Storage Management

1. Creating and mounting partitions on a Hard Disk

First let's see how hard disks are detected by the Linux system. Generally the storage devices within a system can be IDE, SCSI or SATA. These are the interfaces provided to attach a storage device to a computer. Based on the interface, the Linux system detects the storage devices as below. The storage device can be a hard disk, a CD/DVD ROM or a tape drive also.

IDE Devices

Generally IDE provides two channels on the motherboard to connect IDE devices. Each channel supports two devices. One of the devices needs to be configured as master, while the other one is configured as slave. Jumpers are provided on the devices to configure it as either master or slave. Thus in a system there can be four IDE devices that can be connected. Some systems may provide more IDE channels to support more IDE devices, but they are rare.

Today IDE is replaced with Serial ATA (SATA). Thus all newer systems have SATA devices. However there are still a lot of old systems that have IDE devices.

The IDE channels are identified as Primary channel and Secondary channel. Thus the four IDE devices that can be connected to a system are identified as

1. Primary Master 2. Primary Slave 3. Secondary Master 4. Secondary Slave

Thus the Linux system identifies them as

1. /dev/hda 2. /dev/hdb 3. /dev/hdc 4. /dev/hdd

respectively.

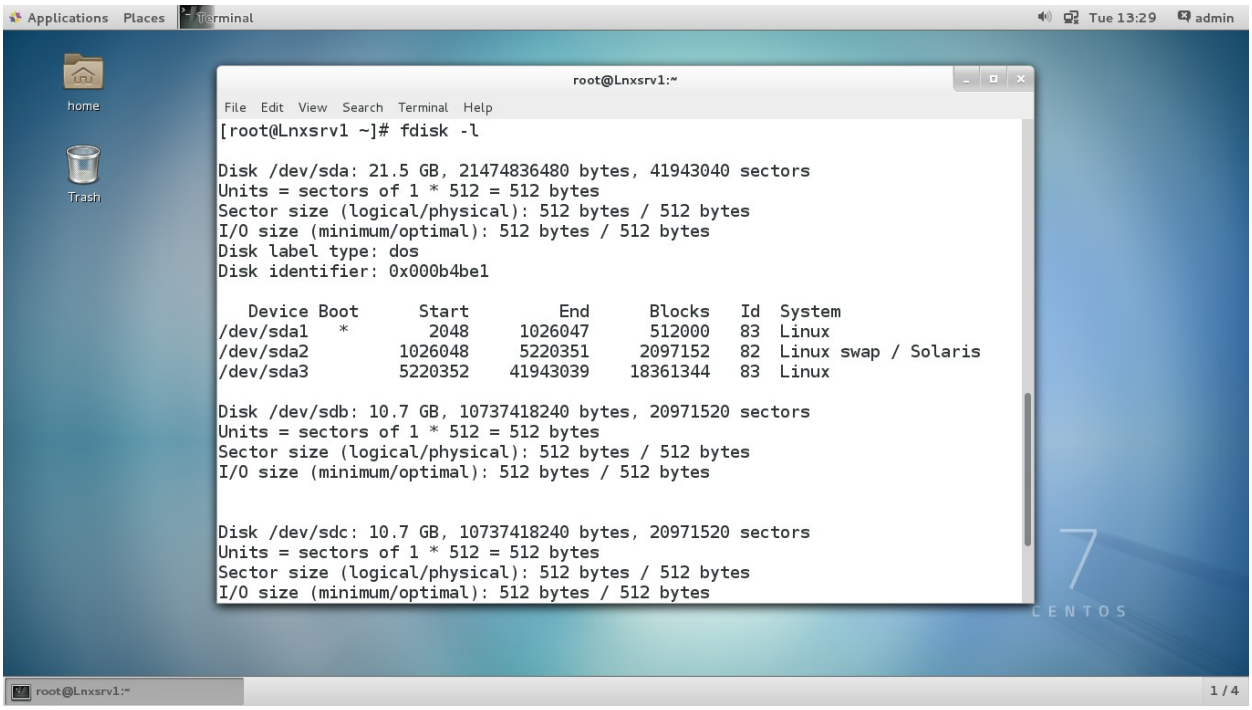
SATA / SCSI Devices /Pen drives

Today most systems use SATA/ SCSI /SAS devices as they provide higher data transfer rates than IDE devices.

The Desktop systems or Laptops generally have SATA devices. The Server systems use SCSI or Serial Attached SCSI (SAS) drives for high performance. These all devices are detected by Linux as

The first Device	/dev/sda
The second device	/dev/sdb
The third device	/dev/sdc ……. And so on.

The command '**fdisk –l'** will display the storage devices connected to the system and partitions created on these devices. The following image shows the output of the **fdisk –l** command.

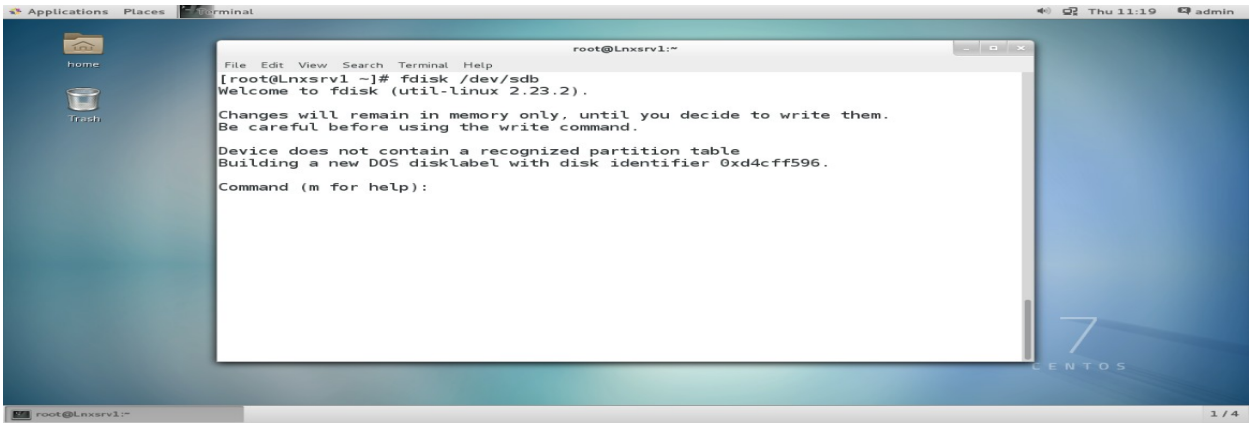


The above output displays three hard disks connected to the system. These are detected as /dev/sda, /dev/sdb and /dev/sdc. Only the hard disk /dev/sda has partitions created. The other two hard disk /dev/sdb and /dev/sdc does not have any valid partitions.

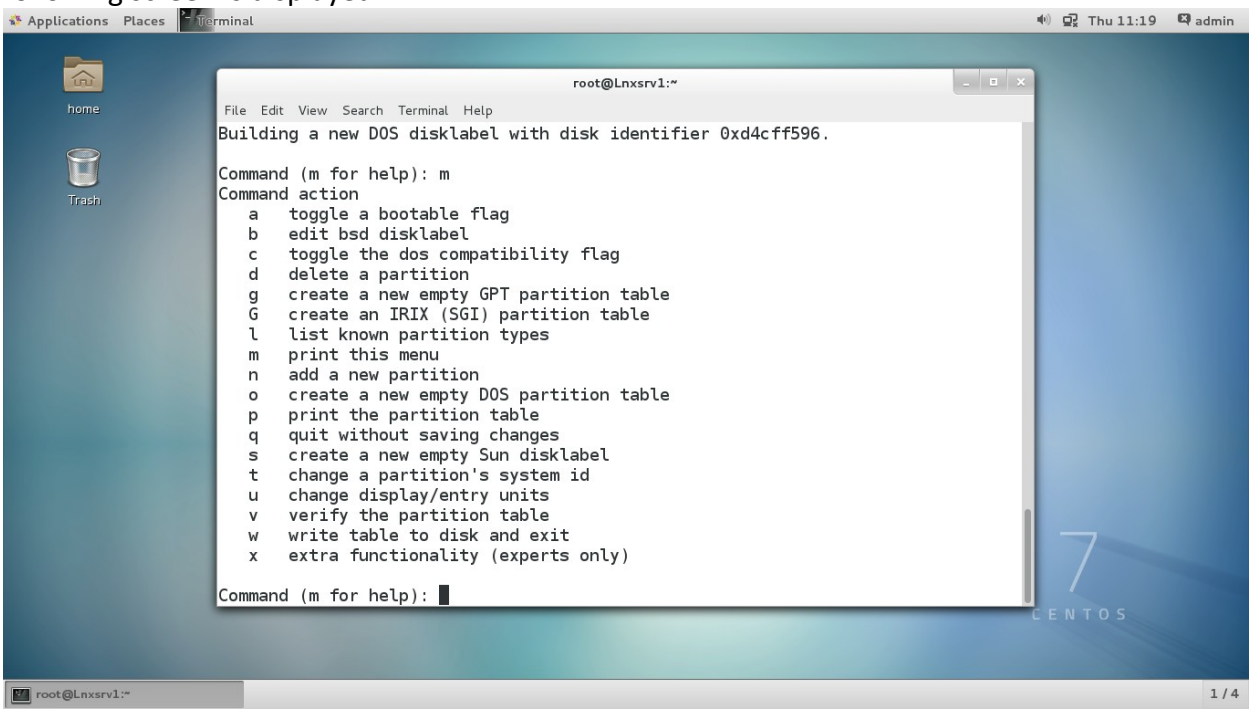
The partitions are given numbers as 1,2,3 etc. Thus the partitions on hard disk /dev/sda are identified as /dev/sda1, dev/sda2 and /dev/sda3 etc. The output also displays the start cylinder number and end cylinder number for a partition.

Creating partitions on a hard disk

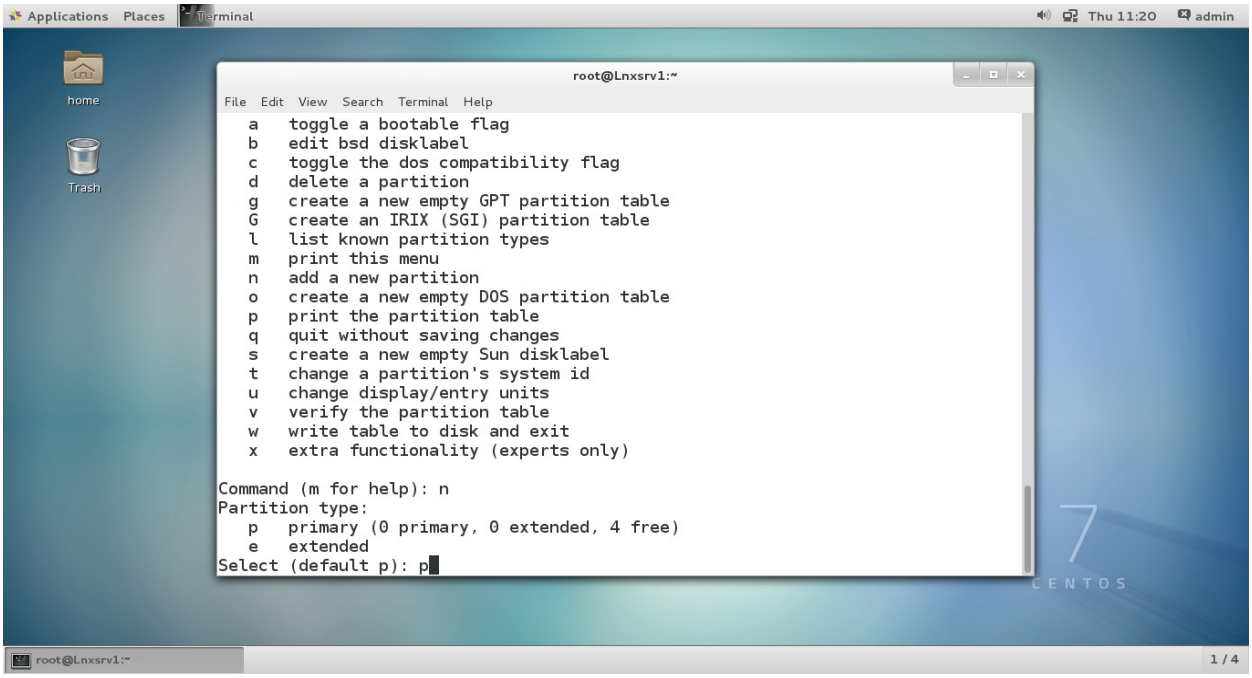
From the above output of fdisk –l find out the hard disk that is newly added. Now to create partitions on the hard disk (e.g. /dev/sdb), give command '**fdisk /dev/sdb.'**



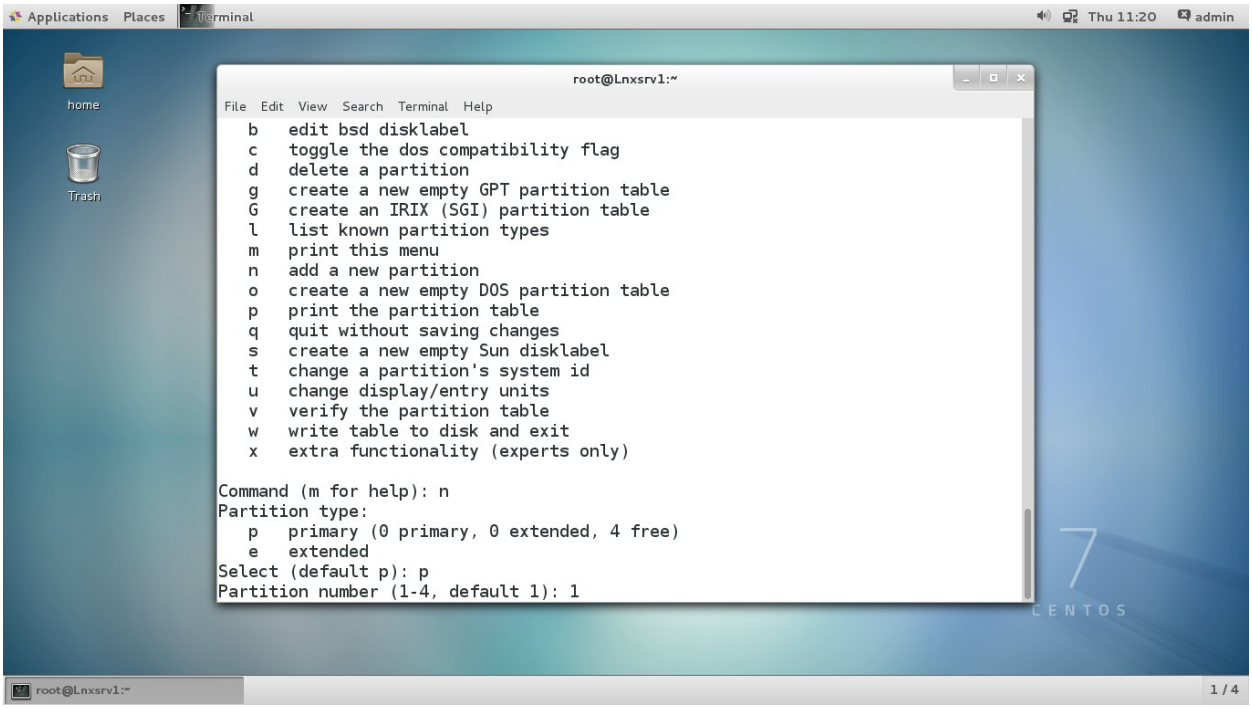
The fdisk command provides a prompt. As suggested in the prompt press ‘m’ for help. The following screen is displayed.



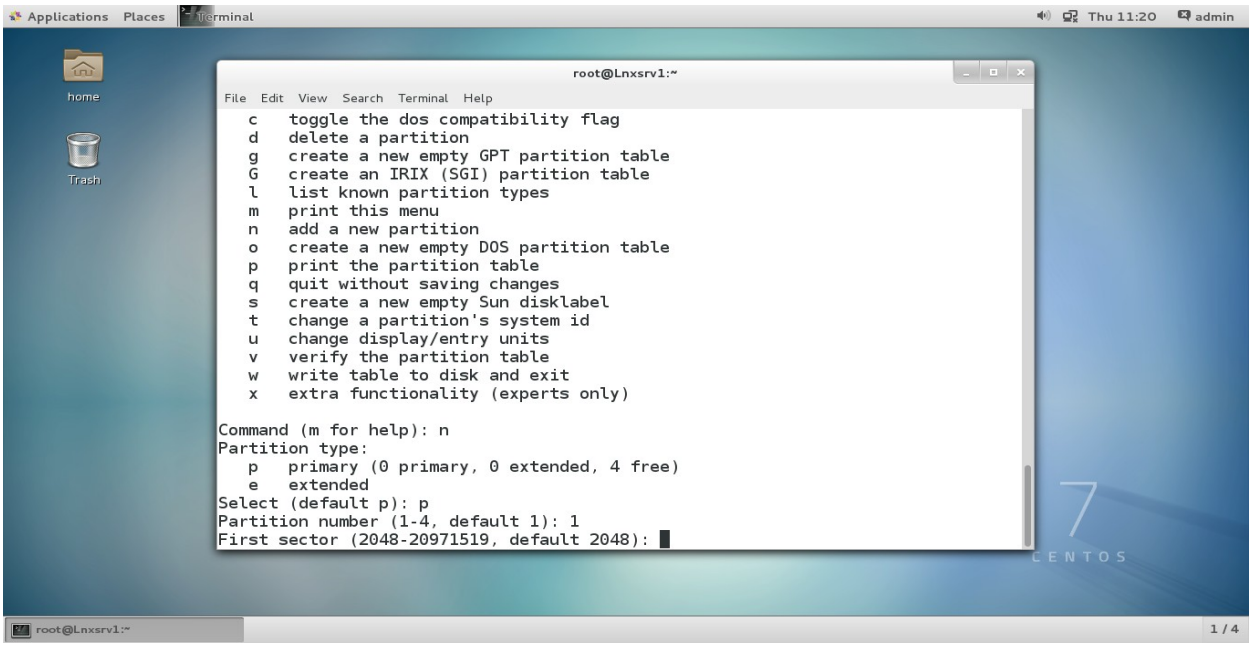
The command displays option that can be used to perform various operations. As we want to create a new partition on the selected hard disk, type n (add a new partition) option and press enter. The following screen appears.



