# Filesystem Administration

* Let’s revisit how we can add additional disks, filesystems, partitions, etc, to Linux
* We’ll start off with adding a secondary hard disk (5 GBs)
  + Open your VM settings
  + Graphical user interface, text, application

    Description automatically generated
  + Here, click on Add …, Add Hard Disk, SCSI, Create a New Virtual Disk, Maximum Size = 5GB:
    - Graphical user interface

      Description automatically generated
  + From there, reboot your Linux VM. (shutdown -r now)
* This disk, when I boot to Linux, will not be usable until I partition, format, and mount it.
* There are a couple of commands we can use to see our devices:
  + fdisk -l (lowercase “el”)
  + lshw -C disk
* fdisk will give you the following output:
  + Text

    Description automatically generated
    - /dev/sda 20 GiB
      * This is our initial disk we added when we installed the VM
      * Sda1,sda2,sda3
        + This has been divided into 3 partitions

1M, 1.8G, 18.2G respectively.

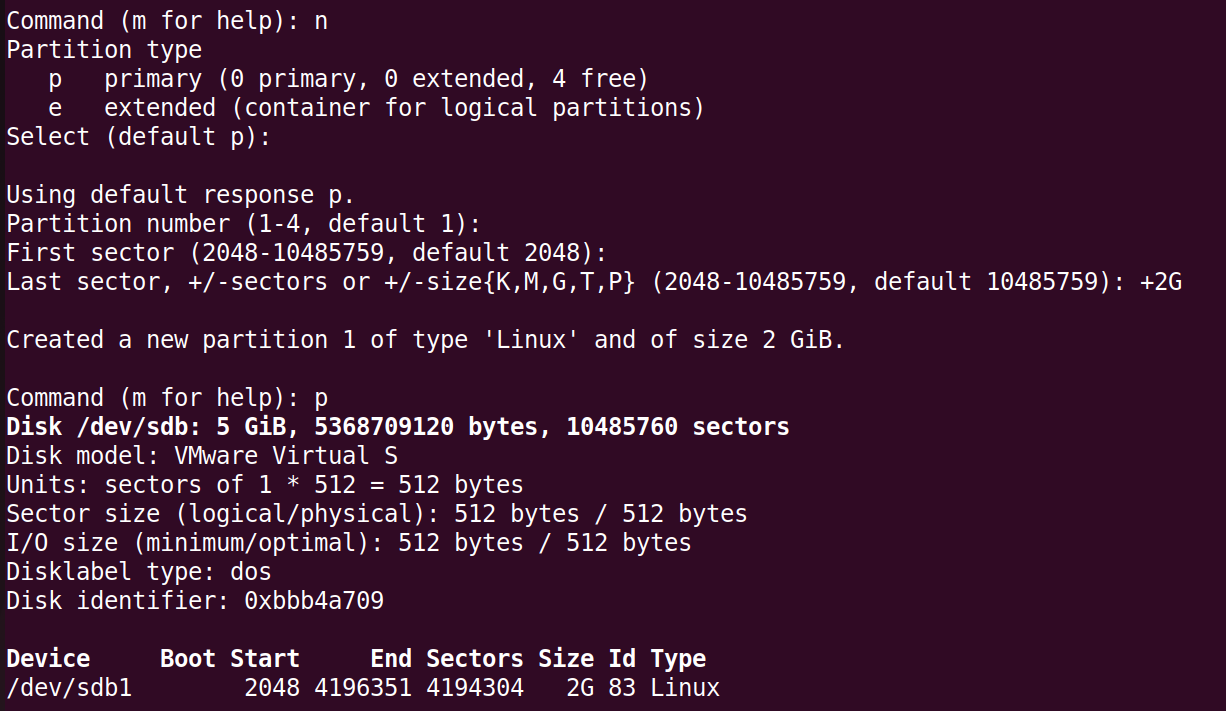
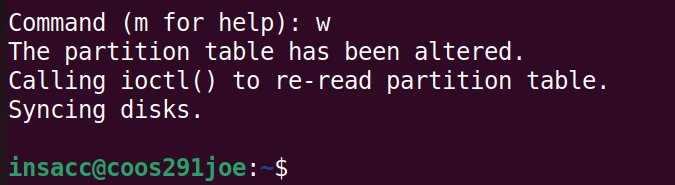
* + - /dev/sdb 5 GiB
      * This is our new disk we added just now.
      * No partitioning performed yet.
* Using “lshw -C disk”:
  + Text

    Description automatically generated
    - Contains additional information regarding the hardware of our devices.

