# Auto-mount disk partition

I am using **Ubuntu 22.04** and **Windows 10** on **dual boot** mode. Only 40 GB of SSD is assigned to Ubuntu and the rest 256 GB of SSD is assigned to Windows along with 1 TB of HDD( which are logically divided in 5 partitions, 1 of which contains system files probably).
Now, I want full access to the partitions of HDD from Ubuntu too, i.e. I want to also mount the HDD partitions on Ubuntu. In short, I want to share HDD between both Windows and Ubuntu.

# Mount non-assigned disk partition to Linux user

## 1. List available partitions

At first, find out the partitions present in the storage devices.

```
lsblk
```

**N.B:**This command shows all the partitions and/or devices (both mounted and unmounted) physically present in the computer.

In my case, ***sda2, sda3, sda4, sda5*** are the partitions that I want to mount. `nvme1p0n3` contains Windows `C:/` drive and `nvme1p0n5` contains Ubuntu `root` directory.

## 2. Create directory for mounting partitions or disks

Now, create mounting points for the partitions. The mounting point must be in `/media` directory. The command is:

```
sudo mkdir /media/username/mounting_point_name
```

**N.B:** `/media` directory contains the **mounted** partitions and/or drives (CD, USB, HDD, SSD, any memory device)

I will create 4 directories in the `media/naeem/` directory of `root`/`/` for mounting the partitions, as I wish to mount 4 partitions.

```
#!/bin/bash
sudo mkdir /media/naeem/Personal
sudo mkdir /media/naeem/study
sudo mkdir /media/naeem/Apps
sudo mkdir /media/naeem/Etc
```

## 3. Mount the partitions on the mounting points

To mount the partitions to the directory, the command structure is :

```
sudo mount /dev/unmounted_partition /media/username/mounting_point
```

**N.B:** Note the `/dev` part of the of the partition address. `lsblk` doesn't include this part when showing the partitions. Include it in the command as it is part of absolute address.

I will mount 4 partitions to their intended directories. The command used is:

```
sudo mount /dev/sda2 /media/naeem/Personal
sudo mount /dev/sda4 /media/naeem/study
sudo mount /dev/sda5 /media/naeem/Apps
sudo mount /dev/sda3 /media/naeem/Etc
```

# Limitation

The mounting is temporary and partitions will be unmounted every time the computer shuts down.

# Bypassing limitation: Making the mounting appear permanent

By running the mounting portion of code every time the computer boots, we can make the mounting of the partitions to appear to be permanent. We can use `crontab` to run a script every time PC boots.

Firstly, create a bash file (let's name it `mount.sh`) with shebang/ `#!bin/bash`, (to avoid typing `bash` in terminal command) with the mounting commands in a directory.

So, I will create the file in `/usr` directory and edit it with `nano` editor. To do that, I will need `sudo` permission as alternating anything outside `/home` directory needs ***super user permission***. So, write

```
sudo touch /usr/mount.sh
sudo nano /usr/mount.sh
```

**N.B:** Devian based distributions face problem when the script or bash file is in user's `/home/` / `~/` directory. Apparently, `crontab` runs even before `/home` is mounted, so, if the script is in home, it will not be executed. That's why file should be in other directory than `~`.
in terminal and write the commands below on that bash file from terminal's nano editor. (Note that any activity ***not*** in home directory needs `sudo` permission, and mounting is done in `/media` directory) .

```
#!/bin/bash
sudo mount /dev/sda2 /media/naeem/Personal
sudo mount /dev/sda4 /media/naeem/study
sudo mount /dev/sda5 /media/naeem/Apps
sudo mount /dev/sda3 /media/naeem/Etc
```

Secondly, give execution permission to the bash file. To do it, open terminal from `mount.sh`'s directory or `cd` to there. Then type

```
sudo chmod +x mount.sh
```

Thirdly, enter the command below to the terminal.

```
sudo crontab -e
```

Now, navigate to the bottom of the opened file and add the absolute path of `mount.sh` after typing `@reboot` in a newline.

```
# some command lines that might be already present.
# ...
@reboot /usr/mount.sh
```

Save and exit the terminal. Now, I should be good to go.

# Conclusion

The bash file *mount.sh* is executed every time Ubuntu is turned on. So, mount.sh mounts the drive partitions every time Ubuntu is booted by using `crontab`.

## Alternatives

This auto execution of commands on startup can also be done by adding the bash file path to the end of `/etc/rc.local` file or by using `/etc/init.d` file. `~/.bashrc` was also suggested, but apparently, it executes every time terminal is opened, not when Ubuntu is booted.