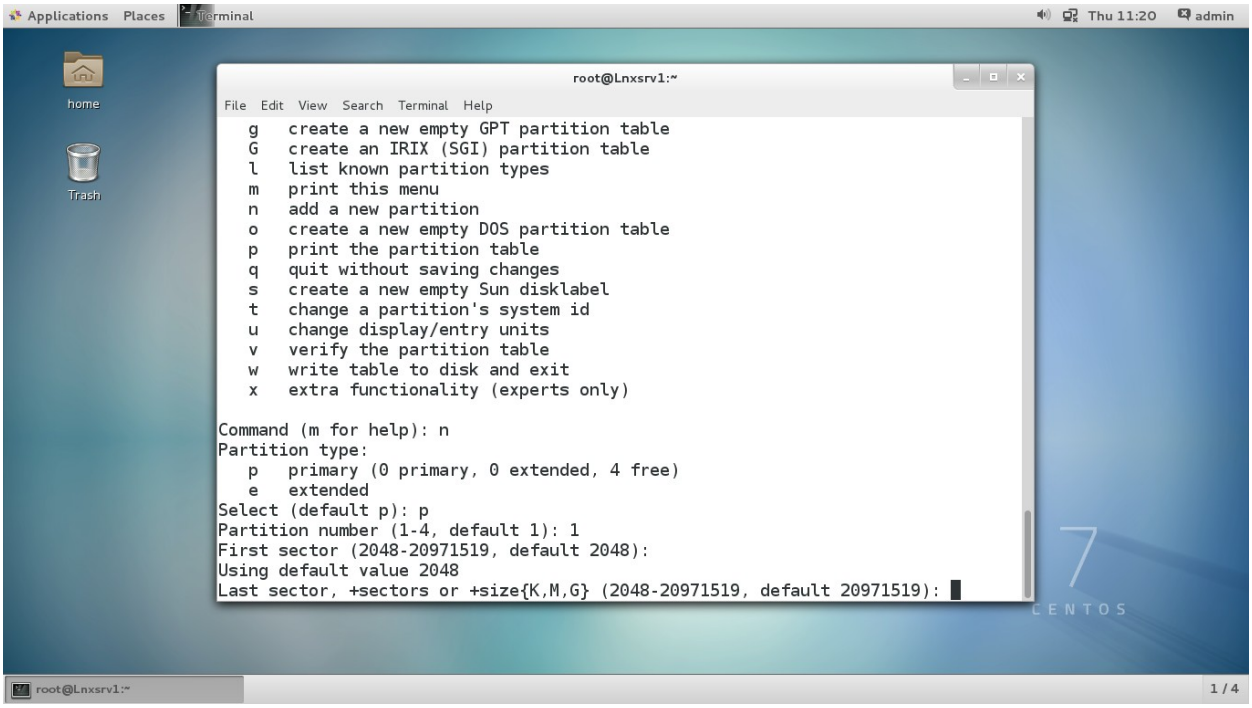
The command requires information about the type of partition to be created. As always the first partition created on the hard disk is primary, type p and press enter. The following options will be displayed.



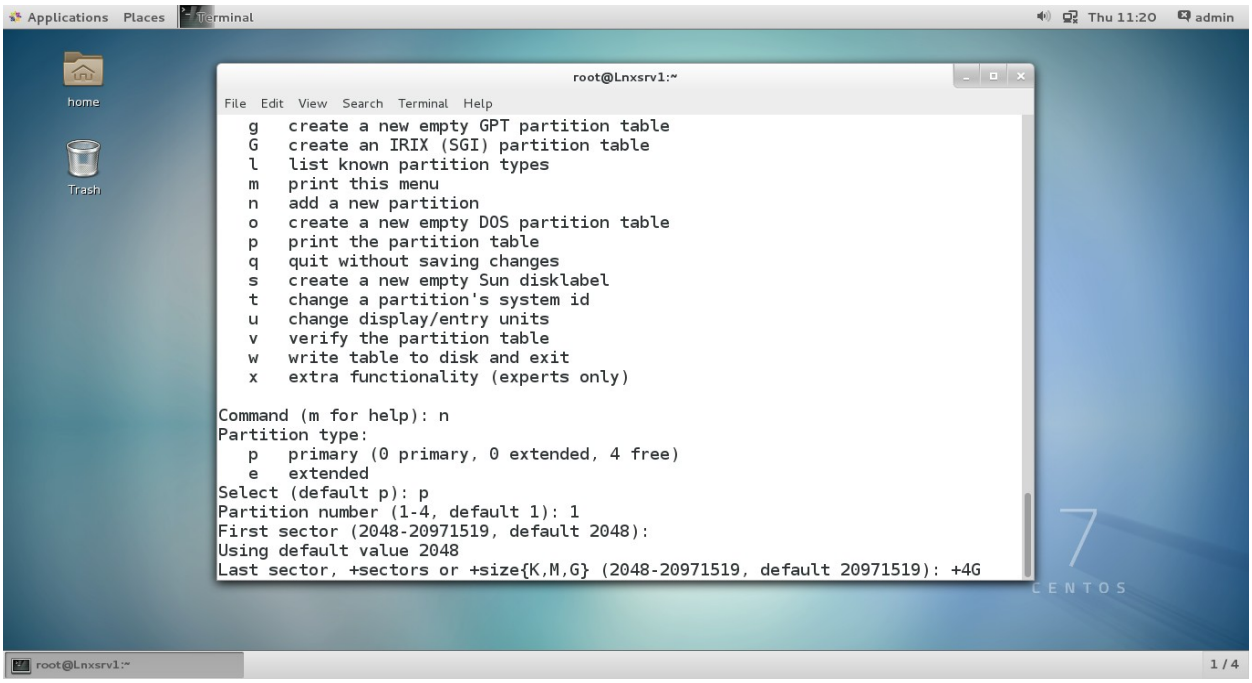
The command requires the number to be assigned to this new partition. As this is the first partition type 1 and press enter.

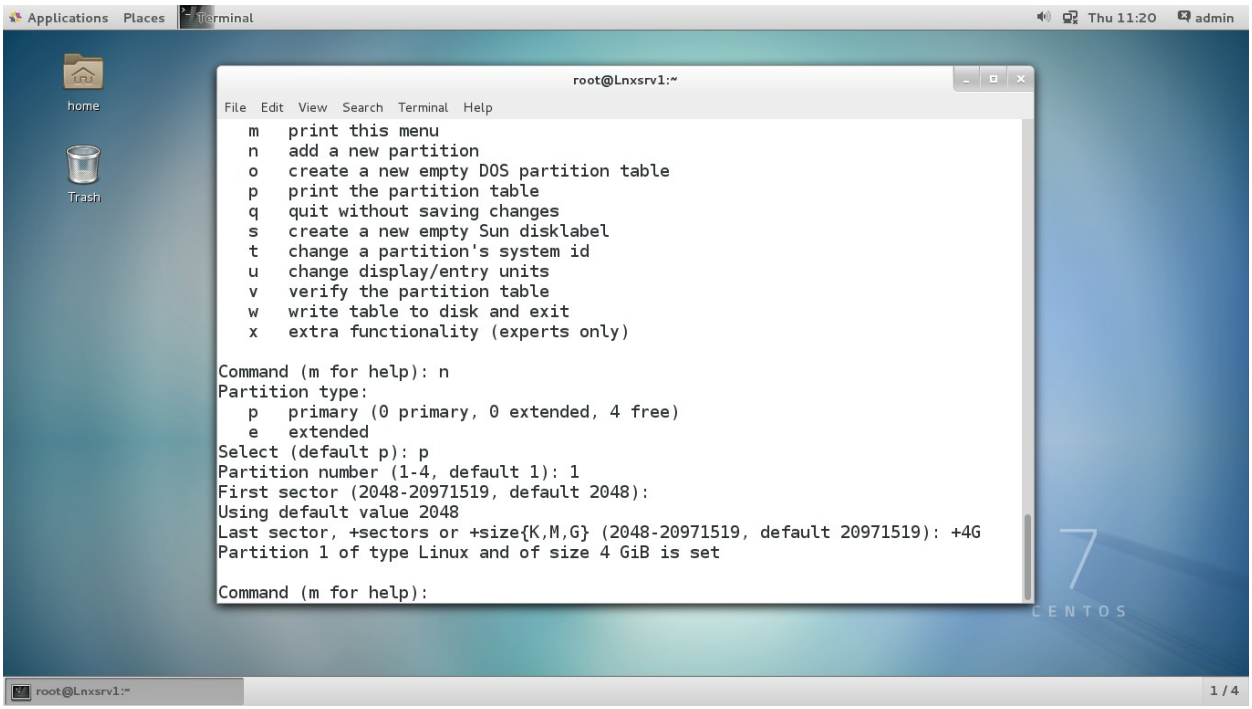


The command needs to know the sector from where to start the partition. Press enter to select the default value of 2048.

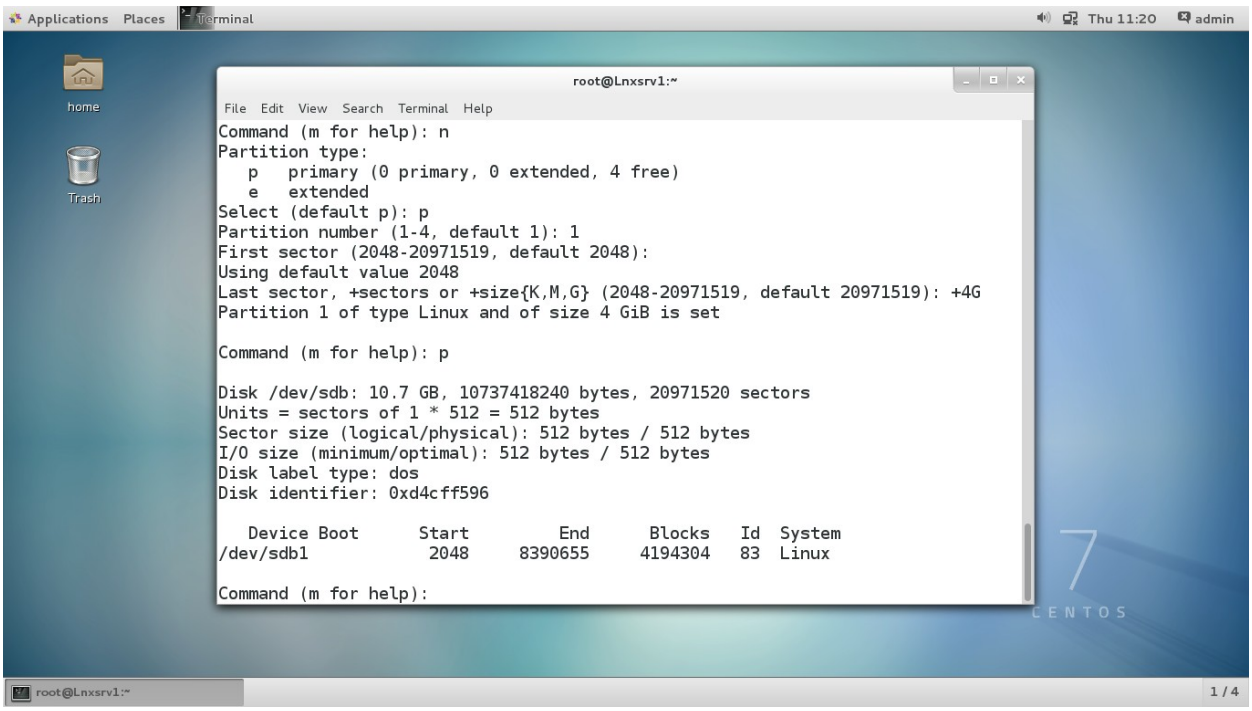


Next it asks to specify the size of the partition. The size of the partition can be specified in terms of last sector (Very difficult to calculate) or directly the size like +300M (300 MB) or +4G (4 GB) etc. Type the required size of the partition and press Enter. In the below picture +4G option is given to create a 4 GB partition.

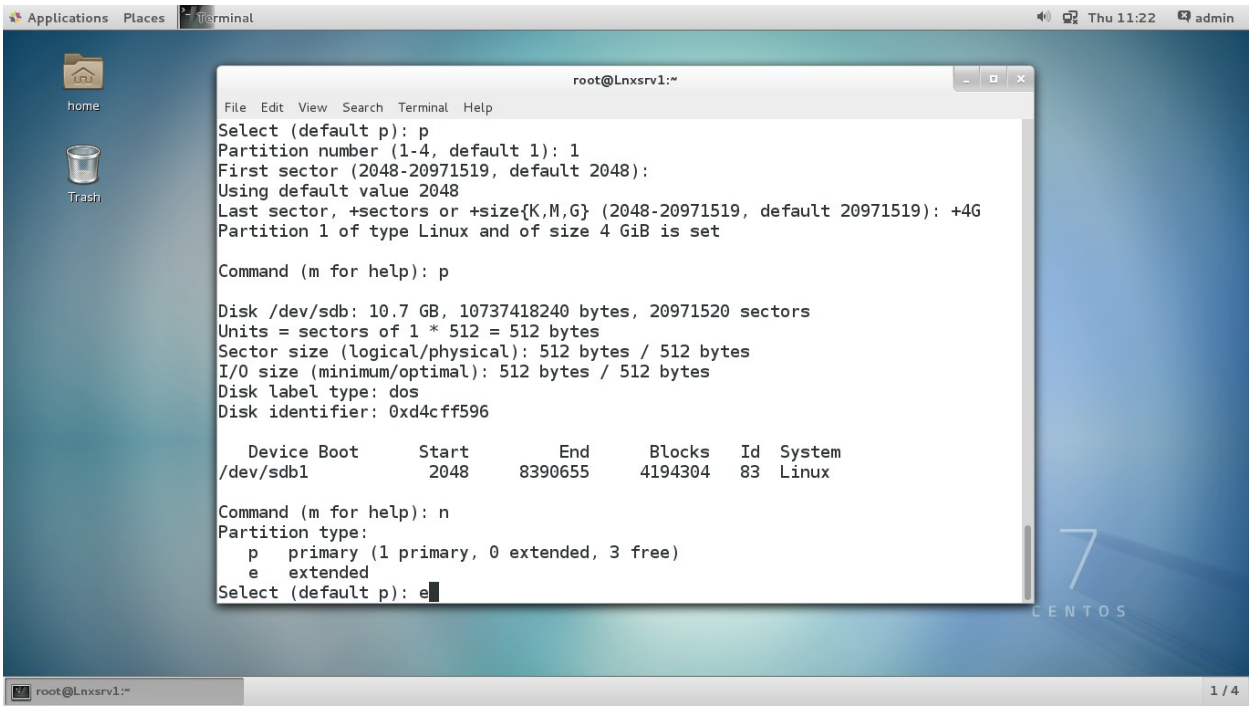




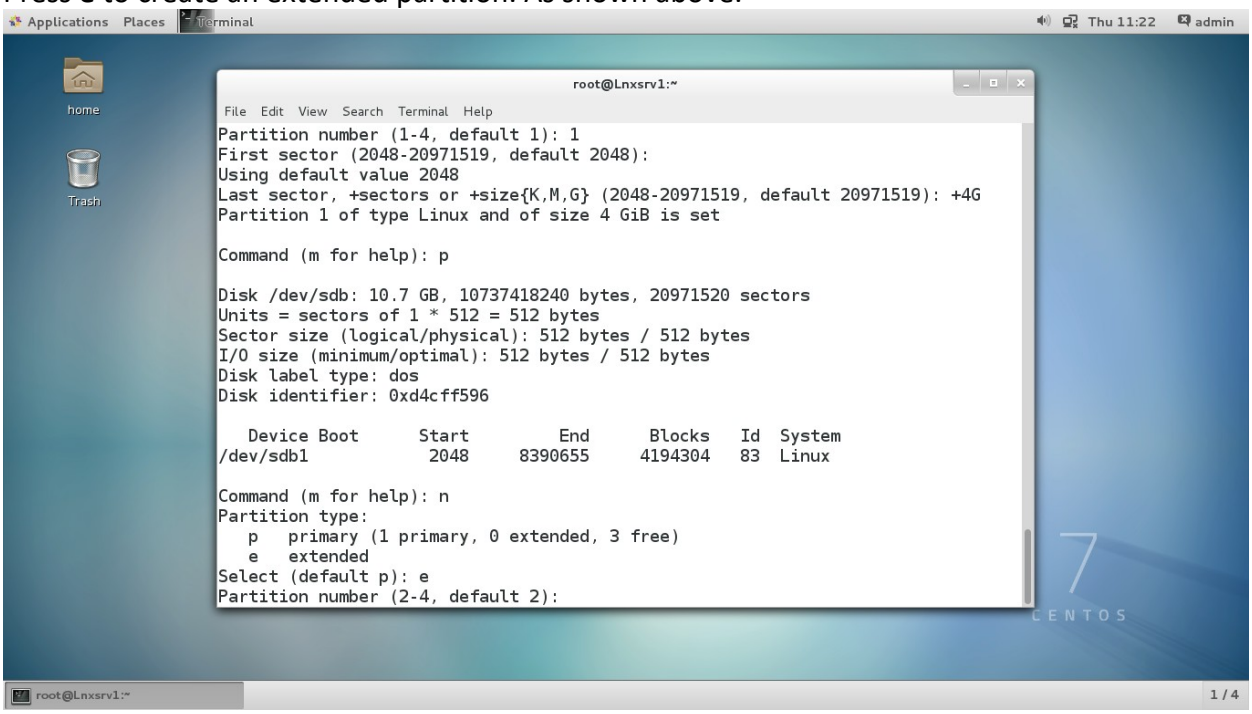
The command will confirm the creation of partition of specified size as shown above. To verify type p and press Enter. This will print the partition table as below.



