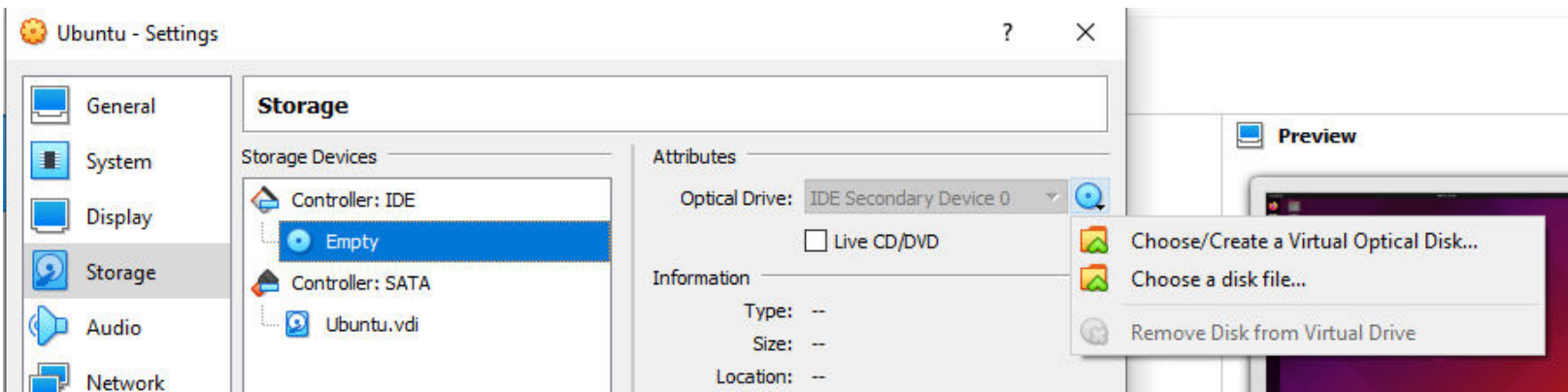

    - Add a name and choose the OS type and version

    - Adjust the RAM and Storage as per your liking, make sure to keep the minimum specs

