Identify and Mount a Drive

This guide assumes you only have 1 external hard drive connected to the Pi. If so then it should be attached to /dev/sda1 – additional drives will use /dev/sdb1 and /dev/sdc1 etc. If you have multiple external hard drives you will need separate mount points for each drive (e.g. /mnt/usbstorage1 and /mnt/usbstorage2).

```
  Prepare the Mount Point
  First make a directory in which to mount the USB drive
  sudo mkdir /mnt/usbstorage
```

```
  Make pi the owner of the mounted drive and make its permissions read, write and execute for it
  sudo chown -R pi:pi /mnt/usbstorage
  sudo chmod -R 775 /mnt/usbstorage
```

```
  Set all future permissions for the mount point to pi user and group.
  sudo setfacl -Rdm g:pi:rwx /mnt/usbstorage
  sudo setfacl -Rm g:pi:rwx /mnt/usbstorage
```

```
  Determine the USB Hard Drive Format
```

```
  You also need to know the file system the drive is formatted with
  sudo blkid
```

```
  You will see something like this. Again it is the sda1 line we are interested in. Note the
TYPE="exfat" at the end, you will need this for the fstab file. It is easiest to copy it now and paste
it after running nano in the next step.
  /dev/sda1: UUID="31db83ca-ac6d-4bc7-b076-848c7c71025e" TYPE="ext2" PARTUUID="279bf5b4-01"
  /dev/mmcblk0: PTUUID="0006dd3f" PTTYPE="dos"
  /dev/mmcblk0p1: LABEL="RECOVERY" UUID="0403-0201" TYPE="vfat" PARTUUID="0006dd3f-01"
  /dev/mmcblk0p5: LABEL="SETTINGS" UUID="705f6e2b-fac6-4f33-8611-d57a9c9f04e1" TYPE="ext4"
PARTUUID="0006dd3f-05"
  /dev/mmcblk0p6: SEC_TYPE="msdos" LABEL="boot" UUID="1495-189B" TYPE="vfat" PARTUUID="0006dd3f-06"
  /dev/mmcblk0p7: LABEL="root0" UUID="759bca6b-5766-4941-b830-cdbfcd861107" TYPE="ext4"
PARTUUID="0006dd3f-07"
  /dev/mmcblk0p8: LABEL="boot-rbp2" UUID="200C-EA5B" TYPE="vfat" PARTUUID="0006dd3f-08"
  /dev/mmcblk0p9: LABEL="root-rbp2" UUID="26d10fa3-fe0a-4044-b24a-9b85c2079122" TYPE="ext4"
PARTUUID="0006dd3f-09"
```

```
  Update your repositories if your hard drive is anything but ext4 as the TYPE above
  sudo apt-get update
```

```
  Now mount the usb stick in there. If it is NTFS you will need to install some utilities first
  sudo apt-get install ntfs-3g -y
```

```
  If the drive is exfat install these utilities
```

```
  sudo apt-get install exfat-utils -y
```

```
  For all drive types mount the usb with this command, -o insures pi is the owner which should avoid
permission issues
```

```
  sudo mount -o uid=pi,gid=pi /dev/sda1 /mnt/usbstorage
```

```
  If you get an error use this syntax
  sudo mount -t uid=pi,gid=pi /dev/sda1 /mnt/usbstorage
```

```
  If the mount -t command returns an error then use this syntax
  sudo mount uid=pi,gid=pi /dev/sda1 /mnt/usbstorage
```

```
  If you are getting this drive is already mounted errors then you are probably using a distro which automounts the drives which you can either continue using but then you should remove the /etc/fstab entries. You will have to uninstall the automounting software if you want to mount using the method in this guide.
  Remove the automounting software with this command
  sudo apt-get remove usbmount --purge
```

```
  Automount the USB Hard Drive on Boot
```

```
  /mnt/usbstorage will be the folder in which you store your media. We want it to be automounted on boot The best way to do this is through the UUID. Get the UUID by using this commmand
```

sudo ls -l /dev/disk/by-uuid/

```
  You will see some output like this. The UUID you want is formatted like this XXXX-XXXX for the sda1 drive. If the drive is NTFS it can have a longer format like UUID="BABA3C2CBA3BE413". Note this UUID, for me it is BA8F-FFE8
```