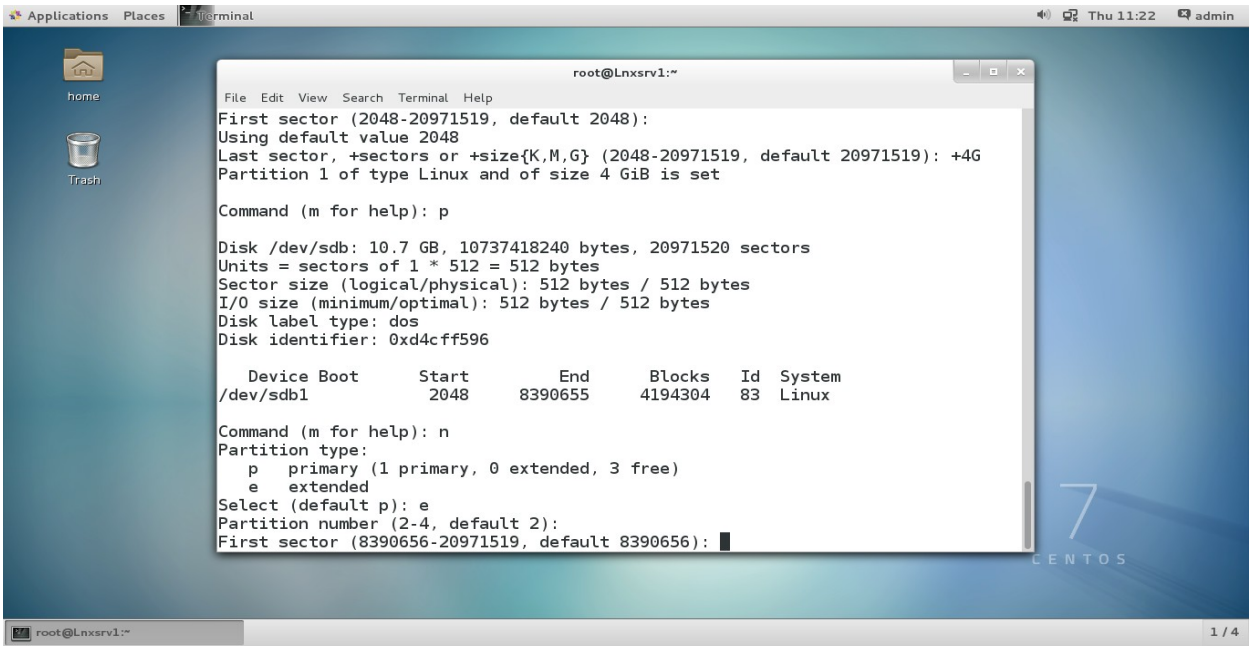
The above picture displays that partition /dev/sdb1 is created. Similarly to create other partitions type n again and press enter.



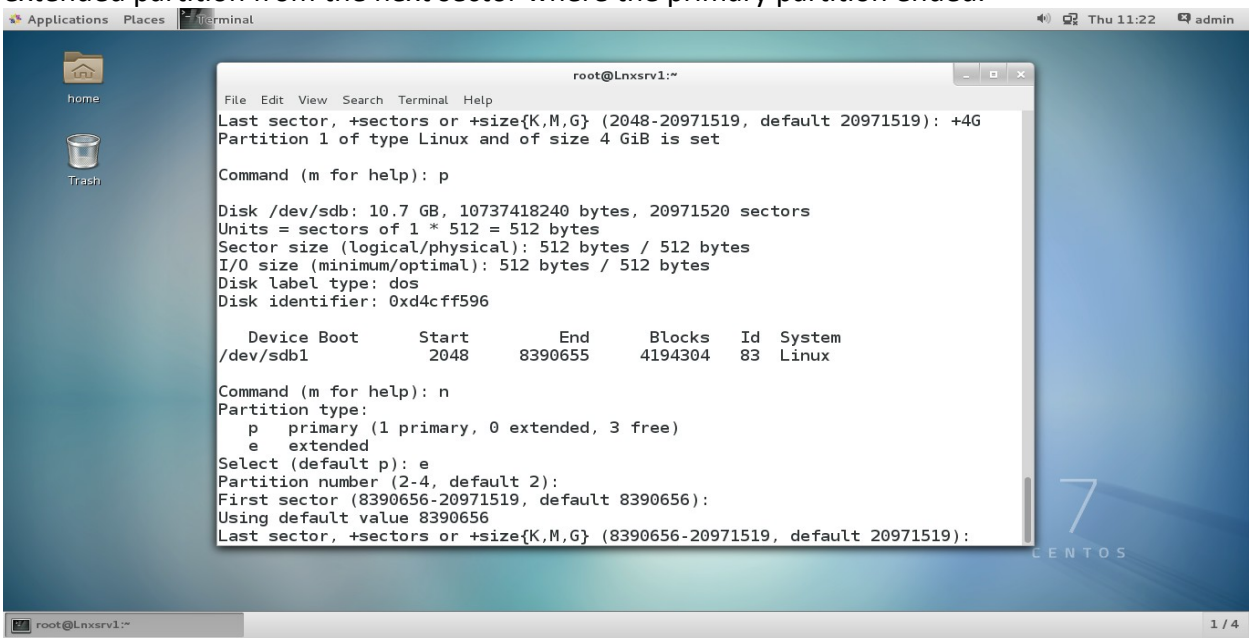
Press **e** to create an extended partition. As shown above.



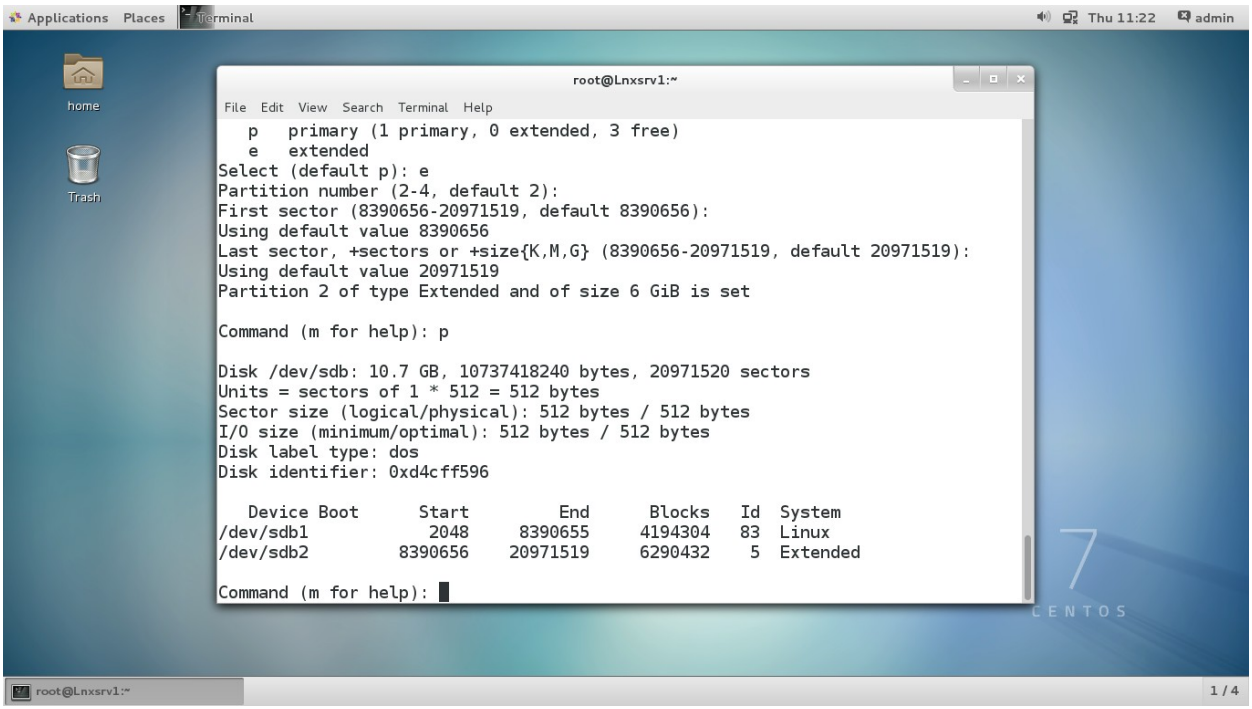
Press enter to accept default. This will give the extended partition number 2.



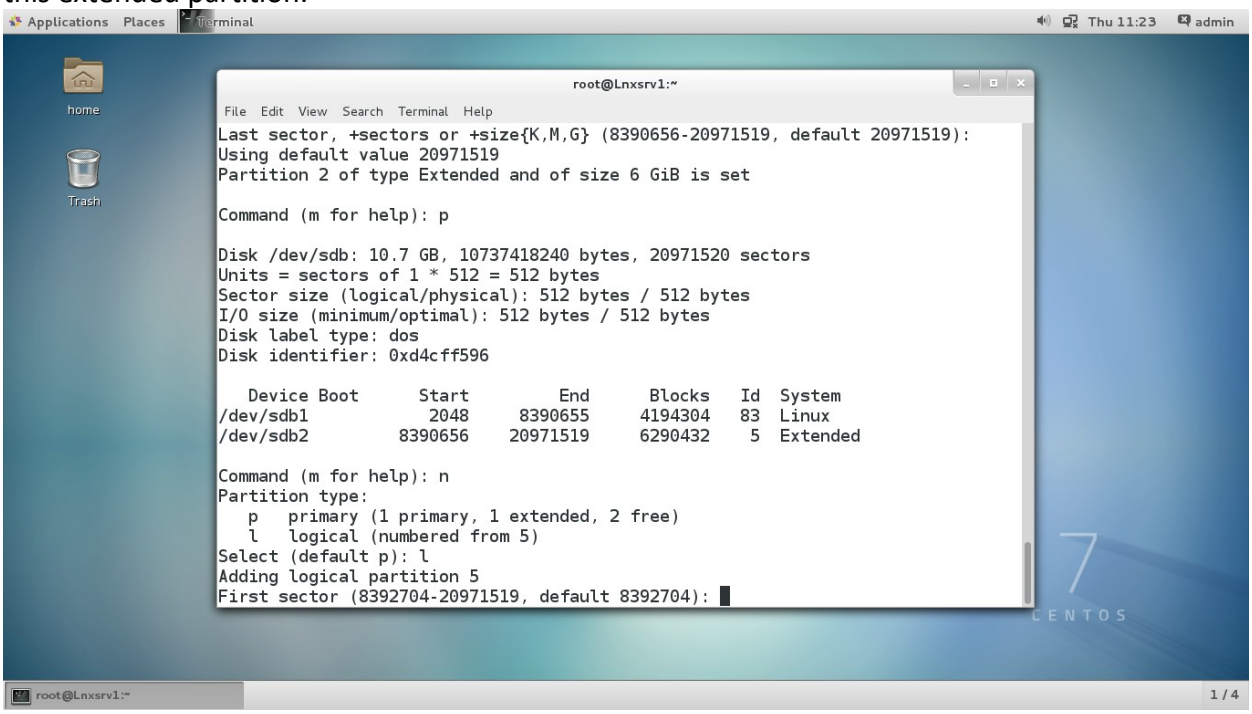
Press enter to select the default starting cylinder for this extended partition. This will start the extended partition from the next sector where the primary partition ended.



Press enter to select the default value for the last sector. This value is the last sector of the hard disk. This will make the extended partition to consume the remaining entire space on the hard disk. Again on the command prompt type **p** and press enter to verify the partitions created. The following picture displays the two partitions – a primary and an extended.

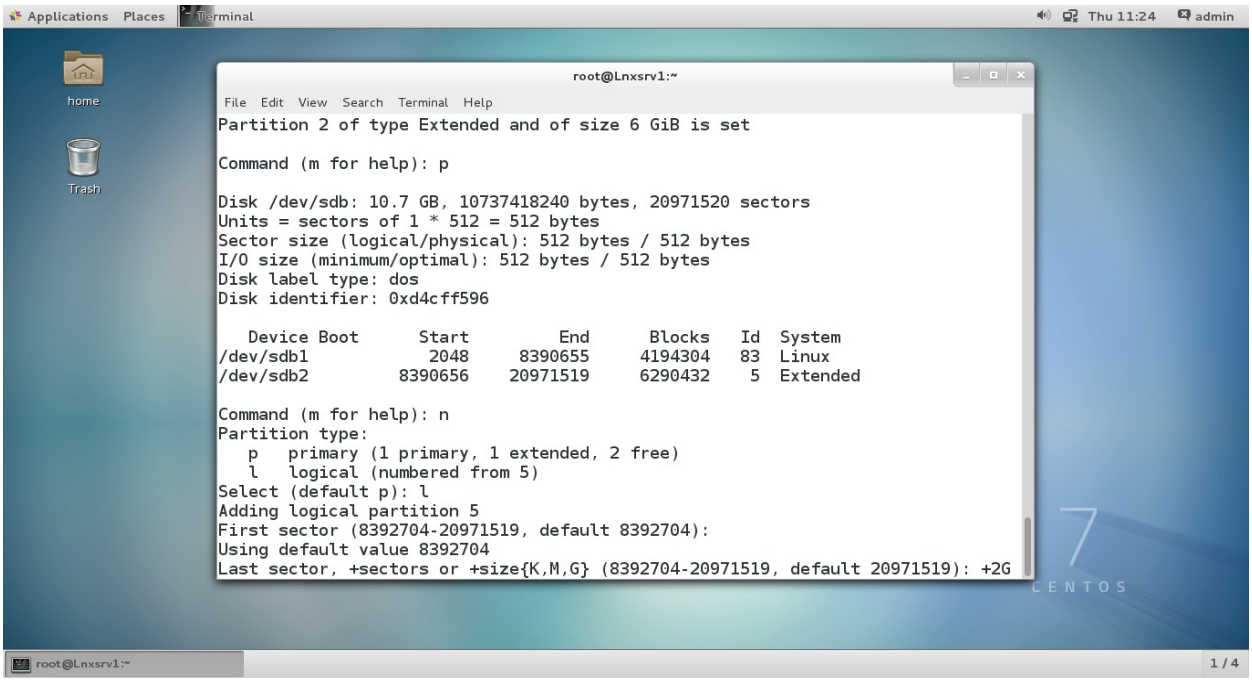


The extended partition cannot be used directly. You need to first create logical drives within this extended partition.

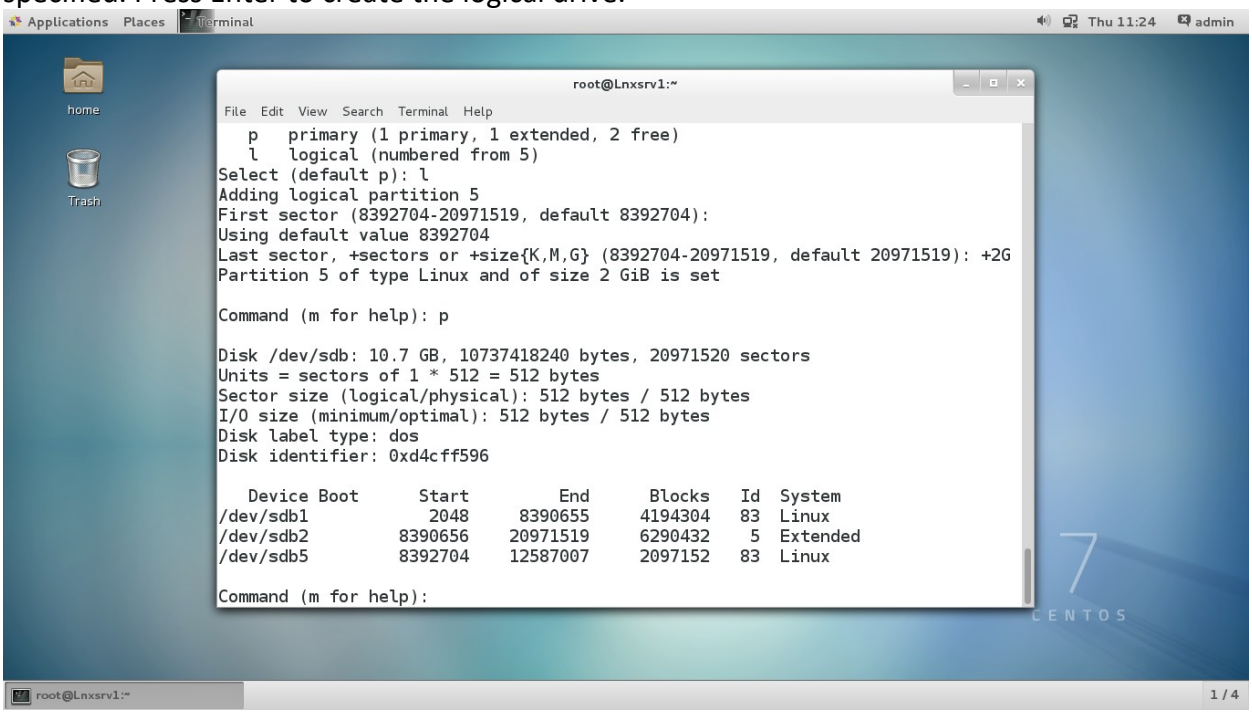


To create a logical drive type n at the **fdisk** command prompt. Now it will not display **e** option. This is because there can be only one extended partition on any hard disk.