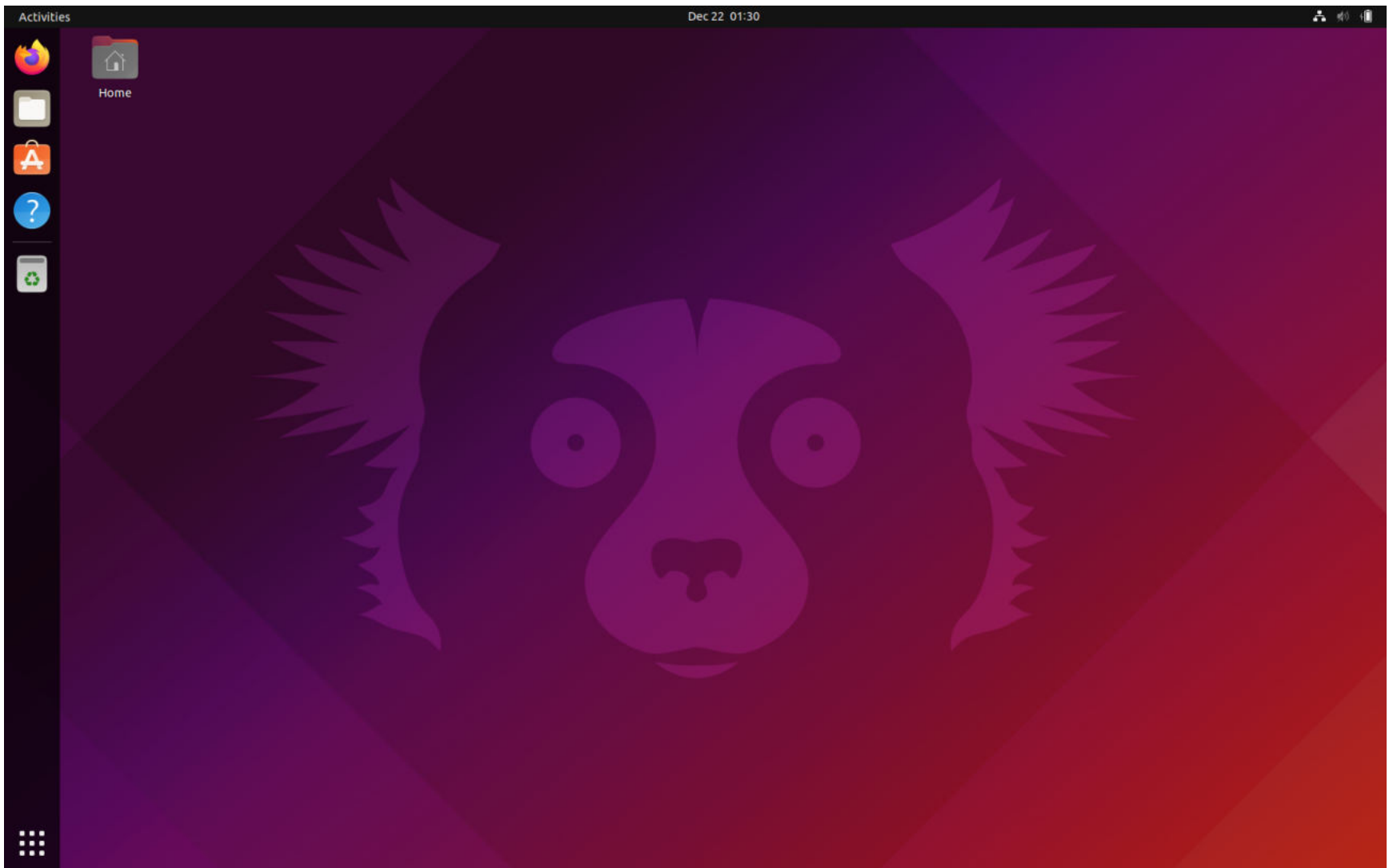   - Select the VM on the left menu and go to settings

- Go to Storage → Storage Devices → Controller → Empty → Click on the CD icon and choose "**Choose/Create a Virtual Optical Disk...**" and choose your .iso file

- Proceed with the installation
    - Uhh it's just Next, Next, Next .... Next ... Restart

When you install it correctly and get it up and running, you might see something like this ...

*(this screenshot here is Ubuntu 21.10 and gosh that's a creepy monke)*

Command Line Environment

⊙ Type	📙 Lecture
🗓 Date	@December 22, 2021
≡ Lecture #	2
⮂ Lecture URL	https://youtu.be/qrAnlpcMyYc
⮂ Notion URL	https://21f1003586.notion.site/Command-Line-Environment-f96a91e782a147a88894028d7848a2c4
# Week #	1

Why use Command Line environment?

- To use linux at its max potential
- Combine commands to form powerful scripts
    - To automate using these scripts
- *To assert dominance over GUI plebs*