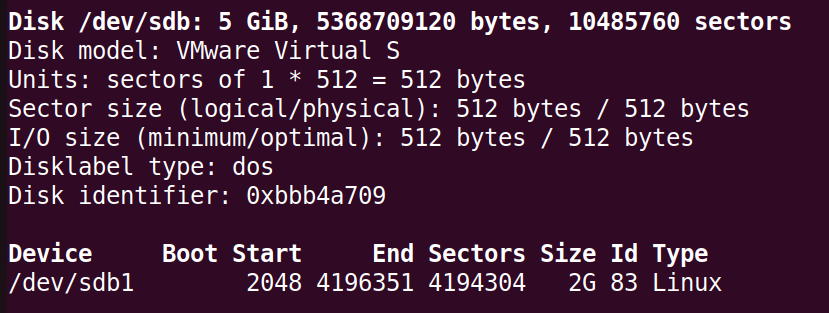
To create, remove and view partitions of the newly attached disk, run the command “fdisk <dev path>”

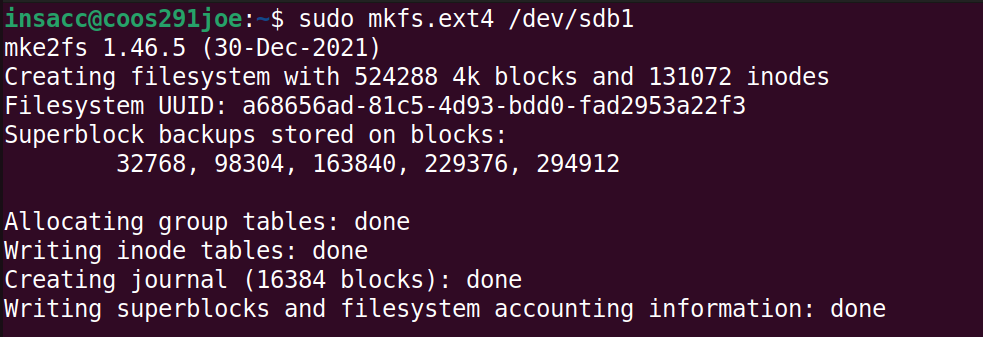
For example, to create a partition on sdb (the 5GB drive we added), run the following:

* sudo fdisk /dev/sdb
* Text

  Description automatically generated
* We can print out information regarding the disk via the “p” command
* We can create a new partition of size 2GB by using the “n” command:
* 
  + It asks for a partition type, we chose Primary (default)
  + It then asks for a partition number (default 1)
  + It then asks for first sector location, and gives us a range of available sectors (we chose the default)
  + Let section will dictate the size of the partition
    - If we don’t want to think in terms of sectors or Bytes, we can use the shorthand for other size types
    - In this case, we created a partition of 2GB on our 5GB drive:
      * +2G
* To save your changes, make sure you use the “w” command to write changes.
  + 

Running fdisk again (sudo fdisk -l):

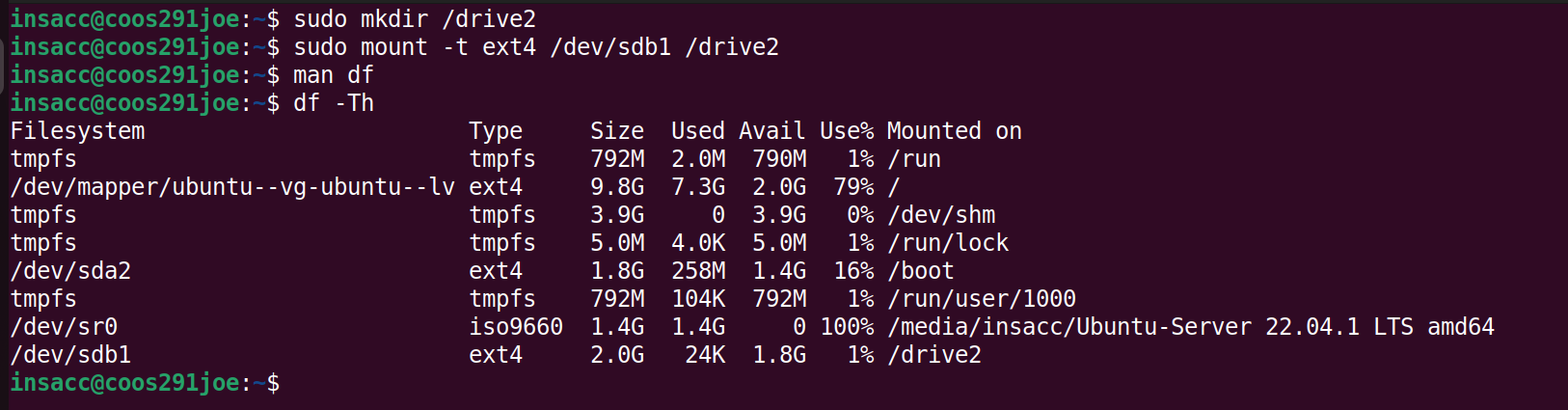


* We can see that we now have a 2GB partition called /dev/sdb1
* We’re still not ready to use it yet, we have to first format it to a particular file system type and mount it
* To format it, use the command mkfs.<whatevertype> (make file system) on sdb1
* 
  + When complete, mkfs will denote the number of blocks, the size of the blocks, and the number of inodes depending on those block sizes.
    - Also specifies the UUID or the unique identifier for the partition.
      * We’ll be using this later on to automatically mount this drive when the system boots.

Once we’ve created a filesystem (ext4) on our partition, we can go ahead and mount it.

Mounting a partition, or any filesystem in general, will give the user access to the filesystem through a directory.