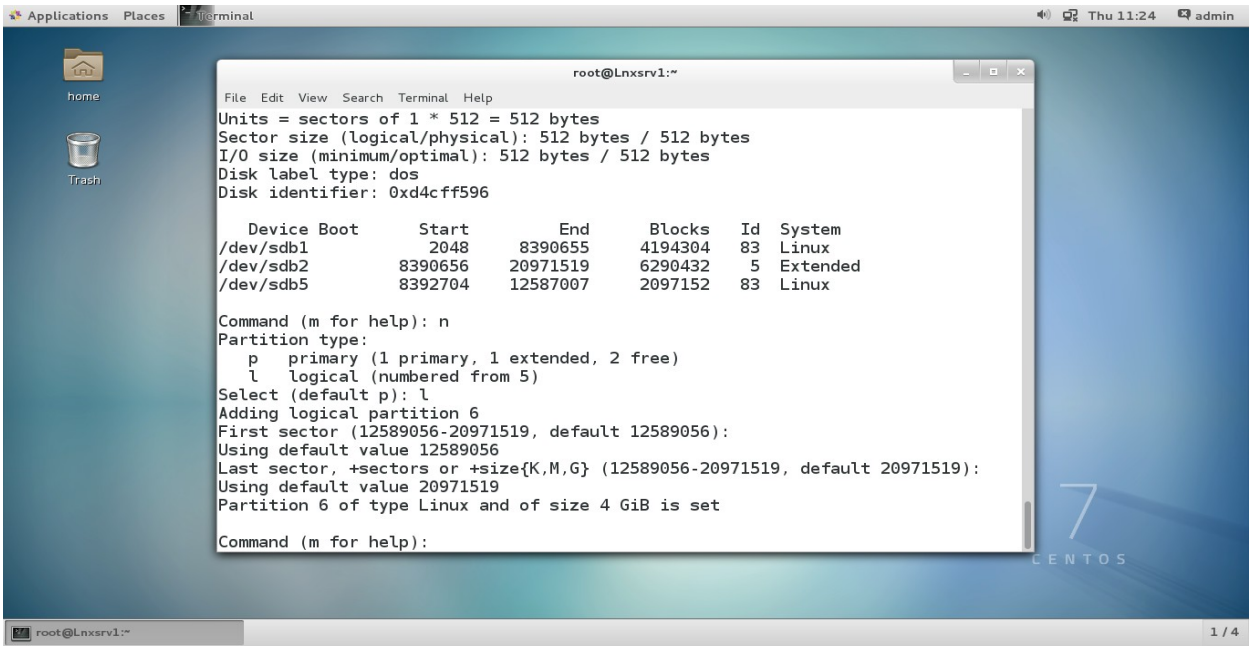
Type '**l**' option to create a logical drive. Press enter to select the default first sector as shown in the above picture.



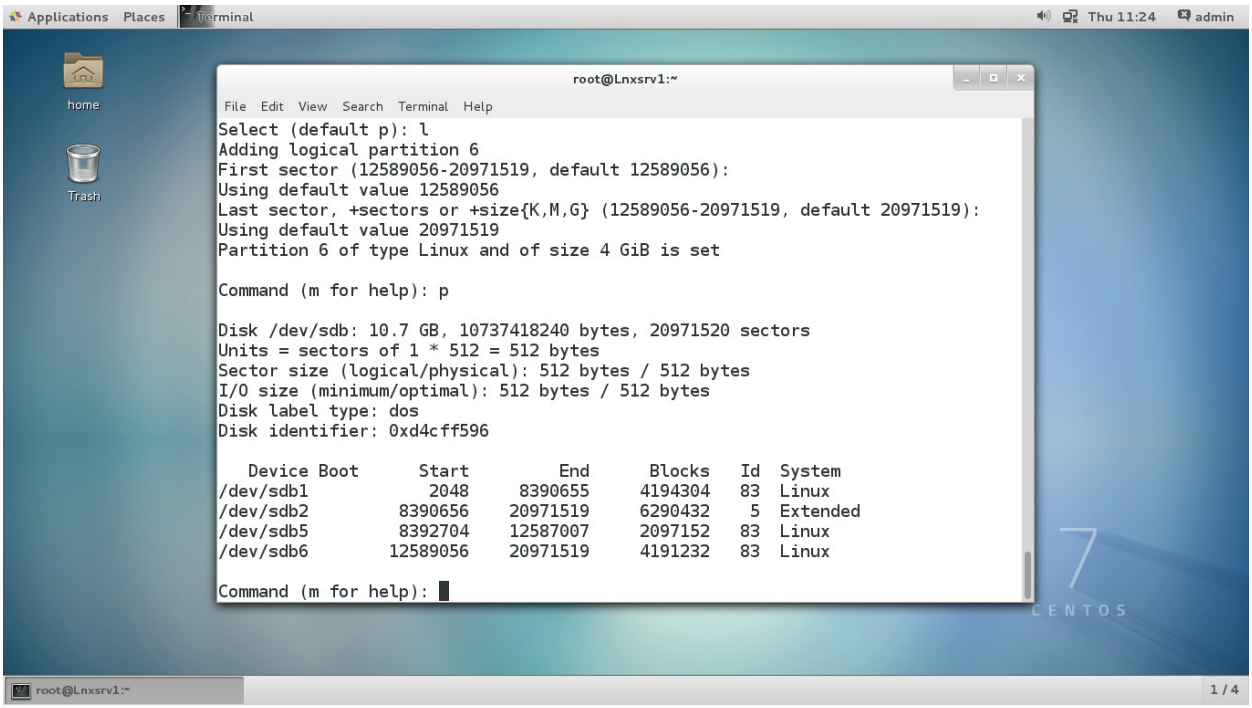
Next specify the size of the logical drive to be created. In the above picture the size of 2GB is specified. Press Enter to create the logical drive.



Press p at the command prompt to display partition table and confirm the creation of logical drive. To create one more logical drive type **n** and enter. Type '**l**' to create another logical drive. Press enter to select the default start sector. Press enter again to select the default last sector. This will allocate all the remaining hard disk space to the logical drive. This is shown the below picture.



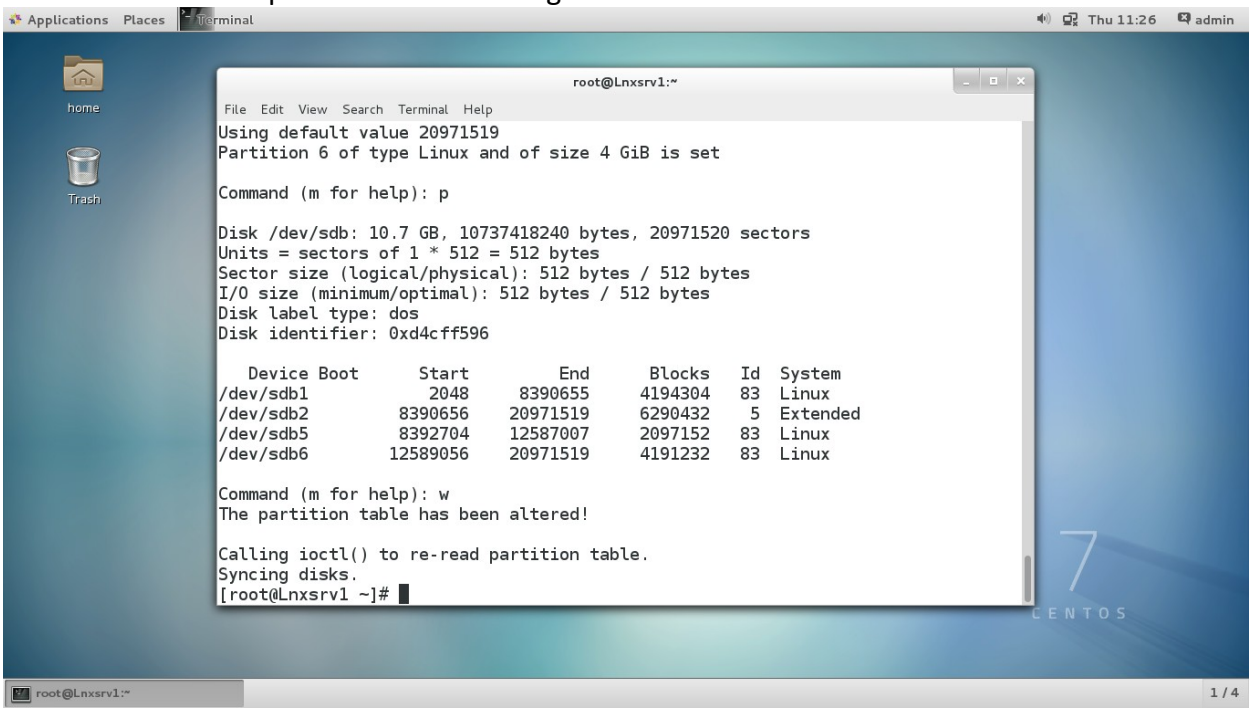
Press p to print partition table and verify the final partition status.



The above picture displays all the partitions created on the hard disk. The first partition /dev/sdb1 is a primary partition. The /dev/sdb2 is an extended partition. The partitions /dev/sdb5 and /dev/sdb6 are the logical drives within the extended partition /dev/sdb2.

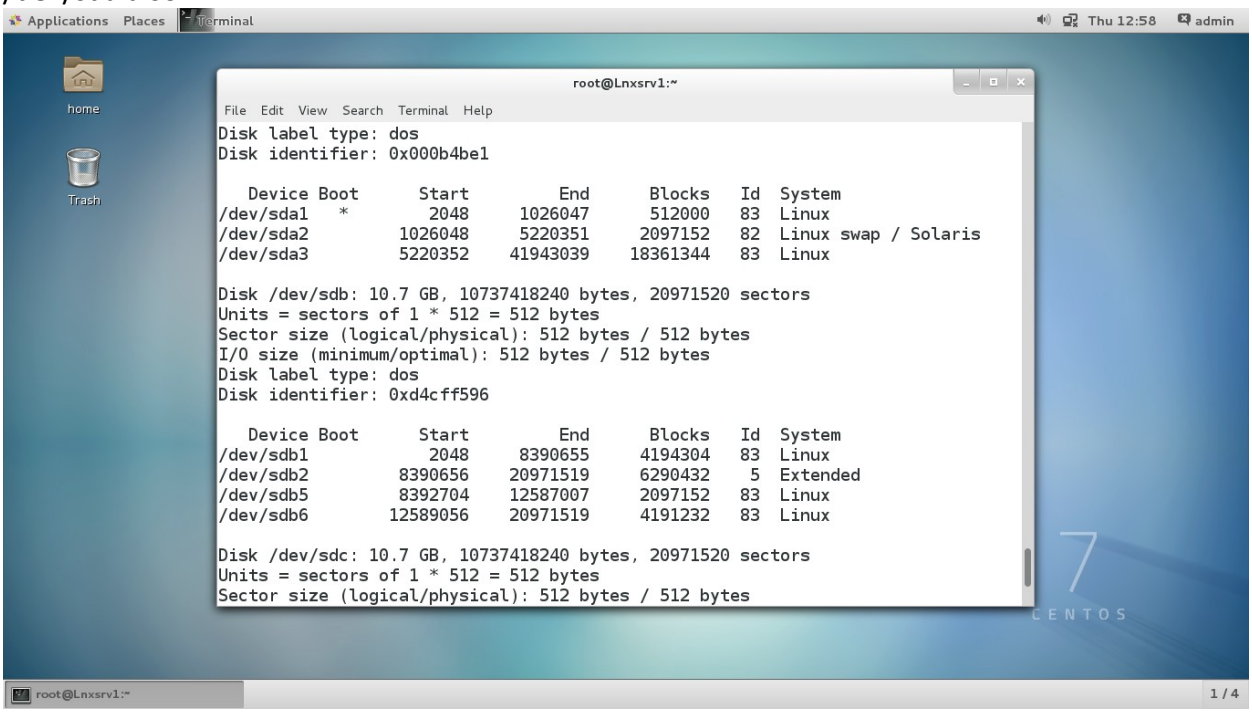
Now as the partitions are created, you need to save this partition table to actual hard disk. The fdisk utility stores this information in the memory. Thus it is temporary. It will be lost if you exit fdisk without saving this information to the actual hard disk.

Thus press w to write changes to the disk.



The fdisk utility will save the partition table to the disk and will exit and the shell command prompt is given.

The fdisk -l command will display the new partitions on the /dev/sdb hard disk and /dev/sda also.

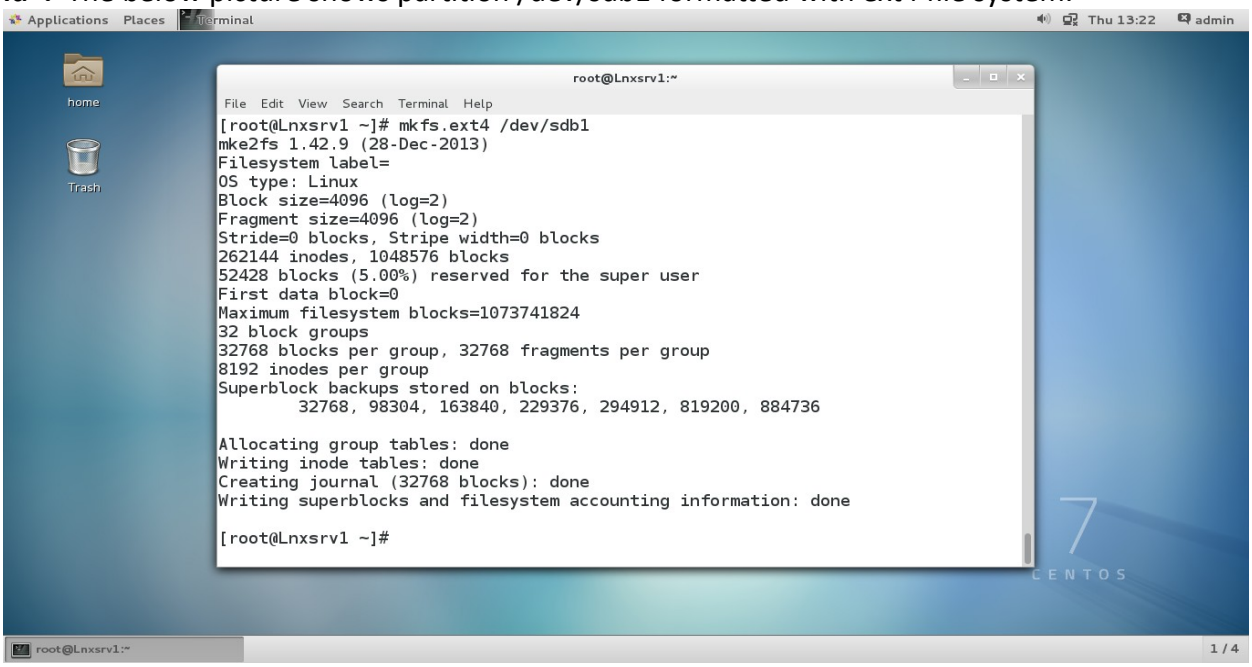


Now to use these partitions they should be formatted with some file system.

Format Partitions

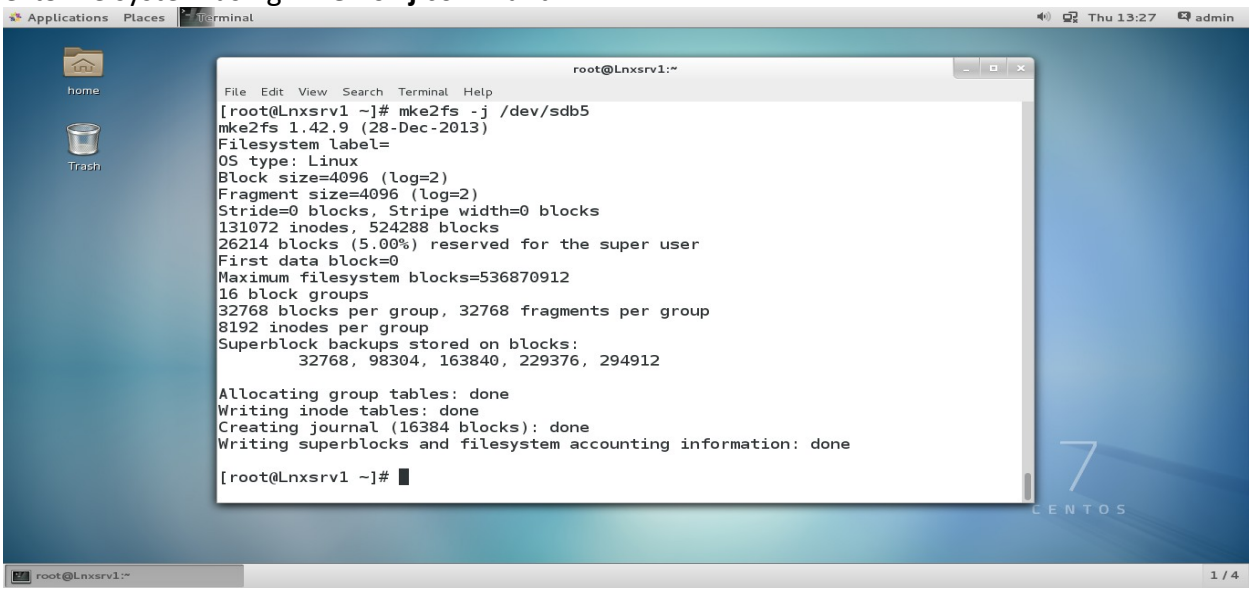
To put file system on newly created partitions, they need to be formatted. Linux supports a lot of file systems like ext4, ext3, ext2, NTFS and vfat etc. The ext4, ext3 and ext2 are the native file systems provided by Linux.