Terminal in Ubuntu
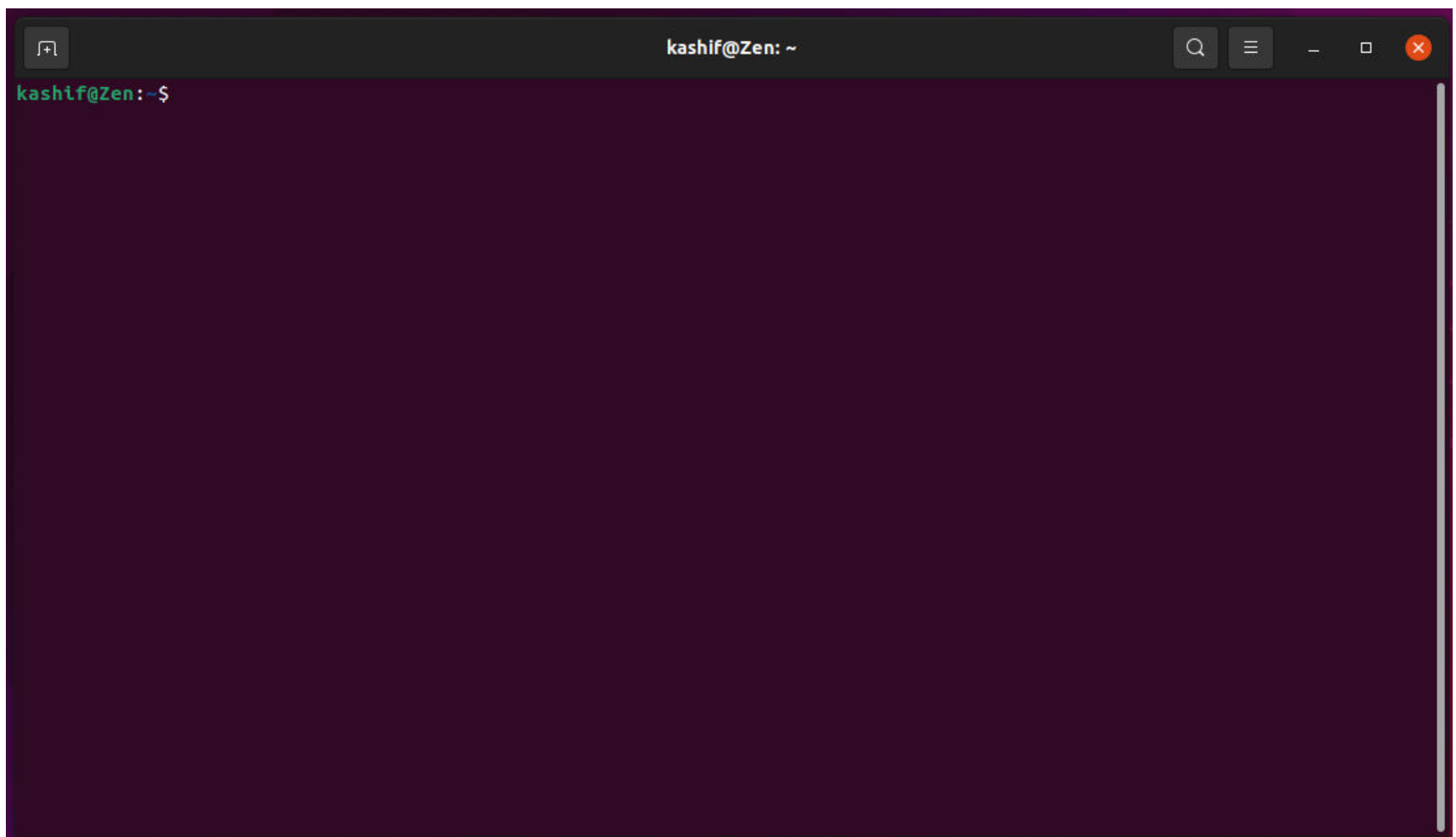
This is what the default terminal looks like in Ubuntu