To do this, we use the “mount” command.



Comments:

* Executing the following command:
  + sudo mount -t ext4 /dev/sdb1 /drive2
    - -t ext4 species the filesystem type that will be mounted
    - /dev/sdb1 is the partition we want to mount
    - /drive2 is a directory where we want to use in order to access the drive
* Text

  Description automatically generated
  + Use the “du -h” command to determine how much usage you have.

Try creating a file in /drive2 and copy to your host machine. Unfortunately, it doesn’t work as /drive2 has a filesystem of ext4 and most-likely you are using NTFS on your host.

Let’s create a new partition on sdb of size 2G, and format it to NTFS.

Text

Description automatically generated

It’s recommending we unmount. Use the command “sudo umount /dev/sdb1”:

A picture containing text

Description automatically generated

Comments:

* Notice on the second df command, our mount point is removed.
  + Our disk, partition, and filesystem still exist, it’s just not accessible (we unmounted it).

Let’s create our new partition now.

Text

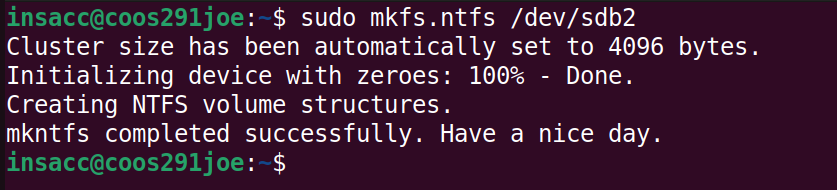
Description automatically generated

Using fdisk -l, we see /dev/sdb1 and /dev/sdb2

Text

Description automatically generated

Format that sdb2 partition to ntfs:



And then we can mount it.

A picture containing graphical user interface

Description automatically generated

Let’s try rebooting the Linux system and retype df -Th after a reboot:

A picture containing graphical user interface

Description automatically generated

Our mount is no longer there after a reboot.

There is a file called /etc/fstab (filesystem table) that we can edit so it’s automatically mounted on system boot:

A picture containing graphical user interface

Description automatically generated

This file contains the filesystem, the mount point, type, options, and any other information regarding our mounts.

We can simply add a new line for our new 2GB partition to be automatically mounted to our /ntfsdrive directory.

But first, we need the UUID for that partition. We can grab this by running the command **blkid <device>**

Text

Description automatically generated with medium confidence

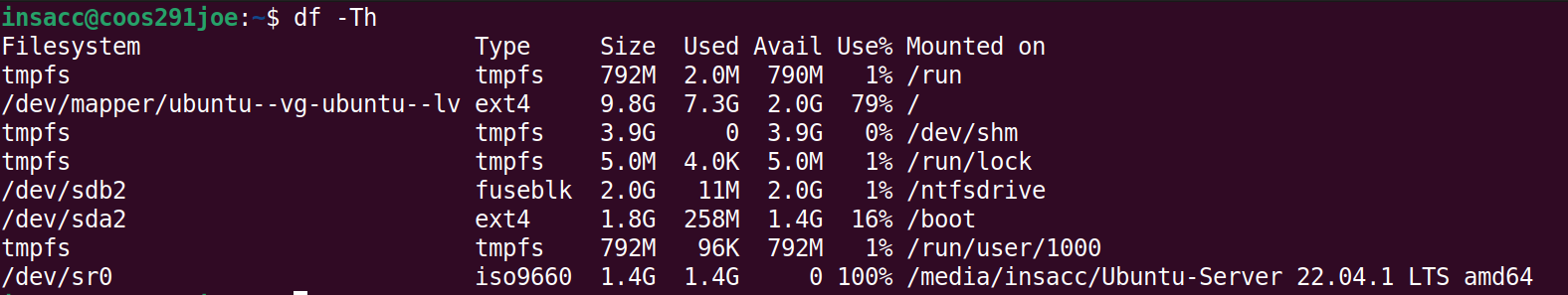
Comments:

* We need to copy the UUID so that we can add it our /etc/fstab file.
* First, let’s create a backup.