To format a partition using ext4 file system, use the command **mkfs.ext4 'partition id' .** The below picture shows partition /dev/sdb1 formatted with ext4 file system.

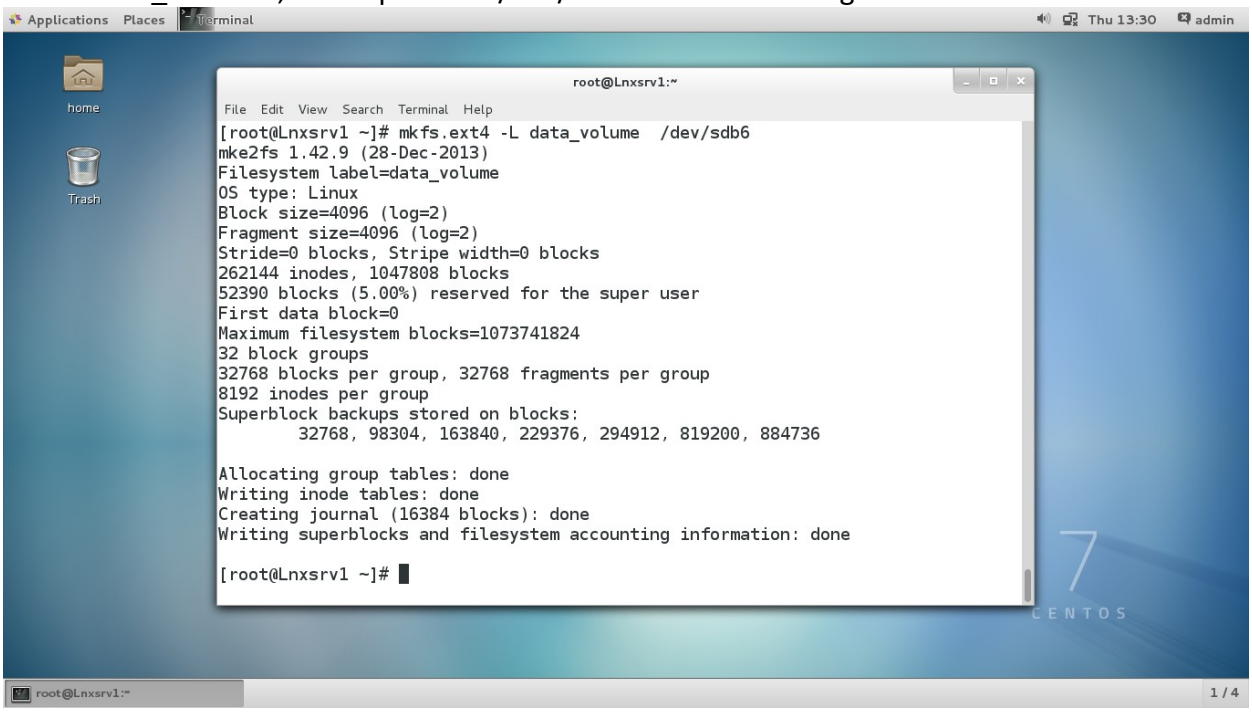


To format a partition using ext3, use either **mkfs.ext3** command or use **mke2fs – j** command.

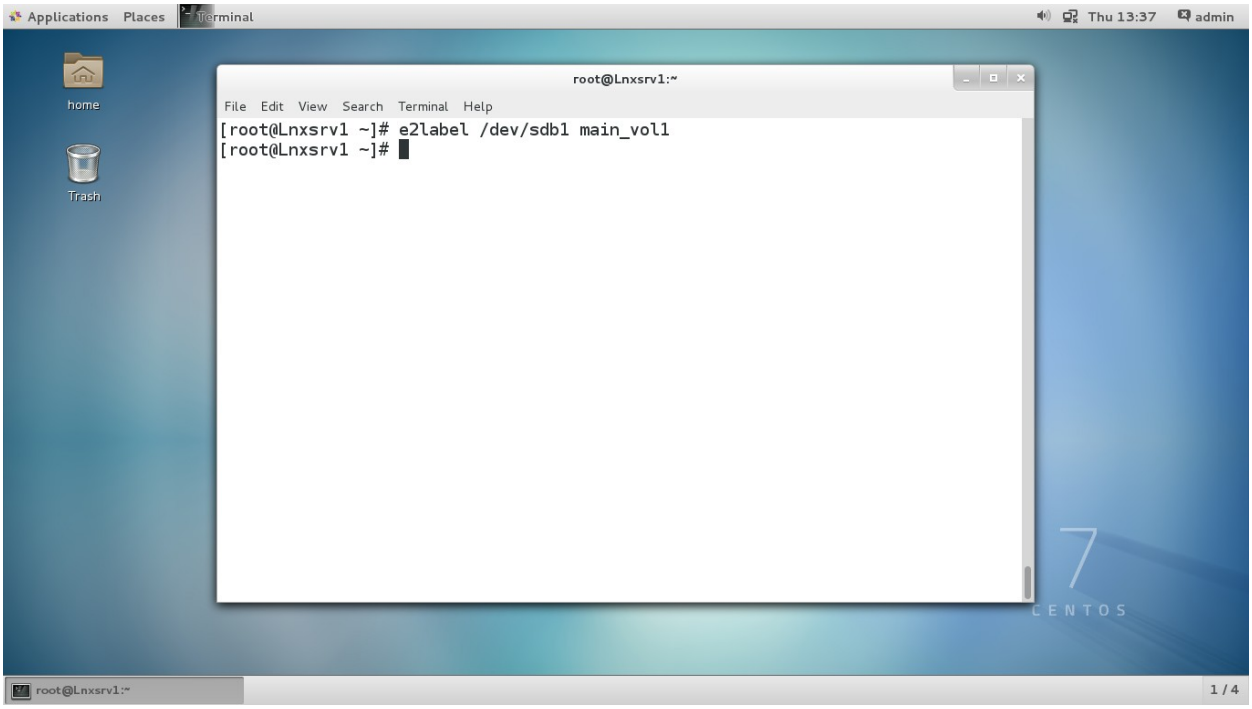
The below picture displays the results of formatting the /dev/sdb5 partition with ext3 file system using **mke2fs –j** command.



The **–L** option of the above commands allows you to specify a volume label for that partition while you format it. The below picture shows the use of –L option to assign a volume label "data_volume", to the partition /dev/sdb6 while formatting it.



Using **e2label** command, the existing volume label of any partition can be changed. Also if during formatting volume label is not specified, using this command a volume label can be assigned. Below picture displays the command to assign a volume label "main_vol1" for the partition /dev/sdb1.



Now that partitions are created and formatted with the file system, they are ready to use. However in Linux unlike windows no drive letter is given to the partitions. Linux is a single root file system. The starting point of it is /. Thus to access the new partitions we need to first attach these partitions to the existing file system. This is known as mounting. In another words these partitions will be linked to a directory under /. Thus whatever data is written in those directories is actually stored on these partitions.

The below picture shows the command to mount the tree partitions to three different directories under /.

```
[root@Lnxsrv1 ~]# mkdir /data1
[root@Lnxsrv1 ~]# mkdir /data2
[root@Lnxsrv1 ~]# mkdir /data3
[root@Lnxsrv1 ~]# mount -t auto /dev/sdb1 /data1
[root@Lnxsrv1 ~]# mount -t auto /dev/sdb5 /data2
[root@Lnxsrv1 ~]# mount -t auto /dev/sdb6 /data3
```

Thus any data created in the directory /data1 will be actually stored on the partition /dev/sdb1. Same happens with other directories and the partitions also.

To check if mount command was successful or to find how devices are mount, just issue mount command. The output will display all the current mountings. Below picture shows the output of the mount command.

```
File  Edit  View  Search  Terminal  Help
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blki
o)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime
,perf_event)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hu
getlb)
configfs on /sys/kernel/config type configfs (rw,relatime)
/dev/sda3 on / type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
selinuxfs on /sys/fs/selinux type selinuxfs (rw,relatime)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=32,pgrp=1,time
out=300,minproto=5,maxproto=5,direct)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
mqueue on /dev/mqueue type mqueue (rw,relatime,seclabel)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,seclabel)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw,relatime)
sunrpc on /proc/fs/nfsd type nfsd (rw,relatime)
/dev/sda1 on /boot type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
gvfsd-fuse on /run/user/1000/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,relatime
,user_id=1000,group_id=1000)
/dev/sdb1 on /data1 type ext4 (rw,relatime,seclabel,data=ordered)
/dev/sdb5 on /data2 type ext3 (rw,relatime,seclabel,data=ordered)
/dev/sdb6 on /data3 type ext4 (rw,relatime,seclabel,data=ordered)
[root@Lnxsrv1 ~]#
```

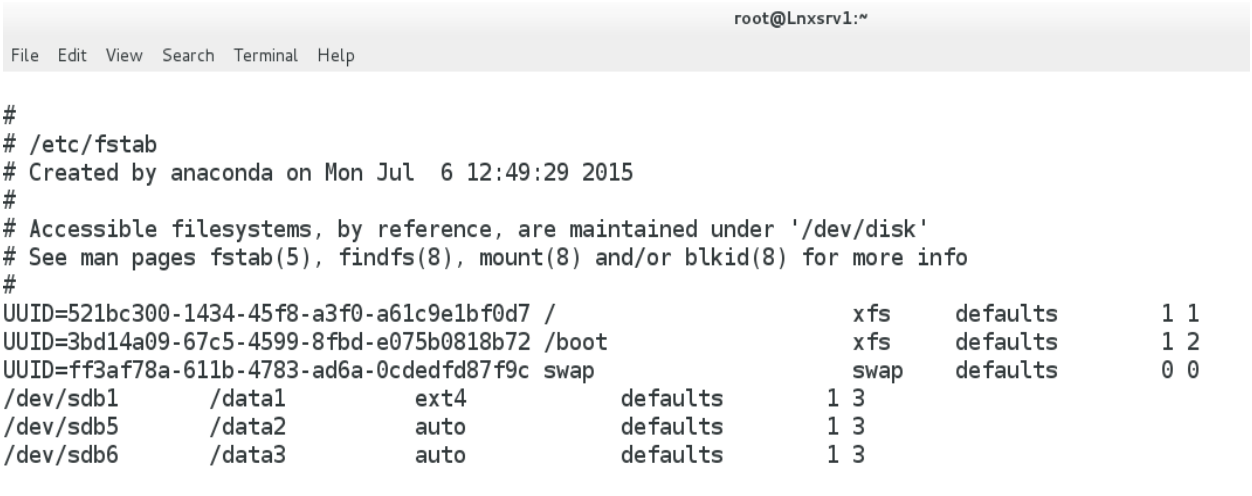
The last three lines in the above picture give details about mounting of /dev/sdb1, /dev/sdb5 and /dev/sdb6 partitions.

However these mountings are dynamic in the sense if you restart the system these mountings will be removed and lost. After restart the mount command will not display the mountings for /dev/sdb1, /dev/sdb5 and /dev/sdb6. As these partitions are not mounted, the users will not be able to access the data. Thus administrator will need to mount these partitions each time the system restarts.

Note: - The mounting of devices can be performed by the root user only. This is the default behavior of the Linux system.

Thus to make these mounting permanent, Linux provides a method. Linux provides a file which it refers each time Linux boots. Thus adding the mount parameters to this file makes Linux mount these devices during booting. The file name is **fstab** and is located in **/etc** directory.

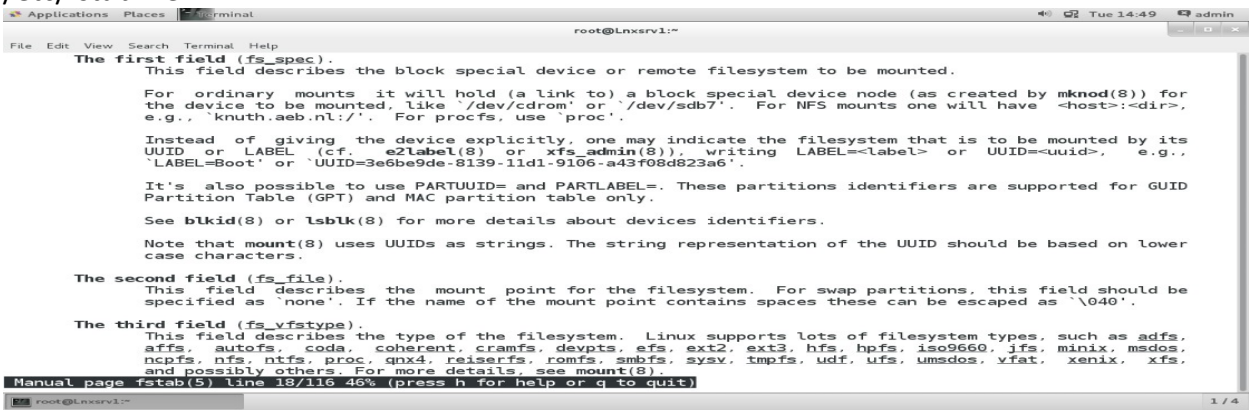
The below picture shows the /etc/fstab entries for the partitions /dev/sdb1, /dev/sdb5 and /dev/sdb6 partitions. To add entries use the command vi /etc/fstab. This will open the fstab file. Now add the entries as shown in the below picture. Save the file.



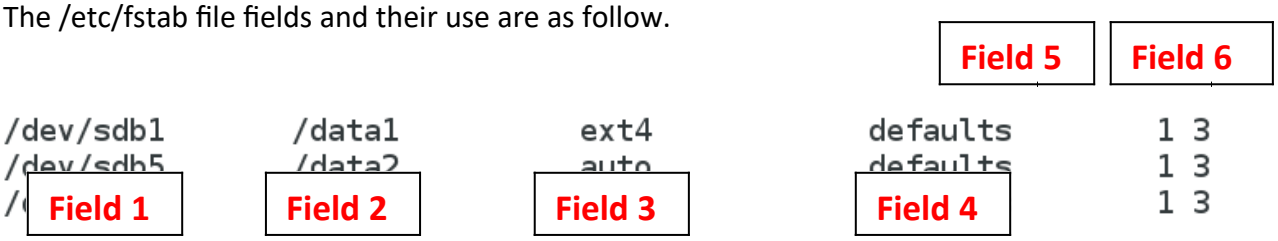
To know the fields of the above /etc/fstab file, give following command.

```
[root@Lnxsrv1 ~]# man 5 fstab
```

The output of this command is as below. The output displays the field and its use in the /etc/fstab file.



The /etc/fstab file fields and their use are as follow.



The **first field** identifies the physical storage device like a partition or a CDROM or a Pen drive etc. that needs to be mounted. In case of nfs or smb file sharing access this can be remote file system path also. The entry for the remote computer will be of the type 192.16810.1:\\ shared.

The **second field** identifies the mount point. This is a directory on the local / partition where the physical storage device will be mounted.

The **third filed** identifies the file system with which the physical storage device is formatted. Specify the correct file system present on that device. If the correct file system present on the mounting device is not known type **auto** in this field. With auto option the Linux will automatically identify the file system present on the disk.

The **fourth field** identifies the mounting options. While Linux is mounting a device it refers to these options and according to the options it mounts the device. Some of the options are

a. noauto – This option tells Linux system not to mount the device while booting. Only root user with the mount command can mount it.
b. auto – Mount the device automatically each time Linux system boots.
c. RW – Mount the device in read/write mode.
d. RO – mount the device in read only mode.
e. user – allows the specified user to mount this device using mount command.
f. owner – allows the device owner to mount the device.

The defaults option contains auto, RW, nouser options.

The **fifth field** is used by the dump command, to decide which file system to dump. The number in this field decided the dump frequency for that file system.

The **sixth field** is used by the fsck program. In case when the system recovers from a crash the number in this field is used by the fsck program to decide the order to check the file systems for any error. The root (/) file system should be the first.

Now onwards whenever the Linux system starts these partitions will be automatically mounted and the data on these partitions becomes available to the users.

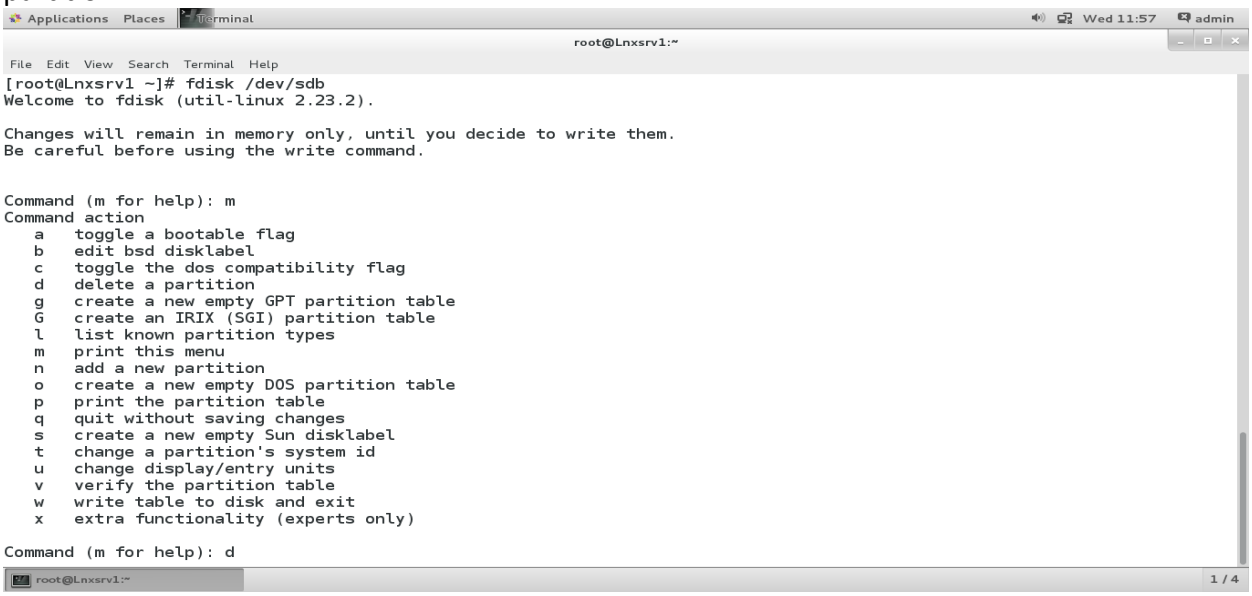
Another advantage of adding an entry to the fstab file is that you can mount a device using mount command without specifying all the parameters. Thus in above case to mount /dev/sdb1 with entry in fstab one can specify just

mount /dev/sdb1 OR mount /data1

The mount command refers to the fstab file and gets the other required parameters like file system.