To clear the command line

```
clear
```

- or you can press `Ctrl + L` to clear the terminal screen

To check which directory we currently are in

```
pwd
```

By default, you are placed in the home directory of the currently logged in user

To list all the files and folders in the current directory

```
ls
```

To view the currently running processes
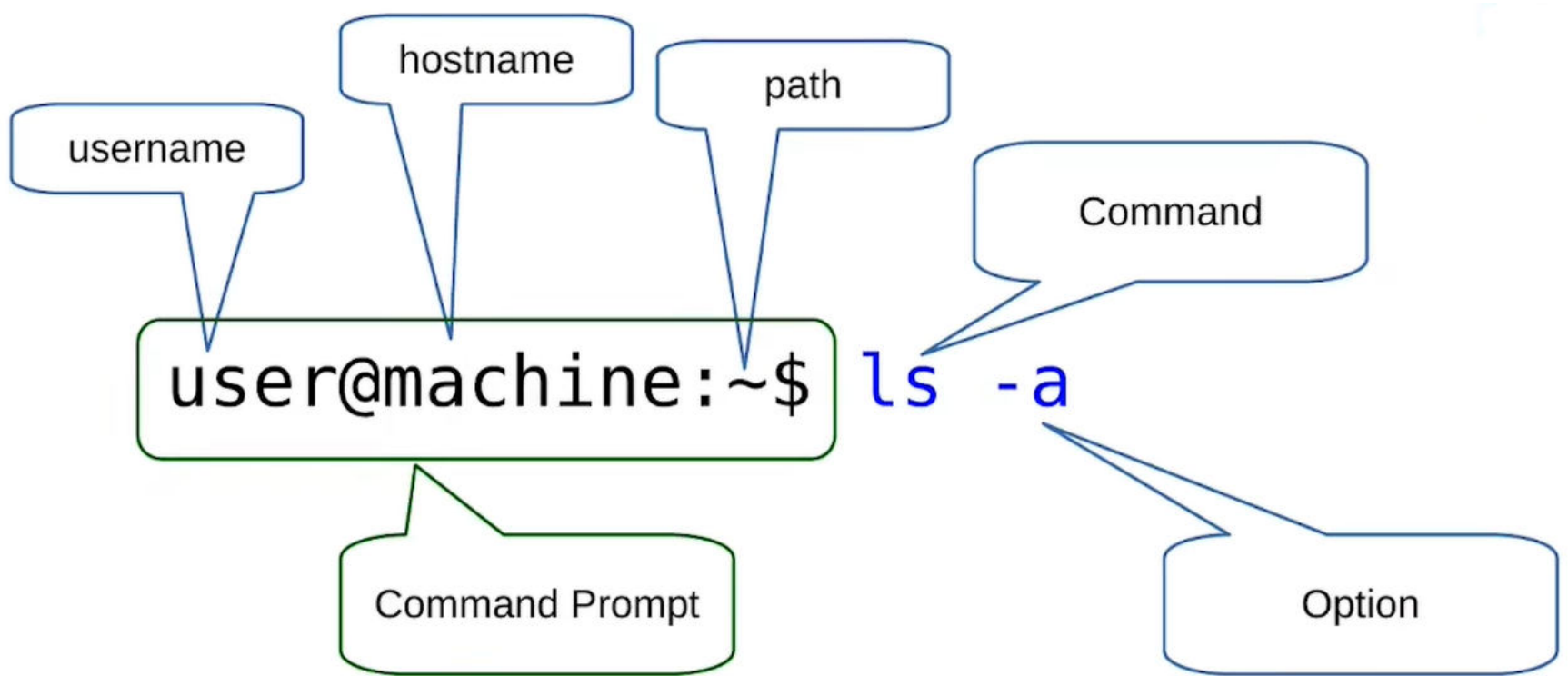
```
ps
```