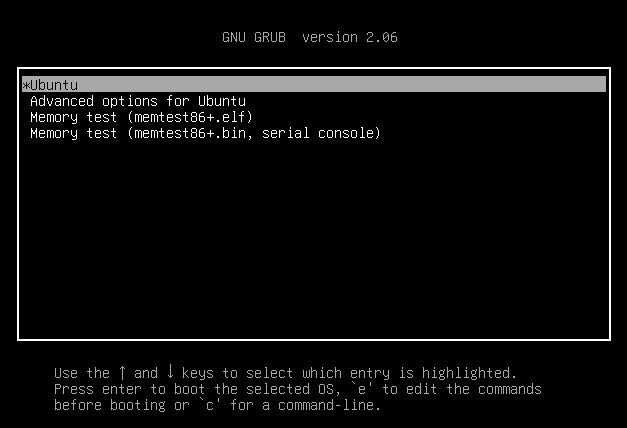
Text

Description automatically generated with medium confidence

Save your changes and reboot



# Booting into Single User Mode

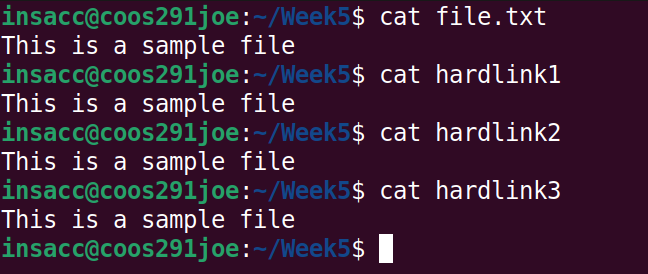
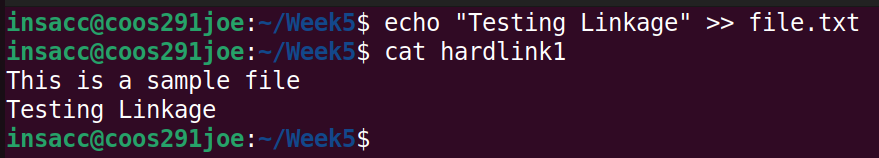
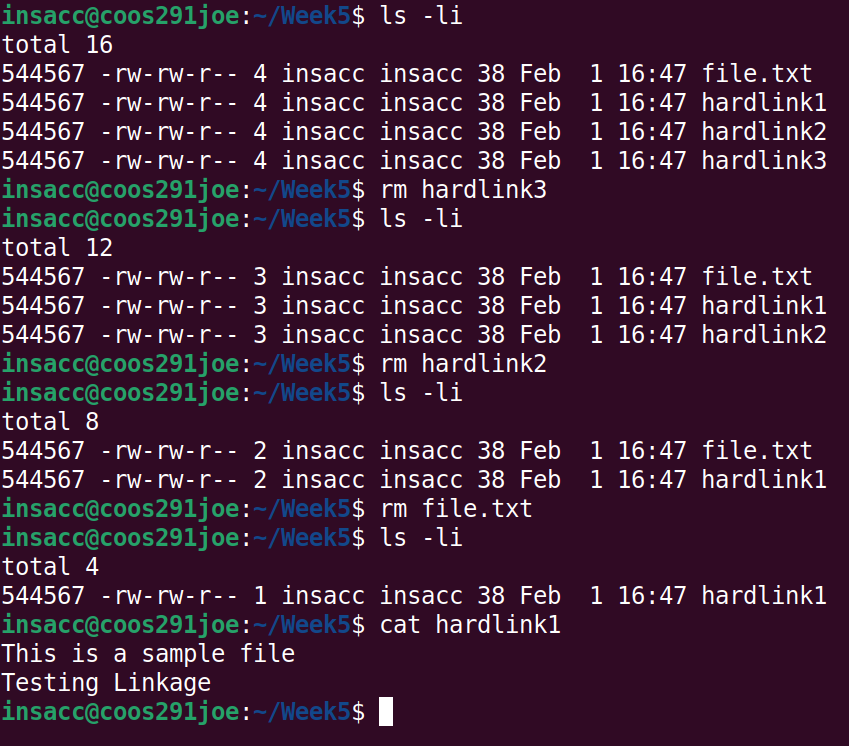
* There’s a special mode that can be entered from something called GRUB. To access GRUB, hold the “right-shift” key during initial boot.
* 
* You’ll be presented (in Ubuntu at least) with a screen that looks similar to that above
* Comments:
  + You’ll be presented with different options, depending on your install and distribution.
  + The first option, Ubuntu, is normal boot (we can edit this, as we’ll see soon).
  + The second option, Advanced Options for Ubuntu, is a boot option with different flag configured
  + The third and fourth options are memory tests, similar to what we ran in the hardware class.
* We can edit these entries by highlighting the desired entry and pressing the “e” key
* Normally, when the system boots, it does a bunch of tasks like checking the filesystem(s) for consistency, mounting filesystems, launching services, and otherwise getting all the necessary components up and running in order.
* Sometimes though, you’ll want to NOT do any of that and start up the bare operating system without any of that started – that’s what single user mode is for.
  + When you’re presented with the boot loader (GRUB), you can hit “e” key to edit.
  + Find where the operating system is started and then find where it specified a command called “ro” (read-only). This starts the filesystem in read-only mode so that it can check disk integrity.
  + We can change or add additional values to end of this line, such as the following:
    - rw \init=/bin/bash
    - This will open the operating system with read/write capability with a path to the shell that will be launched.
  + From here, we can check on the consistency of the filesystem, mount a filesystem to do some repair work, change the root password, etc.
* Text

  Description automatically generated

# Next class: Creating Links to files and directories.

## Creating Links to files and directories

* There are two types of links in Linux that you’ll use most likely:
  + Hard links
  + Symbolic (Soft) links
* A hard link is like an additional name for an existing file. Two or more file names are associated with the same inode.
* Syntax:
  + ln [OPTIONS] <source> <linked>
* Text