Deleting Partitions on a hard disk.

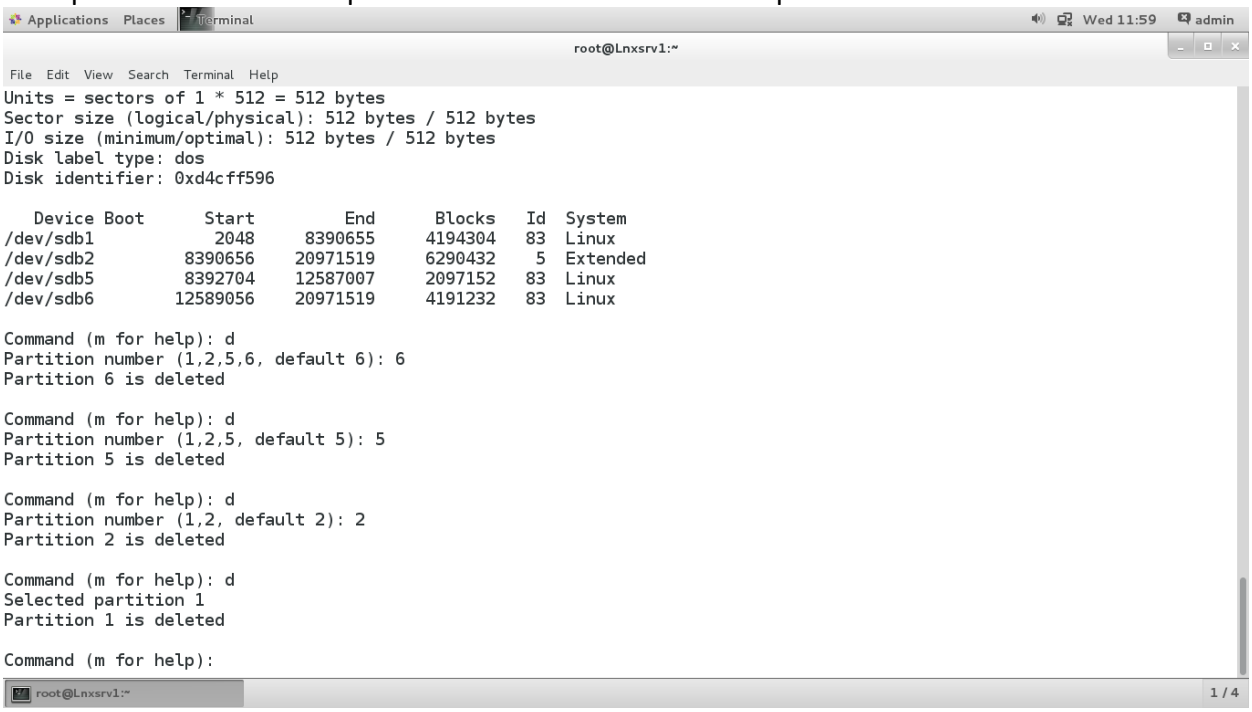
First select the hard disk from which the partitions need to be deleted. Like as shown in the below picture the hard disk /dev/sdb is selected. Thus give command **fdisk /dev/sdb.**
At the fdisk command, prompt type **m** for help. The help shows that **d** is the option to delete a partition.



Type '**p'** at the fdisk command prompt to display the existing partitions on the selected hard disk.

```
[root@Lnxsrv1 ~]# fdisk /dev/sdb
Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.


Command (m for help): p

Disk /dev/sdb: 10.7 GB, 10737418240 bytes, 20971520 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
Disk label type: dos
Disk identifier: 0xd4cff596

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1            2048     8390655     4194304   83  Linux
/dev/sdb2         8390656    20971519     6290432    5  Extended
/dev/sdb5         8392704    12587007     2097152   83  Linux
/dev/sdb6        12589056    20971519     4191232   83  Linux

Command (m for help):
```

Now press **d** to delete the partitions as shown in the below picture.



Once the partitions are deleted press **w** to save the new partition table to the disk and exit the fdisk utility. Check with fdisk -l command from the command prompt and verify that the partitions are deleted.

Note:- After deleting partitions please remove the /etc/fstab file entries for deleted partitions. If the entries remain in the file then after the reboot Linux tries to mount these partitions and fails to find these partitions and it gives error. It may stop further booting process of system. The system may provide a temporary repair command prompt. Using this repair prompt modify the fstab file. Else booting from the Linux DVD and using rescue mode to modify the file is the solution.

Logical Volume Management (LVM)

The Logical Volume Management or LVM is a system that creates an abstraction of storage using the actual physical storage. Thus LVM provides a much flexible way to work with storage. The physical storage brings some restrictions on dynamic management of storage devices and volumes. The LVM overcomes these restrictions and allows a much flexible and efficient way for managing storage.

LVM provides following benefits –

LVM allows you to increase or decrease the volumes. This operation is dynamic in the sense; it does not require repartitioning or reformatting volumes. It allows you to combine the storage space from multiple physical hard disks. It allows adding and removing physical storage as per requirement. LVM supports RAID volumes also. Thus data redundancy can be achieved. It

also allows taking snapshots of logical volumes. This can be used as backup or for testing the results of any modification to data without modifying actual data.

The current version of LVM is 2. It is backward compatible with LVM 1.

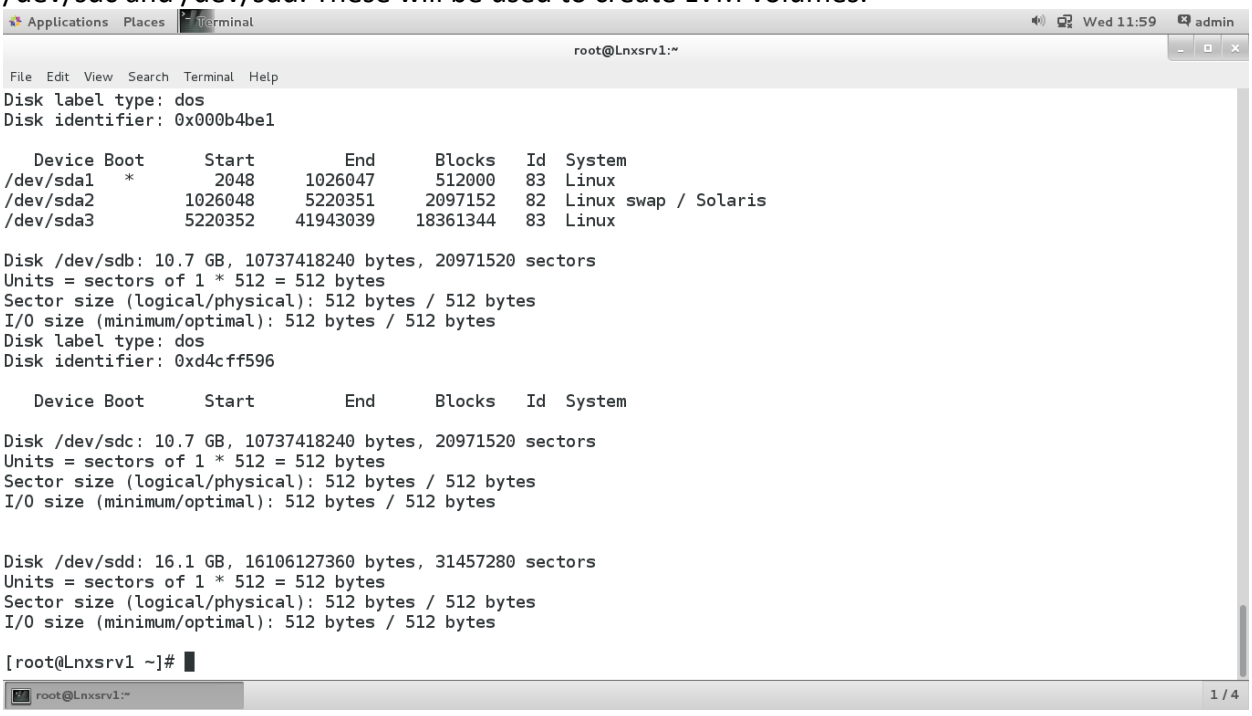
Implementing LVM

1. Creating Physical Volumes

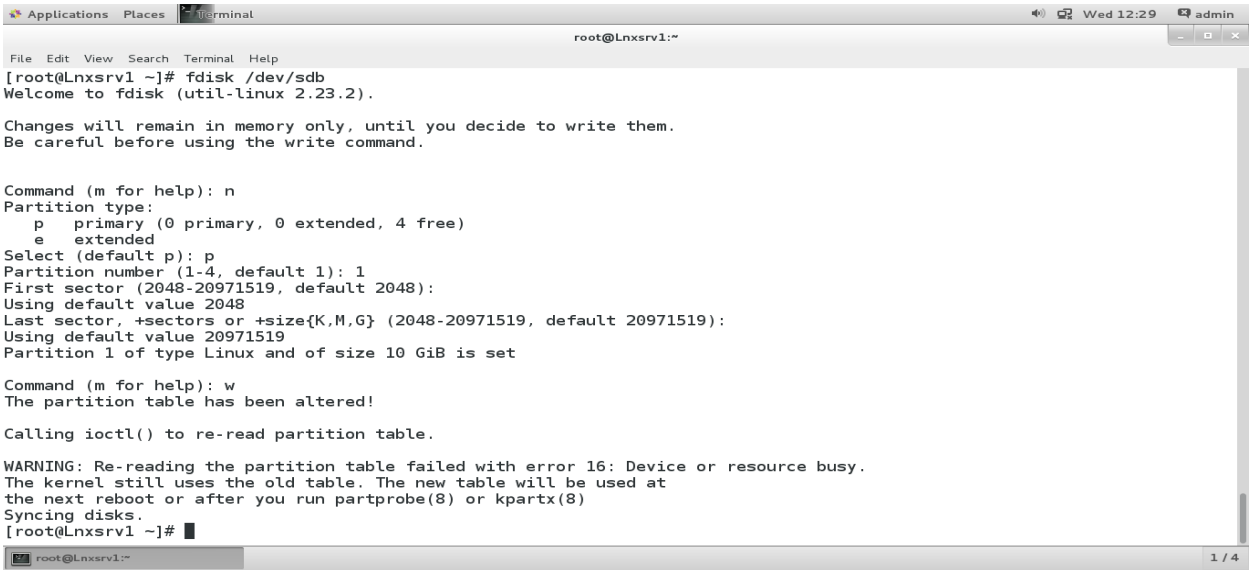
```
[root@Lnxsrv1 ~]# rpm -qa | grep lvm
mesa-private-llvm-3.3-0.8.20131023.el7.x86_64
lvm2-2.02.105-14.el7.x86_64
lvm2-libs-2.02.105-14.el7.x86_64
[root@Lnxsrv1 ~]#
```

Make sure the LVM packages are installed by using above command. Almost all today's Linux flavors' by default install LVM packages and support LVM volumes.

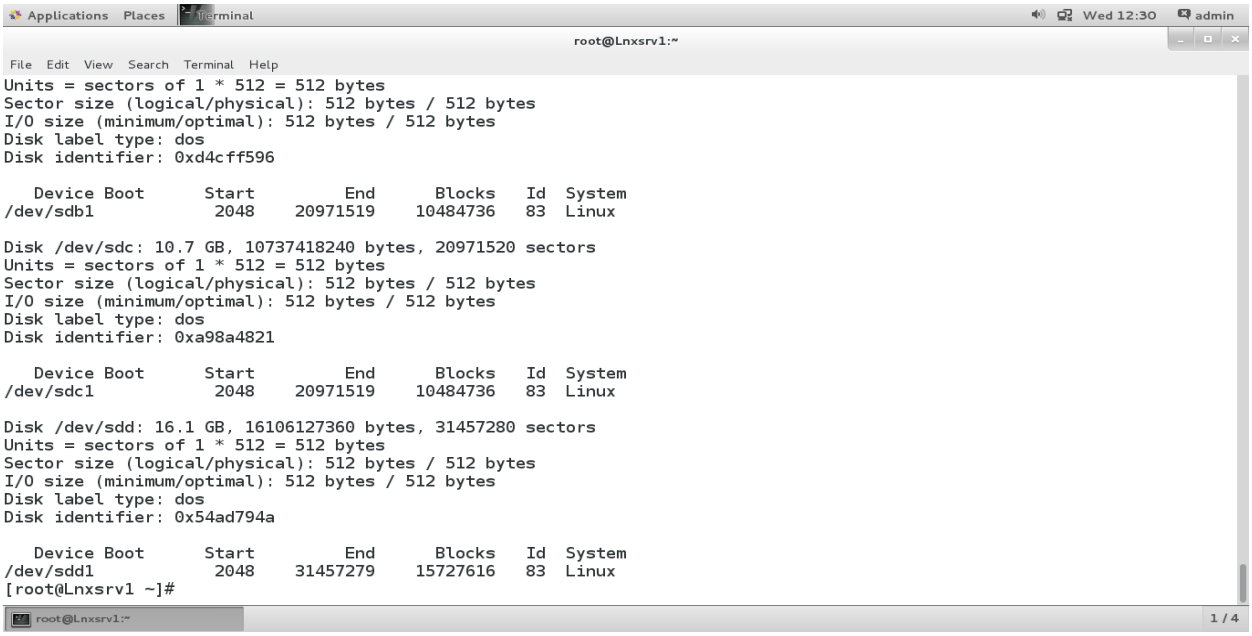
To create LVM volumes attach hard disks to the machine. Once the system starts use **fdisk -l** command to verify that the physical disks are detected by the Linux system. The following picture shows the output of the **fdisk –l** command. Here it shows three hard disks /dev/sdb, /dev/sdc and /dev/sdd. These will be used to create LVM volumes.



First create a single partition consuming all space on all the hard disks that will be part of the LVM. This is shown in the below picture. Use **fdisk /dev/sdb** (the device name may be different other machines). At the fdisk prompt press **n**. Press **p** to create a primary partition. Either press enter to select default value 1 or type 1 and press enter. On next prompt press enter to select the default value for first sector. On next prompt also press enter to select the default value for the last sector. The partition will be created. Press **w** to save partition table and exit fdisk.



Perform the same task for all the other hard disks and create a single partition consuming the entire space. Once done check with 'fdisk –l' command as verify. The output will be similar to shown below.

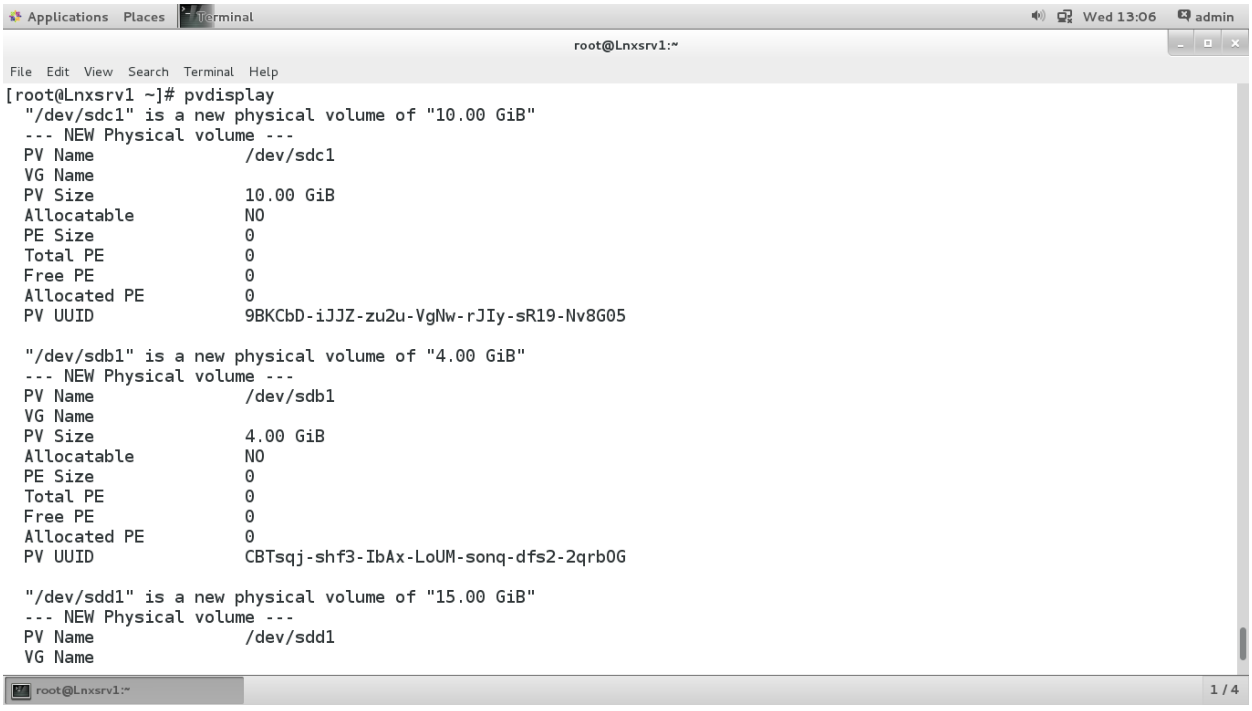


Here all the three hard disks /dev/sdb, /dev/sdc and /dev/sdd show a single partition.