total 0

```
  lrwxrwxrwx 1 root root 15 Jan  1  1970 3d81d9e2-7d1b-4015-8c2c-29ec0875f762 -> ../../mmcblk0p2
  lrwxrwxrwx 1 root root 15 Jan  1  1970 787C-2FD4 -> ../../mmcblk0p1
  lrwxrwxrwx 1 root root 10 Oct 26 21:10 BA8F-FFE8 -> ../../sda1
```

```
  Now we will edit fstab to mount the USB by UUID on boot
  sudo nano /etc/fstab
```

```
  Add the line in red to the bottom, replace XXXX-XXXX with your UUID and exfat with your type if it is different (e.g. ntfs, vfat, ext4). You may or may not need the quotation marks wrapped around the UID, you do not need quotation marks wrapped around the file system type (ext4, vfat, NTFS etc).
```

```
  The umask 0002 sets 775 permissions so the pi user and group can read, write and execute files on the external USB drive. To completely eliminate permission issues you can set the umask to 0000 which equals 777 permissions so anybody can read, write and execute. Note that 777 permissions are considered a security risk.
```

If you have issues here then try replacing uid=pi,gid=pi with just the word defaults (typical for ext4). You can also try replacing the UUID with the /dev/sda1 line.

```
  This is an example for exfat
  /dev/mmcblk0p1 /boot vfat defaults 0 2
  /dev/mmcblk0p2 / ext4 errors=remount-ro,noatime 0 1
  UUID=XXXX-XXXX  /mnt/usbstorage exfat   nofail,uid=pi,gid=pi   0   0
  For NTFS, note that it is ntfs and not ntfs-3g
  /dev/mmcblk0p1 /boot vfat defaults 0 2
  /dev/mmcblk0p2 / ext4 errors=remount-ro,noatime 0 1
  UUID=XXXX-XXXX    /mnt/usbstorage    ntfs   nofail,uid=pi,gid=pi    0   0
```

```
  For ext4 using uid and gid is not recommended so use at your own risk as it could cause issues (thanks
mk2soldier).
  /dev/mmcblk0p1 /boot vfat defaults 0 2
  /dev/mmcblk0p2 / ext4 errors=remount-ro,noatime 0 1
  UUID=XXXX-XXXX    /mnt/usbstorage    ext4   nofail,uid=pi,gid=pi    0   0
```

```
  If you get any errors you can replace uid=pi,gid=pi with defaults or remove it entirely
  /dev/mmcblk0p1 /boot vfat defaults 0 2
  /dev/mmcblk0p2 / ext4 errors=remount-ro,noatime 0 1
  UUID=XXXX-XXXX    /mnt/usbstorage    ext4   nofail,defaults    0   0
```

```
  For using /dev/sda1 and defaults if you have troubles with UUID
  /dev/mmcblk0p1 /boot vfat defaults 0 2
  /dev/mmcblk0p2 / ext4 errors=remount-ro,noatime 0 1
  /dev/sda1    /mnt/usbstorage    ext4   nofail    0   0
```

```
  Now test the fstab file works
  sudo mount -a
```

```
  If you didn’t get errors reboot, otherwise try the suggestions above to get it working then mount -a
again until it succeeds
```

```
  sudo reboot
```

You should be able to access the mounted USB drive and list its contents cd /mnt/usbstorage ls

Every time you reboot, the drives will be mounted as long as the UUID remains the same. If you delete the partitions or format the USB hard drive or stick the UUID changes so bear this in mind. You can always repeat the process for additional hard drives in the future.

Now you can manage the hard drive power using these guides since it will not spin down automatically on Linux

If you have multiple hard drives you will have to make separate mount points (e.g. /mnt/usbstorage2) for each drive’s partition

Fix Raspberry Pi 2 Mounting Issues

Thanks to Jake for bringing this to my attention. Apparently there is a bug in the Pi 2 that messes up automounting. You can fix it by creating a delay.

Open up the /boot/cmdline.txt file sudo nano /boot/cmdline.txt

Add this line to the bottom, you can increase this delay if necessary rootdelay=5

Hit Ctrl+X, Y and Enter to save and exit, then reboot to see if it automounts now. If the Raspberry Pi hard drive still does not automount we can use rc.local (thanks Julian) sudo nano /etc/rc.local