To know the OS, duh

```
uname
```

To exit the shell

```
exit
```

- Or you can also press `Ctrl + D` to exit out of the terminal session

Anatomy of a typical command on the terminal

- To display all the files, press `ls -a`

- To display the files in a list, press `ls -l`

*These two flags are the most commonly used ones, **we can also combine them as one flag** like,* `ls -al`
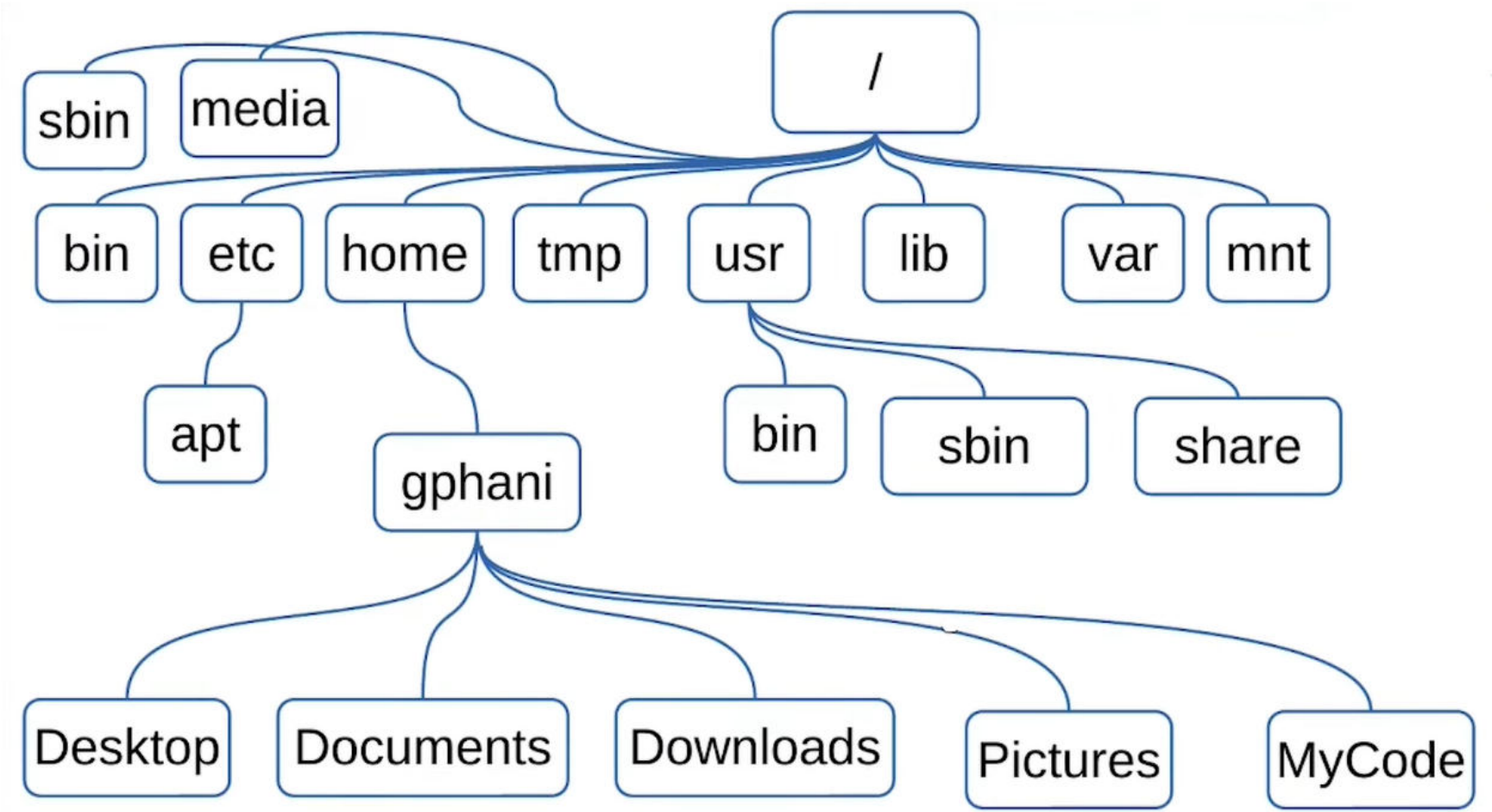
To get help on any command

```
man <command-name>
```

- For example

```
man ls
```

gives us the manual for the command `ls`

Filesystem in Linux

Traversing the tree

`/` is the root of the file system

`/` is also the delimiter for sub-directories

   `.` is the current directory

   `..` is the parent directory

Path for traversal can be absolute or relative

To change directory

```
cd <location>
```

Examples

- `cd` without any arguments will take us to the home directory of the currently logged in user
- `cd <folder-name>` will change the move us into the `folder-name`
- `cd ..` will takes us to the parent of the current directory
- `cd /` will takes us to the root directory

What does this *directory* do?