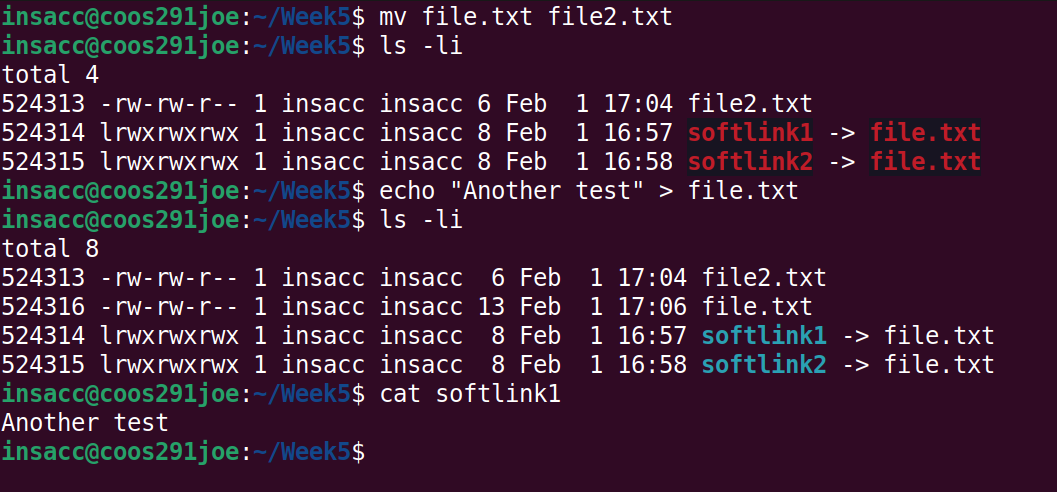
  Description automatically generated with medium confidence
* 
* 
* Comments:
  + ln -v file.txt hardlink1
    - Creates a hardlink to file.txt called hardlink1
    - -v is verbose mode (more detailed output)
  + ls -li
    - Does a (long) listing, with inode information.
  + Notice that when hardlink1 and hardlink2 are created, they have the same inode id (544567) as the original file.
    - This means that there is only one copy of the data (in the inode), but it has multiple file names pointing to it.
    - A change to the original file.txt will result in changes to any of the file names with a hard link on it.
* Removing links:
  + 
  + Comments:
    - Use the rm command to remove files (or links to other files)
    - Remove a hard link (rm hardlink2; rm hardlink3) does things normally in terms of file removal.
    - Removing the original source (file.txt), again has no restrictions.
    - Our first hardlink (hardlink1) still points to the original inode (544567)

## Symbolic Links (Soft Links)

* Symbolic Links are like shortcuts in Windows. It’s an indirect pointer to a file or directory. Can be used to point to a file or a directory on a different filesystem or partition.
* Again, we’ll use the **ln** command with the -s option to create a symbolic link.
* Text

  Description automatically generated
* Comments:
  + Use the command **ln -s <sourcefile> <linkedfile>**
    - This will create a symbolic link between source file and the linked file.
    - The links, if symbolic, will be contained on separate inodes.
      * 524313 (original file.txt)
      * 524314 (symbolic link softlink1)
      * 524315 (symbolic link softlink2)
* Text

  Description automatically generated with medium confidence
* Comments:
  + Editing the original file will have those changes reflected in all symbolic links.
  + Deleting the original file will result in dead symbolic links. Make sure to clean up after yourself.



Here, we can see that symbolic links don’t necessarily tie directly into inodes (just a creation). Renaming or moving files to the original source directory could potentially fix these links.

# Logical Volume Manager (LVM)

Problems with the regular way of partitioning our physical drives.

* Filesystem can only be as big as the disk partition you’ve got.
* Can’t resize partitions without wiping out your disk.
  + You can backup and restore
  + As a result, you need to know how big your partitions needs to be when you start out.

Because of these limitations, the Logical Volume Manager was created. With it, you can have one or more disks be treated as a single large disk, and you can resize partitions on the fly.

Reference Links:

<https://www.thegeekdiary.com/redhat-centos-a-beginners-guide-to-lvm-logical-volume-manager/>

<https://linuxhint.com/install_lvm_centos7/>

<http://www.tldp.org/HOWTO/LVM-HOWTO/>

Structure of LVM:

Diagram

Description automatically generated

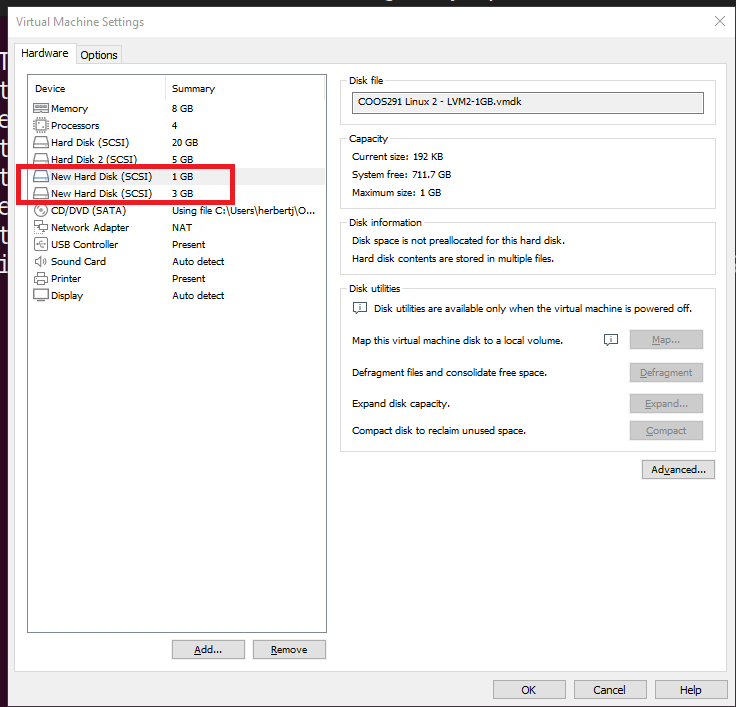
* Physical Volumes: A physical disk partition.
* Volume Groups: a grouping of physical volumes, from which logical volumes can be created.
* Logical Volume: A logical partition, may come from multiple physical devices (physical volumes). Filesystems are put on logical volumes and then mounted for access