Now these physical partitions need to be converted to physical volumes before they are used to create logical volumes by LVM. The LVM command that performs this operation is **pvcreate.**

```
[root@Lnxsrv1 ~]# pvcreate /dev/sdb1
WARNING: ext4 signature detected on /dev/sdb1 at offset 1080. Wipe it? [y/n] y
  Wiping ext4 signature on /dev/sdb1.
  Physical volume "/dev/sdb1" successfully created
[root@Lnxsrv1 ~]# pvcreate /dev/sdc1 /dev/sdd1
  Physical volume "/dev/sdc1" successfully created
  Physical volume "/dev/sdd1" successfully created
[root@Lnxsrv1 ~]#
```

The pvcreate command can convert a single partition to physical volume or it can convert multiple different partitions to physical volumes at the same time. The first command in the above picture **pvcreate /dev/sdb1** converts the single partition /dev/sdb1 to physical volume. The second command **pvcreate /dev/sdc1 /dev/sdd1** converts both /dev/sdc1 and /dev/sdd1 partitions to physical volume. Once all the partitions are converted to physical volumes, confirm it by using **pvdisplay** command. The output of the pvdisplay command is as shown below.



2. Creating Volume Groups

Next step is to combine the space on these physical volumes to create volume groups. A volume group can combine storage space from various physical volumes to show it as a single storage device. Thus by adding new physical volumes the volume group size can be increased. Also same way by removing a physical volume the size of a volume group can be reduced.

Multiple volume groups can be created in a single system. For example a volume group can be created using the physical volumes /dev/sdb1 and /dev/sdc1. At the same time a second volume group can be created using physical volume /dev/sdd1.

To create a volume group **vgcreate** command is used.
The command **vgcreate volgrp1 /dev/sdb1 /dev/sdc1 /dev/sdd1** creates a volume group by name volgrp1 and combines space from the physical volumes /dev/sdb1, /dev/sdc1 and /dev/sdd1. The command and its output is as shown below.

```
[root@Lnxsrv1 ~]# vgcreate  volgrp1 /dev/sdb1 /dev/sdc1 /dev/sdd1
  Volume group "volgrp1" successfully created
[root@Lnxsrv1 ~]#
```

To confirm the creation of volume group execute the command **vgdisplay.** The vgdisplay command displays all the volume groups created within the system and their information. The output of the vgdisplay command is as below. As the three physical volumes used for this volume group are of 10 GB size. The volume group volgrp1 size is shown as near about 30 GB.

```
[root@Lnxsrv1 ~]# vgdisplay
  --- Volume group ---
  VG Name               volgrp1
  System ID
  Format                lvm2
  Metadata Areas        3
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                3
  Act PV                3
  VG Size               28.99 GiB
  PE Size               4.00 MiB
  Total PE              7421
  Alloc PE / Size       0 / 0
  Free  PE / Size       7421 / 28.99 GiB
  VG UUID               fk0cZ5-pn7v-y1AO-OOyj-PuAa-CAbm-rIrNBC

[root@Lnxsrv1 ~]#
```

Another command related to volume groups is **vgscan**. In case if vgdisplay fails to display a volume group, use vgscan command. The command checks all the hard disk and finds the configured volume groups. The output of **vgscan** command is as shown below.

```
[root@Lnxsrv1 ~]# vgscan
  Reading all physical volumes.  This may take a while...
  Found volume group "volgrp1" using metadata type lvm2
[root@Lnxsrv1 ~]#
```

To rename the volume group created use the command **vgrename**. The command **vgrename volgrp1 datavol1** will rename the volume group volgrp1 to datavol1. The output is shown below.

```
[root@Lnxsrv1 ~]# vgrename volgrp1 datavol1
  Volume group "volgrp1" successfully renamed to "datavol1"
[root@Lnxsrv1 ~]#
```

Verify the name change using vgdisplay command. The output of vgdisplay now showing the volume group name as datavol1 instead of volgrp1 is shown below.

```
[root@Lnxsrv1 ~]# vgdisplay
  --- Volume group ---
  VG Name               datavol1
  System ID
  Format                lvm2
  Metadata Areas        3
  Metadata Sequence No  2
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                3
  Act PV                3
  VG Size               28.99 GiB
  PE Size               4.00 MiB
  Total PE              7421
  Alloc PE / Size       0 / 0
  Free  PE / Size       7421 / 28.99 GiB
  VG UUID               fk0cZ5-pn7v-y1AO-OOyj-PuAa-CAbm-rIrNBC

[root@Lnxsrv1 ~]#
```

3. Creating Logical Volume

As the volume group is created, however they cannot be directly used for storing data. Logical volumes need to be created which will take space from the volume group. A volume group can provide space to multiple logical volumes. Logical volumes are formatted with a file system and are used to store actual data.

The **lvcreate** command creates a logical volume. The **-n** option of the lvcreate command allows to specify name of the new logical volume. The **–L** option of the command decides the size of the volume. Thus the command **lvcreate -n vol1 -L 10G datavol1** will create a logical volume with name vol1 and size of 10GB by using space from the volume group named datavol1. The output of lvcreate command is as below.

```
[root@Lnxsrv1 ~]# lvcreate -n vol1 -L 10G datavol1
  Logical volume "vol1" created
[root@Lnxsrv1 ~]#
```

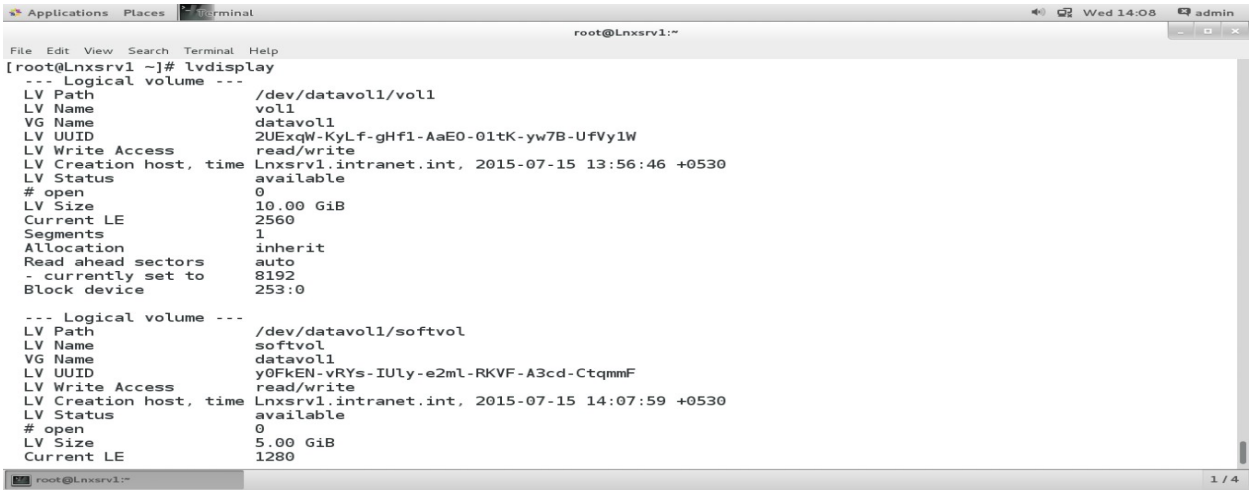
Once the logical volume is created verify using the **lvdisplay** command. This command displays the information about the logical volume. The output of the lvdisplay command is as below.

```
[root@Lnxsrv1 ~]# lvdisplay
  --- Logical volume ---
  LV Path                /dev/datavol1/vol1
  LV Name                vol1
  VG Name                datavol1
  LV UUID                2UExqW-KyLf-gHf1-AaEO-01tK-yw7B-UfVy1W
  LV Write Access        read/write
  LV Creation host, time Lnxsrv1.intranet.int, 2015-07-15 13:56:46 +0530
  LV Status              available
  # open                 0
  LV Size                10.00 GiB
  Current LE             2560
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     8192
  Block device           253:0

[root@Lnxsrv1 ~]#
```

Same way another logical drives can be created as per requirements. The command lvcreate -n softvol -L 5G datavol1 creates another logical volume of 5GB by name softvol in the same volume group datavol1. This is shown below.

```
[root@Lnxsrv1 ~]# lvcreate -n softvol -L 5G datavol1
  Logical volume "softvol" created
[root@Lnxsrv1 ~]#
```

The lvdisplay command after this gives the following output. It displays two logical volumes.



The **lvscan** command also displays the information about the logical volumes.

```
[root@Lnxsrv1 ~]# lvscan
  ACTIVE            '/dev/datavol1/vol1' [10.00 GiB] inherit
  ACTIVE            '/dev/datavol1/softvol' [5.00 GiB] inherit
[root@Lnxsrv1 ~]#
```

As the logical volume is created it can be formatted with any file system and mounted for use. For mounting these logical volumes permanently add their entries in the /etc/fstab file. The following output displays results of formatting both the logical volumes.

The output of mkfs.ext4 /dev/datavol1/vol1 (Formats first logical volume with ext4 file system).

```
[root@Lnxsrv1 ~]# mkfs.ext4  /dev/datavol1/vol1
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
655360 inodes, 2621440 blocks
131072 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=2151677952
80 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

[root@Lnxsrv1 ~]#
```

The output of mkfs.ext4 /dev/datavol1/softvol (Formats second logical volume with ext4 file system).

```
[root@Lnxsrv1 ~]# mkfs.ext4  /dev/datavol1/softvol
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
327680 inodes, 1310720 blocks
65536 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=1342177280
40 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

[root@Lnxsrv1 ~]#
```

The mount command to mount the above volumes is shown below.

```
[root@Lnxsrv1 ~]# mount -t auto /dev/datavol1/vol1 /data1
[root@Lnxsrv1 ~]# mount -t auto /dev/datavol1/softvol /data2
[root@Lnxsrv1 ~]#
```

4. Extending the Logical Volume

Sometimes the volumes used to store the data require more space than allocated. The data getting stored on a volume increases rapidly. Sometimes when users try to create new files on the volume and the operation fails with “no sufficient space” message. Thus now more space is required. On actual physical hard disk if a partition space is full it cannot be extended. A new hard disk with higher capacity needs to be added to the system. A partition needs to be created and formatted on this new disk. Then copy all data from old hard disk to new hard disk. Then the data becomes available to users and they continue with their work. However this procedure requires a lot of time depending on the amount of old data.

This is where the LVM comes to help. In above condition, if the data is stored on a logical volume then within few minutes you can increase the size of the logical volume.

In the scenario used in this demonstration, the volume group datavol1 is of 30GB. The logical volume /dev/datavol1/vol1 is of 10GB. The other logical volume /dev/datavol1/softvol is of 5GB. Thus there is free space of 15GB in the volume group datavol1. Now free space can be assigned from datavol1 to any of these logical volumes to increase their size.

The **lvextend** command extends or increases the size of a logical volume. The command lvextend -L 15G /dev/datavol1/vol1 will increase the size of the vol1 logical volume from 10GB to 15GB.

The first lvscan output shows the /dev/datavol1/vol1 volume size as 10GB

```
[root@Lnxsrv1 ~]# lvscan
  ACTIVE            '/dev/datavol1/vol1' [10.00 GiB] inherit
  ACTIVE            '/dev/datavol1/softvol' [5.00 GiB] inherit
[root@Lnxsrv1 ~]#
```

Now lvextend command is executed to increase the volume size to 15GB. This is shown below.

```
[root@Lnxsrv1 ~]# lvextend -L 15G /dev/datavol1/vol1
  Extending logical volume vol1 to 15.00 GiB
  Logical volume vol1 successfully resized
[root@Lnxsrv1 ~]#
```

Then executing the lvscan command again shows the volume size of the /dev/datavol1/vol1 as 15GB.

```
[root@Lnxsrv1 ~]# lvscan
  ACTIVE            '/dev/datavol1/vol1' [15.00 GiB] inherit
  ACTIVE            '/dev/datavol1/softvol' [5.00 GiB] inherit
[root@Lnxsrv1 ~]#
```

Similarly it is possible to increase a specific logical volume by a specific amount. The following command in the picture below increase the logical volume /dev/datavol1/softvol by 1GB. The above picture shows the size as 5GB for the /dev/datavol1/softvol volume.

```
[root@Lnxsrv1 ~]# lvextend -L +1G /dev/datavol1/softvol
  Extending logical volume softvol to 6.00 GiB
  Logical volume softvol successfully resized
[root@Lnxsrv1 ~]#
```

Use lvscan to confirm that the logical volume size changed from 5GB to 6GB.

```
[root@Lnxsrv1 ~]# lvscan
  ACTIVE            '/dev/datavol1/vol1' [15.00 GiB] inherit
  ACTIVE            '/dev/datavol1/softvol' [6.00 GiB] inherit
[root@Lnxsrv1 ~]#
```

However if checked with the file system command df –kHT the output still shows the /dev/datavol1/vol1 logical volume as 10GB and the /dev/datavol1/softvol of 5GB size.

```
[root@Lnxsrv1 ~]# df -kHT
Filesystem                     Type      Size  Used Avail Use% Mounted on
/dev/sda3                      xfs        19G  5.4G   14G  29% /
devtmpfs                       devtmpfs  509M     0  509M   0% /dev
tmpfs                          tmpfs     518M  152k  518M   1% /dev/shm
tmpfs                          tmpfs     518M  7.5M  511M   2% /run
tmpfs                          tmpfs     518M     0  518M   0% /sys/fs/cgroup
/dev/sda1                      xfs       521M  122M  400M  24% /boot
/dev/mapper/datavol1-vol1      ext4       11G   38M  9.9G   1% /data1
/dev/mapper/datavol1-softvol   ext4      5.2G   21M  4.9G   1% /data2
[root@Lnxsrv1 ~]#
```

This is because the LVM has updated the size information for the logical volumes. However the file system is not aware about this change. Thus using resize2fs command the file system needs to be updated with the change in size of the logical volume.
The command **resize2fs /dev/datavol1/vol1** will update the vol1 logical volume.
Similarly the command **resize2fs /dev/datavol1/softvol** will update the softvol logical volume.

```
[root@Lnxsrv1 ~]# resize2fs /dev/datavol1/vol1
resize2fs 1.42.9 (28-Dec-2013)
Filesystem at /dev/datavol1/vol1 is mounted on /data1; on-line resizing required
old_desc_blocks = 2, new_desc_blocks = 2
The filesystem on /dev/datavol1/vol1 is now 3932160 blocks long.

[root@Lnxsrv1 ~]# resize2fs /dev/datavol1/softvol
resize2fs 1.42.9 (28-Dec-2013)
Filesystem at /dev/datavol1/softvol is mounted on /data2; on-line resizing required
old_desc_blocks = 1, new_desc_blocks = 1
The filesystem on /dev/datavol1/softvol is now 1572864 blocks long.

[root@Lnxsrv1 ~]#
```

Now the df command output will show the modified volume size for both the volumes.

```
[root@Lnxsrv1 ~]# df -kHT
Filesystem                     Type      Size  Used Avail Use% Mounted on
/dev/sda3                      xfs        19G  5.4G   14G  29% /
devtmpfs                       devtmpfs  509M     0  509M   0% /dev
tmpfs                          tmpfs     518M  152k  518M   1% /dev/shm
tmpfs                          tmpfs     518M  7.5M  511M   2% /run
tmpfs                          tmpfs     518M     0  518M   0% /sys/fs/cgroup
/dev/sda1                      xfs       521M  122M  400M  24% /boot
/dev/mapper/datavol1-vol1      ext4       16G   42M   15G   1% /data1
/dev/mapper/datavol1-softvol   ext4      6.3G   21M  5.9G   1% /data2
[root@Lnxsrv1 ~]#
```

5. Reducing Logical Volume

Sometimes there may be volumes which are not used much. However the initial space allotted to them is very big. Thus these volumes can be reduced in size and the space can be returned to the volume group. This space then can be used to extend another logical volume.

However before reducing a logical volume, the file system needs to be resized first. Thus the file system makes sure that all the data is brought within the new reduced size, and then the actual logical volume can be reduced.