- `/bin` → Essential command binaries
- `/boot` → Static files for the bootloader
- `/dev` → Device files
- `/etc` → Host specific system configuration
- `/lib` → Essential shared libraries and kernel modules
- `/media` → Mount points for removable devices
- `/mnt` → Mount points
- `/opt` → Add on application software packages
- `/run` → Data relevant to running processes
- `/sbin` → Essential system binaries
- `/srv` → Data for services
- `/tmp` → Temporary files
- `/usr` → Secondary hierarchy
- `/var` → Variable data

`/usr` hierarchy

- `/usr/bin` → User commands
- `/usr/lib` → Libraries
- `/usr/local` → Local hierarchy
- `/usr/sbin` → Non-vital system binaries
- `/usr/share` → Architecture dependent data
- `/usr/include` → Header files included by C programs
- `/usr/src` → Source code

`/var` hierarchy

- `/var/cache` → Application cache data
- `/var/lib` → Variable state information
- `/var/local` → Variable data for `/usr/local`
- `/var/lock` → Lock files

- `/var/log` → Log files and directories

- `/var/run` → Data relevant to running processes

- `/var/tmp` → Temporary files preserved between reboots

	sharable	unsharable
static	/usr /opt	/etc /boot
variable	/var/mail	/var/run /var/lock

![server icon]

Simple Commands in Linux - 1

⊙ Type	📔 Lecture
📅 Date	@December 22, 2021
≡ Lecture #	3
↻ Lecture URL	https://youtu.be/DIpBEmRDHnw
↻ Notion URL	https://21f1003586.notion.site/Simple-Commands-in-Linux-1-3e5b2c25e6d04a7499afcb89af041b20
# Week #	1

Some basic commands

- `date` → Date and time

    - `date -R` → Gives the date in RFC5322 standard
- `cal` → Calendar of a month

- `free` → Memory statistics

    - `free -h` → Makes the output human readable
- `groups` → Groups to which the user belongs

```
kashif@Zen:~/Desktop$ groups
kashif adm cdrom sudo dip plugdev lpadmin lxd sambashare
```

idk what the junk is this

- `file` → What type of a file it is

```
kashif@Zen:~$ file .bashrc
.bashrc: ASCII text
```

- `cd -` → To visit the previous directory we were in

```
kashif@Zen:~$ cd -
/home/kashif/Desktop
kashif@Zen:~/Desktop$
```