Add this lines before the exit line sleep 30 sudo mount -a exit

Ctrl+X, Y and Enter to save

Reboot again to test sudo reboot

Identify and Mount a Drive

First we need to identify the disk(s):

```
  sudo blkid
```

This will list any recognised devices:

```
  /dev/mmcblk0p1: LABEL="RECOVERY" UUID="0403-0201" TYPE="vfat" PARTUUID="0006dd3f-01"
  /dev/mmcblk0p5: LABEL="SETTINGS" UUID="705f6e2b-fac6-4f33-8611-d57a9c9f04e1" TYPE="ext4"
PARTUUID="0006dd3f-05"
  /dev/mmcblk0p6: SEC_TYPE="msdos" LABEL="boot" UUID="1495-189B" TYPE="vfat" PARTUUID="0006dd3f-06"
  /dev/mmcblk0p7: LABEL="root0" UUID="759bca6b-5766-4941-b830-cdbfcd861107" TYPE="ext4"
PARTUUID="0006dd3f-07"
  /dev/mmcblk0p8: LABEL="boot-rbp2" UUID="200C-EA5B" TYPE="vfat" PARTUUID="0006dd3f-08"
  /dev/mmcblk0p9: LABEL="root-rbp2" UUID="26d10fa3-fe0a-4044-b24a-9b85c2079122" TYPE="ext4"
PARTUUID="0006dd3f-09"
  /dev/mmcblk0: PTUUID="0006dd3f" PTTYPE="dos"
  /dev/sda: PTUUID="279bf5b4" PTTYPE="dos"
```

In this example, the first 6 items are the SD card that Raspbian booted from **/dev/mmcblk0px**. The last device **/dev/sda** is a USB Hard Disk. This is the disk I want to add to Raspbian.

Now that we know the disk we wish to work on is /dev/sda we can use:

```
  sudo fdisk /dev/sda
```

```
  enter p to display partition information
  Disk /dev/sda: 298.1 GiB, 320072933376 bytes, 625142448 sectors
  Units: sectors of 1 * 512 = 512 bytes
  Sector size (logical/physical): 512 bytes / 512 bytes
  I/O size (minimum/optimal): 512 bytes / 512 bytes
  Disklabel type: dos
  Disk identifier: 0x279bf5b4
```

We can see the size is 298.1 GB.

```
  Use the d command to delete existing partition
  No partition is defined yet!
  Could not delete partition 81165
```

In this case, there are no partitions to delete
To create a new partition, use:

```
  n - This creates a new partition
  p - This is for a primary partition
  Enter - To default to partition 1
  Enter - To select first sector
  Enter - To select last sector.
```

You should now have a new partition.

```
  p - To display the new partition
  Device     Boot Start       End   Sectors   Size Id Type
  /dev/sda1        2048 625142447 625140400 298.1G 83 Linux
```

The changes need to be written to the partition table:

```
  w - To commit changes
  The partition table has been altered.
  Calling ioctl() to re-read partition table.
  Syncing disks.
```

Now run the following command to see your disk, which will now include /dev/sda1

```
  sudo fdisk -l
```

There will be a large output, but the important part is at the end:

```
  Device     Boot Start       End   Sectors   Size Id Type
  /dev/sda1        2048 625142447 625140400 298.1G 83 Linux
```

/dev/sda1 is the partition we have just created on device /dev/sda
Now we need to create the file system:

```
  sudo mkfs /dev/sda1
  mke2fs 1.42.12 (29-Aug-2014)
  /dev/sda1 contains a ntfs file system labelled '300gb'
  Proceed anyway? (y,n)    <---------------------------------You have to say 'Y' Here.
  Depending on drive size, this will take a minute or two
  Creating filesystem with 78142550 4k blocks and 19537920 inodes
  Filesystem UUID: 6af40af7-759f-4ee5-afea-882e9f58f17e
  Superblock backups stored on blocks:
          32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
          4096000, 7962624, 11239424, 20480000, 23887872, 71663616
  Allocating group tables: done
  Writing inode tables: done
  Writing superblocks and filesystem accounting information: done
```

After the superblocks are created and you get a command prompt, Now you are ready to mount your disk.
Lets create a mount point and call it NewDisk

```
  sudo mkdir /mydisk    <-- This creates a mount point (a folder) to mount our disk, the folder is
called mydisk
```

To Mount the Disk

```
  sudo mount /dev/sda1 /NewDisk    <--- bear in mind that your disk might not be sda1
```

Use df to verify disk is mounted. If you reboot you will need to remount it (you might want to add it to /etc/fstab)

```
  df
  Filesystem     1K-blocks    Used Available Use% Mounted on
  /dev/root        7928236 3577848   3924612  48% /
  devtmpfs          469544       0    469544   0% /dev
  tmpfs             473880       0    473880   0% /dev/shm
  tmpfs             473880    6520    467360   2% /run
  tmpfs               5120       4      5116   1% /run/lock
  tmpfs             473880       0    473880   0% /sys/fs/cgroup
  /dev/mmcblk0p6     64366   20436     43930  32% /boot
  tmpfs              94776       0     94776   0% /run/user/1000
  /dev/sda1      307665360   64344 291972508   1% /NewDisk    <---- here is our new disk
```

Try writing a file to the disk to test it:

```
  sudo touch /NewDisk/test
  ls /NewDsik
  lost+found  test
```