Because a logical volume can come from multiple physical devices, you can have a filesystem larger than a single disk. For example, if you’ve got 4 1TB disks, without LVM, you are limited to a maximum filesystem size of 1 TB. With LVM, you can combine those disks, and you can have a maximum filesystem of 4 TB (4 x 1TB).

Often, LVM is set up when a system is initially installed. That way, you can change the size of any partitions (including your root partition) in addition to adding additional disk space as required.

Demo:

Add two more hard drives – one is 3GB, the other is 1GB. We’re going to use them to set up LVM. We’ll assume single partitions on each (for now).



Text

Description automatically generated

Text

Description automatically generated

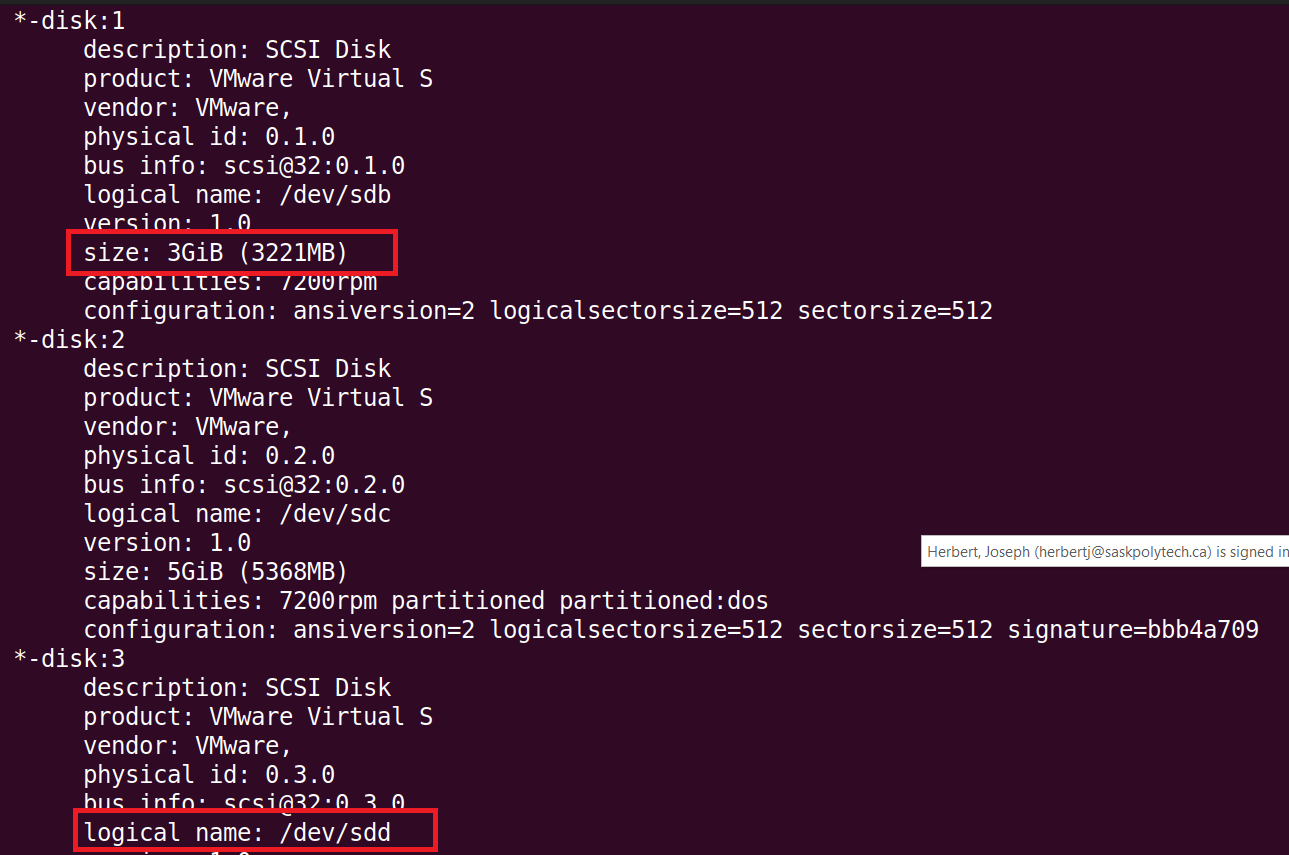
Text

Description automatically generated

I’ve added two drives, they have been detected, and marked as the following:

* /dev/sdb – 3GB drive
* /dev/sdd – 1GB drive

Running the lshw -C disk command:



## Physical Volumes

The first thing we need to do is mark these as “physical volumes”. This will mark the partitions as able to be used by LVM, and will write some LVM information to the disk. Use the “pvcreate” command to do that:

* pvcreate /dev/sdb
* pvcreate /dev/sdd

Text

Description automatically generated

(if you list multiple disks, it marks each of them as a physical volume)

Note: You can also do an existing partition for this, as long as it’s empty. There are differing opinions on whether you should partition or not. For example, Red Hat recommends not doing it – to the point of saying you should erase any existing partition tables.

We have several commands to display information regarding our physical volumes, namely, pvscan, pvs, pvdisplay:

Text

Description automatically generated

Our physical volumes are now ready to go.

## Volume Groups

Now that we’ve got physical volumes created, we can create volume groups. A volume group can contain zero or more physical volumes; we can allocate our logical volumes from the volume group. The logical volumes know nothing about the physical volumes; everything is abstracted away via the volume group.

First, let’s create a volume group. You can name it whatever you want; one convention is to give it the same name as your computer. I’m going to go against convention, and name it “coos”. Note that you can have multiple volume groups on a machine, but typically you only have one ( or maybe two). In this case, a volume group was created when we installed Linux (we won’t use that one).

Initially, we’ll create a volume group with only one of our physical volumes. We can have more than one to start with, but I want to show you how you can add a physical volume to an already existing volume group. To create a volume group:

vgcreate <volume group name> <logical path to physical volume>

In this case, we’ll execute “vgcreate coos /dev/sdb”

Text

Description automatically generated

We can see the information volume groups by using the commands vgscan, vgs, vgdisplay:

Text

Description automatically generated

Text

Description automatically generated

Ok, now let’s add our other physical volume to the volume group via the command “vgextend”:

Syntax: vgextend <VGName> <PhysicalVolume>

In this case, my second physical volume is /dev/sdd. We’ll execute the command “vgextend coos /dev/sdd”

Text

Description automatically generated

We can see that the 3GB coos volume group was extended to include our last remaining physical volume (1GB), for a total of ~ 4GB for our volume group.

## Logical Volumes

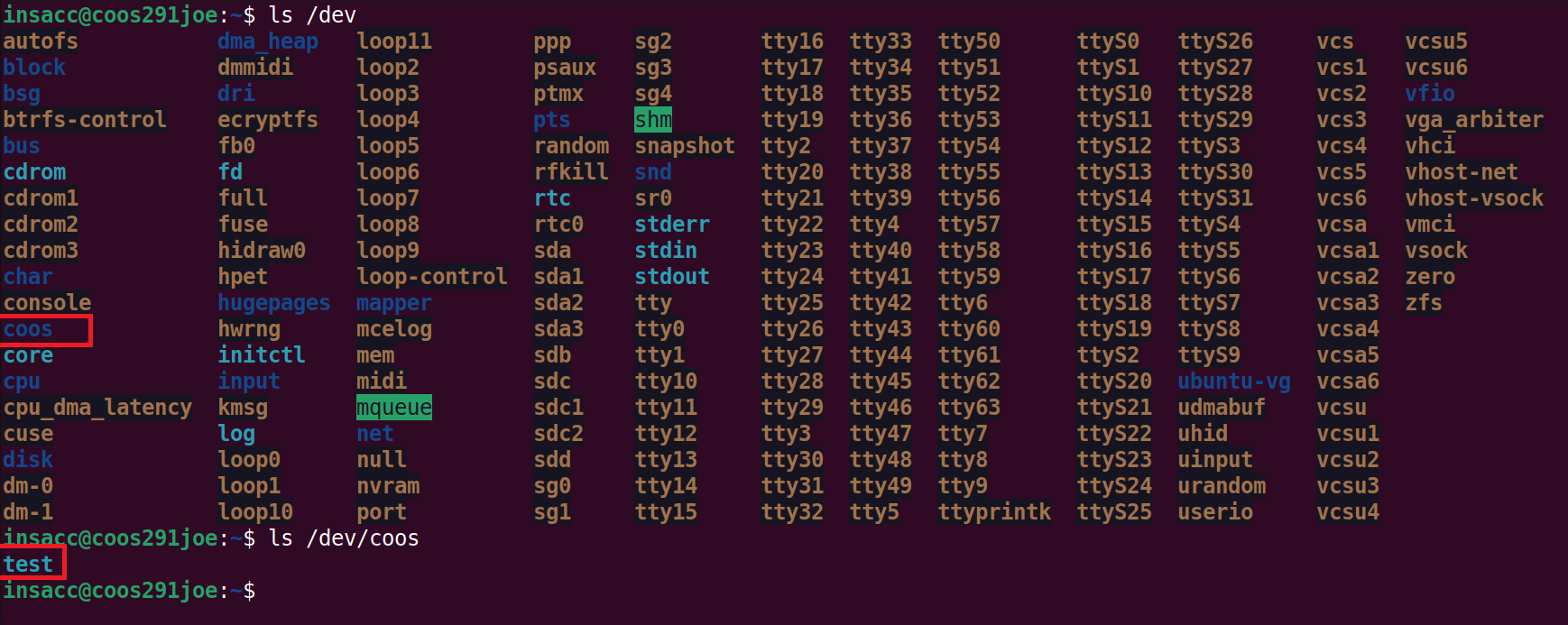
Now that we’ve got our volume group (coos) set up, we can allocate a logical volume and put a filesystem on it.