Typical output of `ls -l`
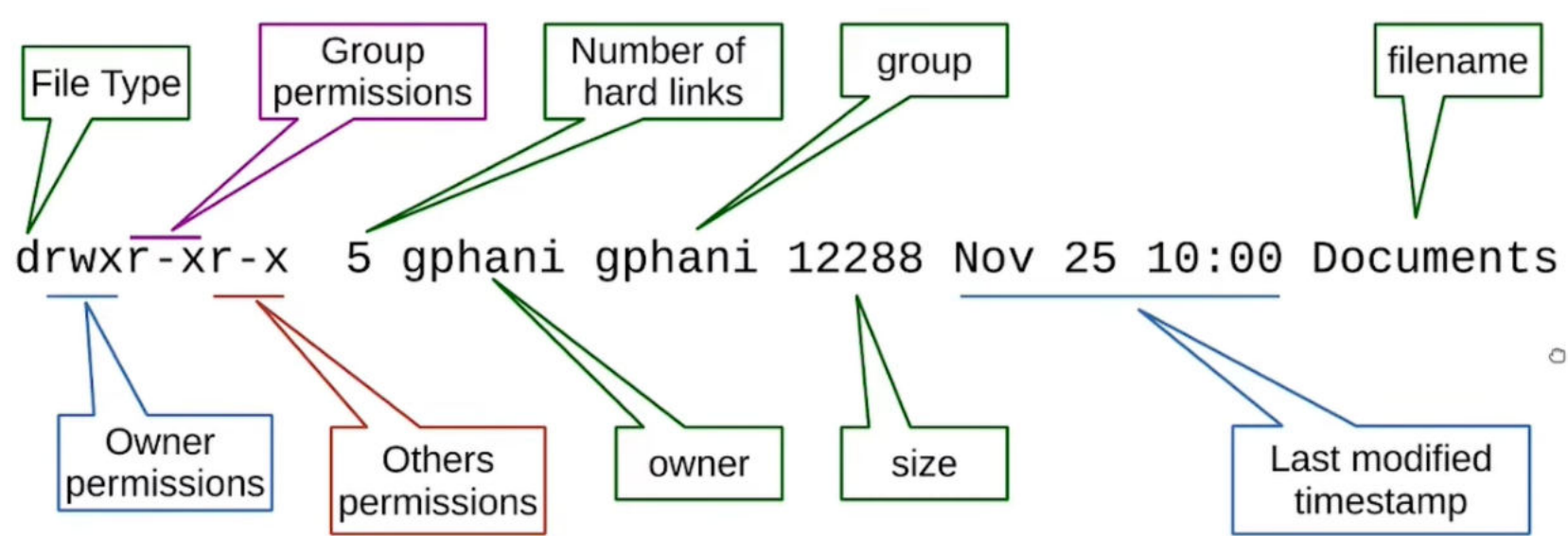
File types

- `-` → Regular file
- `d` → Directory
- `l` → Symbolic link
- `c` → Character file
- `b` → Block file
- `s` → Socket file
- `p` → Named pipe

inode (Read: eye node)

`ls -l <name>`

- An entry in the filesystem table about the location in storage media

Permission string

7 (rwx)
6 (rw-)
5 (r-x)
4 (r--)
1 (--x)

2. Group permissions

rwxr-xr-x

755

1. Owner Permissions
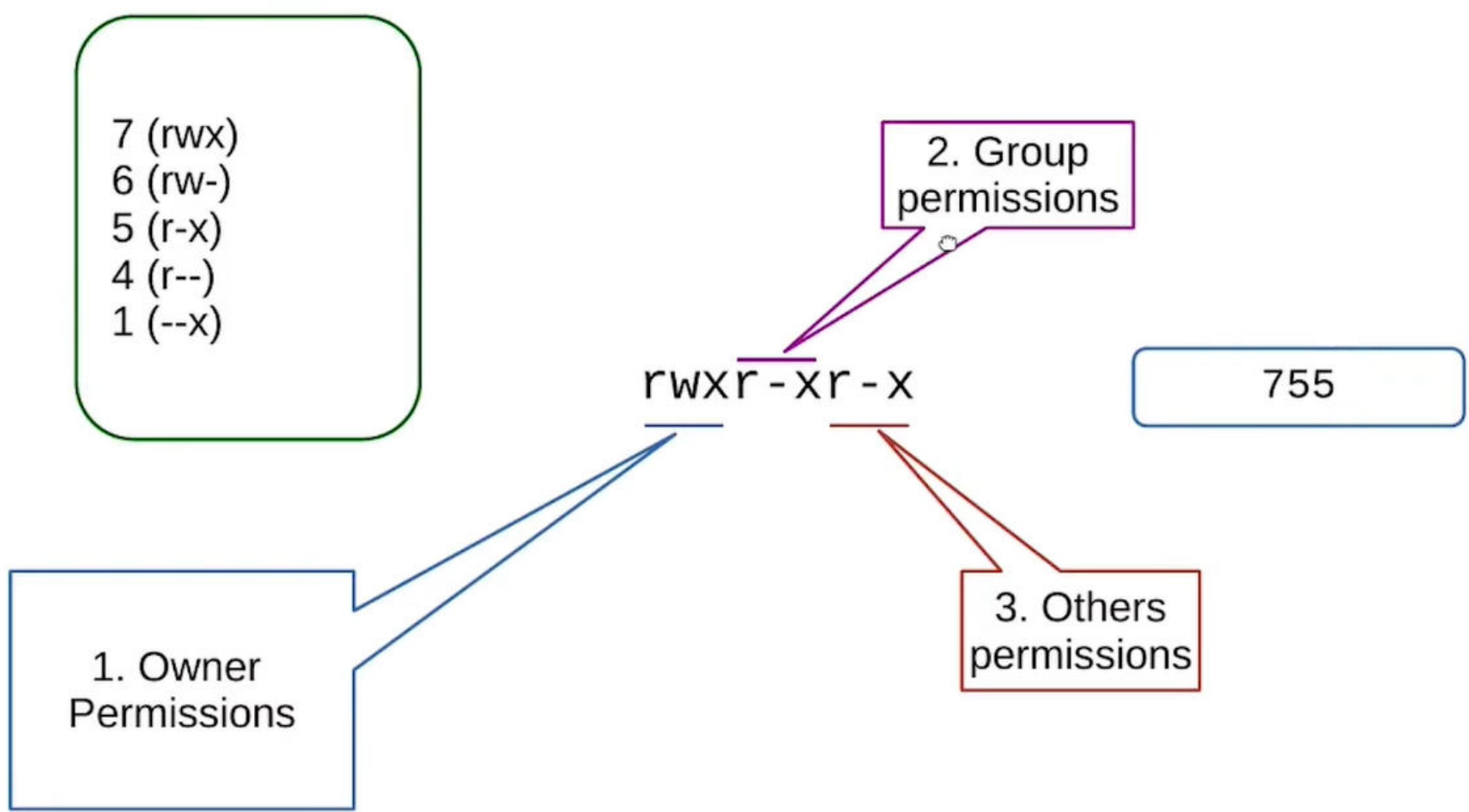
3. Others permissions

To modify permissions

- Create a folder

  - `mkdir <folder-name>`

  - `chmod g-w <folder-name>` to remove the write permission from the group

  - Similarly, `chmod g-x <folder-name>` to remove the execute permission from the group

  - To add permission, `chmod g+w <folder-name>` to give write permission to the group

- So, a general structure of permission syntax is something like ...

  - `chmod <user-group><plus/minus><r/w/x> <folder-name/file-name>`

    - Where `<user-group>` are ...

      - `u` → User

      - `g` → Group

      - `o` → Others

    - `<plus/minus>` are ...

      - `-` → To remove permission

      - `+` → To add permission

    - `<r/w/x>` are ...

      - `r` → Read

      - `w` → Write

      - `x` → Execute

- We can also use numerical values for permissions

  - `chmod 700 <folder-name>` to give the `rwx` permission to user only

`touch` command

- Used to modify the timestamp of a file or folder

  - If a file does not exist, it will be created

- `touch <file-name>` to create a new file

  - `chmod 700 <file-name>` to give `rwx` permission to user only

`cp` command

- `cp <file-name> <new-name>` to copy a file to a new name
  - `cp <file-name> <new-path>` can be used to copy a file to a new path

`mv` command

- `mv <file-name> <new-path>` to move a file to a new path
  - `mv <file-name> <new-file-name>` can be used to rename a file

Also, use quotation marks if the file name includes a space

`rm` command

- `rm <file-name>` to remove a file
  - **IT WILL NOT ASK FOR YOUR CONFIRMATION**
    - Just straight up delete

  - This is the default behaviour
  - We can pass `-i` flag for the confirm remove prompt

Alias

- We can also set an alias for long commands, for example ...
  - `alias ll="ls -altrhF"`

Know current user

```
whoami
```

Read a text file, page-by-page

```
less <file-name>
```

To know the type of a file

```
file <file-name>
```

Some commands

- `chmod` → Change permissions of a file
- `touch` → Change modified timestamp of a file
- `cp` → Create a copy of a file
- `mv` → Rename/Move a file
- `mkdir` → Create a directory
- `rm` → Remove a file