To create a logical volume named “test” with 2GB of space allocated:

* lvcreate -L 2G -n test coos
* Text

  Description automatically generated
* This is the output you’ll get on a empty drive. If this drive had volumes on it previously, you’d be prompted to wipe out their signatures on the drive.
  + This creates a logical volume called test.

Let’s take a look in the /dev directory. You’ll see that there is a “coos” directory, and beneath that, there is a “test” device. We can use this for our filesystem!



Again, we can see the information on logical volumes via the following commands: lvscan, lvs, lvdisplay

Text

Description automatically generated

Text

Description automatically generated

Let’s now create a filesystem on it:

* mkfs -t ext4 /dev/coos/test
* Text

  Description automatically generated

We can then mount the filesystem:

* mount /dev/coos/test /mnt
* A picture containing text

  Description automatically generated

## Resizing a Logical Volume

Our existing logical volume is 2GB in size. We can also change the size of the logical volume. If you are increasing the size, you can even change a logical volume that has a filesystem on it while the filesystem is mounted:

A picture containing text

Description automatically generated

Comments:

* The command “lvresize” is used to resize our logical volume
  + We can specify a total desired size.
  + Or specify how much we want to add/subtract by using “+”/”-“.
    - We could specify the following:
      * lvresize -L +1GB --resizefs coos/test
* --resizefs
  + This options notifies the filesystem that they have additional space to utilize on the logical volume.
* If you leave off the resizefs option, it resizes the logical volume, but not the filesystem. You can do that manually by (or at least with ext4, the command will be different depending on the filesystem type and filesystem capabilities)
* For ext4
  + resize2fs /dev/coos/test
* For XFS
  + It can be grown with xfs\_growfs command
  + It cannot be easily shrunk.

Decreasing the size of a logical volume is trickier and can’t be done on a mounted logical volume. Be sure that your new space will be large enough to contain all of the existing on the logical volume!

Let’s resize our logical volume back down to 2GB

First, we need to unmount the filesystem

* umount /mnt
* Text

  Description automatically generated

Then, let’s run a filesystem consistency check to be sure nothing is corrupted on our filesystem. Be sure to use the “-f” command to force a thorough check.

* fsck -f /dev/coos/test

Text

Description automatically generated

Next, we’ll need to resize the filesystem. For ext4, use resize2fs:

* resize2fs /dev/coos/test 2G
* Text

  Description automatically generated

Then, after the filesystem is shrunk, we can shrink the logical volume itself:

* lvresize -L 2G coos/test
* Text

  Description automatically generated
  + Note: You will be warned that you might lose data. If you’ve got enough space, and you’re changing it to the same size as you specified for the filesystem, you should be OK.

You can remount it:

Text

Description automatically generated with medium confidence

## Removing the Logical Volume

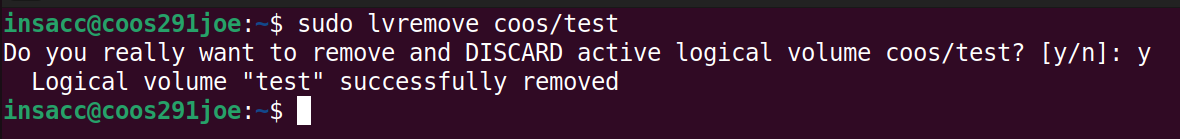
You can remove logical volumes. This will get rid of everything on that logical volume, so either be sure this is what you want to do, and ensure you have backups!

First, unmount the filesystem if necessary:

Text

Description automatically generated

Once unmounted, you can remove it via the lvremove command:



## Removing the Volume Group

If you want to get rid of a volume group, first be sure that you have unmounted any logical volumes in it. It’s usually best to remove the logical volumes from the volume group before you remove the group itself. If you don’t, they’ll be removed when you remove the volume group:

* vgremove coos

Text

Description automatically generated

We can see our volume group “coos” no longer exists.

## Remove a Physical Volume from a volume Group

You can remove a physical volume from a volume group, even if the physical volume is being used.

First, we need to move all data off the physical volume. There needs to be enough room on the other physical volumes to account the data being removed.

* pvmove /dev/sdb

Then, remove the physical volume from the volume group:

* vgreduce coos /dev/sdb

Once you’ve removed it from the volume group, it’s usually a good idea to wipe the label on the device so that LVM no longer thinks it’s a physical volume (pvremove):

Text

Description automatically generated