When the resize2fs command is executed with size specified and if the volume specified is mounted, resize2fs cannot work. It will give the error as shown below.

```
[root@Lnxsrv1 ~]# resize2fs /dev/datavol1/softvol 4G
resize2fs 1.42.9 (28-Dec-2013)
Filesystem at /dev/datavol1/softvol is mounted on /data2; on-line resizing required
resize2fs: On-line shrinking not supported
[root@Lnxsrv1 ~]#
```

Thus un-mount the required logical volume using **umount** command as shown below. After that again the resize2fs requires e2fsck command to be executed to check for any file system errors. This is shown below.

```
[root@Lnxsrv1 ~]# umount /data2
[root@Lnxsrv1 ~]# resize2fs /dev/datavol1/softvol 4G
resize2fs 1.42.9 (28-Dec-2013)
Please run 'e2fsck -f /dev/datavol1/softvol' first.

[root@Lnxsrv1 ~]# e2fsck -f /dev/datavol1/softvol
e2fsck 1.42.9 (28-Dec-2013)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/datavol1/softvol: 11/393216 files (0.0% non-contiguous), 62574/1572864 blocks
```

Once the e2fsck command executes successfully then resize2fs command will be successful.

```
[root@Lnxsrv1 ~]# resize2fs /dev/datavol1/softvol 4G
resize2fs 1.42.9 (28-Dec-2013)
Resizing the filesystem on /dev/datavol1/softvol to 1048576 (4k) blocks.
The filesystem on /dev/datavol1/softvol is now 1048576 blocks long.

[root@Lnxsrv1 ~]#
```

Now lvreduce command can be executed to reduce the logical volume.

```
[root@Lnxsrv1 ~]# lvreduce -L 4G /dev/datavol1/softvol
  WARNING: Reducing active logical volume to 4.00 GiB
  THIS MAY DESTROY YOUR DATA (filesystem etc.)
Do you really want to reduce softvol? [y/n]: y
  Reducing logical volume softvol to 4.00 GiB
  Logical volume softvol successfully resized
[root@Lnxsrv1 ~]#
```

Use lvscan command to verify the effect.

```
[root@Lnxsrv1 ~]# lvscan
  ACTIVE            '/dev/datavol1/vol1' [15.00 GiB] inherit
  ACTIVE            '/dev/datavol1/softvol' [4.00 GiB] inherit
[root@Lnxsrv1 ~]#
```

Now mount the logical volume and use df command to verify the file system also shows the same size.

```
[root@Lnxsrv1 ~]# mount -t auto  /dev/datavol1/softvol /data2
[root@Lnxsrv1 ~]# df -kHT
Filesystem                   Type      Size  Used Avail Use% Mounted on
/dev/sda3                    xfs        19G  5.4G   14G  29% /
devtmpfs                     devtmpfs  509M     0  509M   0% /dev
tmpfs                        tmpfs     518M  152k  518M   1% /dev/shm
tmpfs                        tmpfs     518M  7.5M  511M   2% /run
tmpfs                        tmpfs     518M     0  518M   0% /sys/fs/cgroup
/dev/sda1                    xfs       521M  122M  400M  24% /boot
/dev/mapper/datavol1-softvol ext4      4.1G   21M  3.9G   1% /data2
[root@Lnxsrv1 ~]#
```

Thus the logical volume /dev/datavol1/softvol was reduced from 6GB to 4GB. The 2 GB space gets added to the volume group datavol1.

6. Adding new Physical Volume.

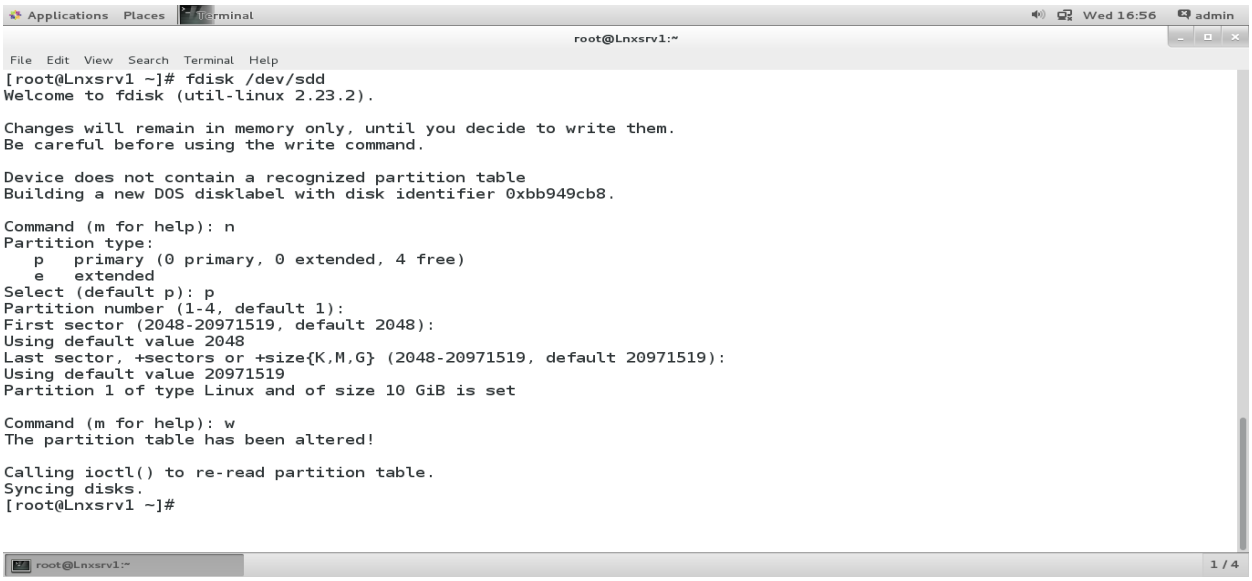
When more space is required new hard disks are added to the system. To increase the space these hard disks first needs to be converted to physical volumes. Then these physical volumes can be added to the required volume group to increase the free space and then to allocate it to the required logical volume.

Use fdisk -l to identify the newly added hard disk.

Here the new hard disk is shown as /dev/sdd as shown below. No partitions present on that hard disk.

```
Disk /dev/sdd: 10.7 GB, 10737418240 bytes, 20971520 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


Disk /dev/sde: 16.1 GB, 16106127360 bytes, 31457280 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
Disk label type: dos
Disk identifier: 0x54ad794a

   Device Boot      Start         End      Blocks   Id  System
/dev/sde1            2048    31457279    15727616   83  Linux
```

Create a single partition to consume the entire space as shown below. The procedure is same as done in the beginning.



Now use the pvcreate command as below.

```
[root@Lnxsrv1 ~]# pvcreate /dev/sdd1
  Physical volume "/dev/sdd1" successfully created
[root@Lnxsrv1 ~]#
```

7. Extending the Volume Group

As the new physical volume is added, it requires to be added to the existing volume group to increase the free space.

The vgdisplay command shows the initial size of the volume group datavol1 as below.

```
[root@Lnxsrv1 ~]# vgdisplay
  --- Volume group ---
  VG Name               datavol1
  System ID
  Format                lvm2
  Metadata Areas        3
  Metadata Sequence No  2
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                3
  Act PV                3
  VG Size               28.99 GiB
  PE Size               4.00 MiB
  Total PE              7421
  Alloc PE / Size       0 / 0
  Free  PE / Size       7421 / 28.99 GiB
  VG UUID               fk0cZ5-pn7v-y1AO-OOyj-PuAa-CAbm-rIrNBC

[root@Lnxsrv1 ~]#
```

The size of the volume group shown is 28.99 GiB (Marked by a red line in the above picture).

The vgextend command is used add a physical volume to a volume group.
The command **vgextend datavol1 /dev/sdd1** adds the physical volume /dev/sdd1 to the volume group datavol1.

```
[root@Lnxsrv1 ~]# vgextend datavol1 /dev/sdd1
  Volume group "datavol1" successfully extended
```

The output of the vgdisplay command after this will show the new volume group size which is 38.98. Thus the size increased by 10GB which is the size of the physical volume.

```
[root@Lnxsrv1 ~]# vgdisplay
  --- Volume group ---
  VG Name               datavol1
  System ID
  Format                lvm2
  Metadata Areas        4
  Metadata Sequence No  8
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                2
  Open LV               0
  Max PV                0
  Cur PV                4
  Act PV                4
  VG Size               38.98 GiB
  PE Size               4.00 MiB
  Total PE              9980
  Alloc PE / Size       4864 / 19.00 GiB
  Free  PE / Size       5116 / 19.98 GiB
  VG UUID               fk0cZ5-pn7v-y1AO-OOyj-PuAa-CAbm-rIrNBC
```

8. Creating a Snapshot Volume.

The commands pvs, vgs and lvs display information about the physical volumes, volume groups and logical volumes respectively. The outputs are as shown below.

```
[root@Lnxsrv1 ~]# pvs
  PV         VG       Fmt  Attr PSize  PFree
  /dev/sdb1  datavol1 lvm2 a--   4.00g  3.99g
  /dev/sdc1  datavol1 lvm2 a--  10.00g  6.00g
  /dev/sdd1  datavol1 lvm2 a--  10.00g 10.00g
  /dev/sde1  datavol1 lvm2 a--  15.00g     0
[root@Lnxsrv1 ~]#
```

```
[root@Lnxsrv1 ~]# vgs
  VG       #PV #LV #SN Attr   VSize  VFree
  datavol1   4   2   0 wz--n- 38.98g 19.98g
[root@Lnxsrv1 ~]#
```

```
[root@Lnxsrv1 ~]# lvs
  LV      VG       Attr       LSize  Pool Origin Data%  Move Log Cpy%Sync Convert
  softvol datavol1 -wi-a-----  4.00g
  vol1    datavol1 -wi-a----- 15.00g
[root@Lnxsrv1 ~]#
```

LVM Snapshot Volumes

LVM provides special type of logical volumes called snapshot volumes. The snapshot volumes are noting but an exact copy of an existing logical volume. The LVM1 provided by Linux Kernel 2.4 supports read-only snapshot volumes. The LVM2 provided by Linux Kernel 2.6 however supports read-write snapshot volumes.

The snapshots are generally used to take a backup of data. When backup of data is required without disturbing the applications running, snapshots are used. As a snapshot contains the exact copy of a logical volume at that point of snapshot creation, the snapshot backup can be taken without disturbing data on original volume. Once the backup completes the snapshot volume can be deleted. Thus whenever backup is required at any point of time just by creating snapshot, backup can be performed without disturbing the operations.

Another use of snapshots is to protect the original data. During new application testing or up gradation process there may be a chance of corrupting the data. Sometimes the results of these operations are unpredictable. In such case snapshots can be used. Instead of real logical volume snapshot of that volume can be mounted and used to perform the operations. If some error takes place and corrupts the data the original logical volume can be mounted and the earlier data becomes available as it is. Also if the operations finish successfully and if expected

results are obtained then you can merge the snapshot volume with the original logical volume so that the updated data is available in the logical volume.

A logical volume stores data using metadata pointers and data blocks. The data blocks contain the actual data. When a snapshot is created the Logical Volume Manager creates a separate volume and copies the metadata pointers of the original volume. But it does not copy the actual data blocks. The data blocks are actually accessed from the original volume when data is accessed from the snapshot volume. As metadata pointers do not require much space thus the snapshot volume size can be very small as compared to the original logical volume size. The snapshot volume starts growing as you modify data on the original volume. The data in the blocks which are getting modified are first copied to the snapshot volume and then modified in the original volume. Thus snapshot stores the backup data of the time when it was created. For all unmodified data the Meta data pointers point to the original volume data blocks. Thus as more and more data gets modified on the original volume, more and more data blocks are written to the snapshot volume. Thus snapshot volume grows. Thus while creating a snapshot volume based on the amount of data that will be modified in the original volume, the size of the snapshot volume should be decided. If the snapshot logical volume becomes full it is dropped and cannot be used. So the snapshot volumes should be monitored and should be extended if they are getting full.

Creating a snapshot volume.

Following command displays the current logical volumes in the system.

```
[root@Lnxsrv1 ~]# lvs
  LV      VG       Attr       LSize  Pool Origin Data%  Move Log Cpy%Sync Conver
t
  softvol datavol1 -wi-ao----  4.00g

  vol1    datavol1 -wi-a----- 15.00g

[root@Lnxsrv1 ~]#
```

Thus there are only two logical volumes. Let's create a snapshot volume for softvol logical volume. But before that we will mount the softvol volume to data1 folder and create a file named test1 as shown below.

```
[root@Lnxsrv1 ~]# mount -t auto /dev/datavol1/softvol /data1
[root@Lnxsrv1 data1]# ls
lost+found
[root@Lnxsrv1 data1]# cat >> test1
This is line is added before creating the snapshot.
This file is a test for snapshot.
^Z
[1]+  Stopped                 cat >> test1
[root@Lnxsrv1 data1]# cat test1
This is line is added before creating the snapshot.
This file is a test for snapshot.
[root@Lnxsrv1 data1]#
```

Now using lvcreate command snapshot will be created. The –s option tells the lvcreate command to create a snapshot volume. The –L option decides the size of the snapshot volume

to be created. The -n options specifies the name for the snapshot volume. The last volume name tells the lvcreate command to use it as the origin of the snapshot volume. Once the lvcreate command is executed the snapshot volume is created. The lvs command displays the new snapshot volume. This is shown in the below picture.

```
[root@Lnxsrv1 data1]# lvcreate -s -L1G -n softvol-snap /dev/datavol1/softvol
  Logical volume "softvol-snap" created
[root@Lnxsrv1 data1]# lvs
  LV           VG       Attr       LSize  Pool Origin  Data%  Move Log Cpy%Sync
Convert
  softvol      datavol1 owi-aos---  4.00g

  softvol-snap datavol1 swi-a-s---  1.00g      softvol   0.00

  vol1         datavol1 -wi-a----- 15.00g

[root@Lnxsrv1 data1]#
```

Let's mount the new snapshot volume named softvol-snap volume in a directory named data1bak and check the data in it.

```
[root@Lnxsrv1 data1]# mount -t auto /dev/datavol1/softvol-snap /data1bak
[root@Lnxsrv1 data1]# cd /data1bak
[root@Lnxsrv1 data1bak]# ls
lost+found  test1
[root@Lnxsrv1 data1bak]# cat test1
This is line is added before creating the snapshot.
This file is a test for snapshot.
```

Thus the data in both the volumes is same. Now let's create a file in the original volume and see if it appears in the snap-shot volume. Thus as shown in the below picture a **test2** file is created in the /data1 directory where the original volume softvol is mounted. However the test2 file does not appear in the data1bak folder where the snapshot volume is mounted.

```
[root@Lnxsrv1 data1bak]# cd /data1
[root@Lnxsrv1 data1]# ls
lost+found  test1
[root@Lnxsrv1 data1]# cat >> test2
This file is created after the snapshot is created.
Test for snapshot.
^Z
[2]+  Stopped                 cat >> test2
[root@Lnxsrv1 data1]# cd /data1bak
[root@Lnxsrv1 data1bak]# ls
lost+found  test1
[root@Lnxsrv1 data1bak]# cd /data1
[root@Lnxsrv1 data1]# ls
lost+found  test1  test2
[root@Lnxsrv1 data1]#
```

This shows that snapshot is a picture of the data in the original volume at the point it is taken. Any modification to data within any of these volumes remains local and is not replicated to other.

Merging snapshot volume with original

As discussed earlier if a snapshot volume is used for modifying data then may be after successful testing it is required that the updated data becomes available from the original volume. In such case the **merge** option of the **lvconvert** command can be used.

If both the volumes are not mounted then the merge will take place immediately. However if the original volume is mounted then the merger will be deferred until the next time the original volume is mounted.

Let's see how to merge two volumes.

The following careen shows the two volumes mounted in /data1 and /data1bak directories.

```
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw,relatime)
sunrpc on /proc/fs/nfsd type nfsd (rw,relatime)
/dev/sda1 on /boot type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
fusectl on /sys/fs/fuse/connections type fusectl (rw,relatime)
gvfsd-fuse on /run/user/1000/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,relatime
,user_id=1000,group_id=1000)
/dev/mapper/datavol1-softvol on /data1 type ext4 (rw,relatime,seclabel,data=orde
red)
/dev/mapper/datavol1-softvol--snap on /data1bak type ext4 (rw,relatime,seclabel,
data=ordered)
[root@Lnxsrv1 data1bak]#
```

The following screen displays the contents of these two volumes.

```
[root@Lnxsrv1 data1]# ls
file1  lost+found  test1
[root@Lnxsrv1 data1]# cd /data1bak
[root@Lnxsrv1 data1bak]# ls
file1  lost+found  test1
[root@Lnxsrv1 data1bak]#
```

Thus now both the volumes contain same data. Let's create some new files on the snapshot volume i.e. in /data1bak directory. The following screen displays this action.

```
[root@Lnxsrv1 data1bak]# pwd
/data1bak
[root@Lnxsrv1 data1bak]# cat >> file3
file created after snapshot
test
^Z
[1]+  Stopped                 cat >> file3
[root@Lnxsrv1 data1bak]# cat >> file4
Test for snapshot migration
test
^Z
[2]+  Stopped                 cat >> file4
[root@Lnxsrv1 data1bak]# ls
file1  file3  file4  lost+found  test1
[root@Lnxsrv1 data1bak]# cd /data1
[root@Lnxsrv1 data1]# ls
file1  lost+found  test1
[root@Lnxsrv1 data1]#
```

The above screen also displays that the files created in snapshot volume does not appear in original volume. Thus the snapshot volume now contains file3 and file4 additionally. Now um mount both the volumes so that merging of volumes can be done.

```
[admin@Lnxsrv1 Desktop]$ su -
Password:
Last login: Mon Jul 20 17:57:27 IST 2015 on pts/0
[root@Lnxsrv1 ~]# umount /data1bak
[root@Lnxsrv1 ~]# umount /data1
```

The above screen displays the command used to unmount the volumes.

Now use the **lvconvert** command with **–merge** option to merge the snapshot volume and the original volume as shown below.

```
[root@Lnxsrv1 ~]# lvconvert --merge /dev/datavol1/softvol-snap
  Merging of volume softvol-snap started.
  softvol: Merged: 100.0%
  softvol: Merged: 100.0%
  Merge of snapshot into logical volume softvol has finished.
  Logical volume "softvol-snap" successfully removed
[root@Lnxsrv1 ~]# lvs
  LV      VG       Attr       LSize  Pool Origin Data%  Move Log Cpy%Sync Conver
t
  softvol datavol1 -wi-a-----  4.00g

  vol1    datavol1 -wi-a----- 15.00g

[root@Lnxsrv1 ~]# mount -t auto /dev/datavol1/softvol /data1
```

After the lvconvert command executes successfully, the **lvs** command will display that the snapshot volume **softvol-snap** is deleted. Now mount the original volume with the last command shown in the above screen. The data in the /data1 folder is as below.

```
[root@Lnxsrv1 ~]# cd /data1
[root@Lnxsrv1 data1]# ls
file1  file3  file4  lost+found  test1
[root@Lnxsrv1 data1]#
```

Thus the updated data within the snapshot volume is merged to the original volume.
Note:- Any parallel updating to the original volume data after the snapshot volume is created will be lost while merging the snapshot volume.

Deleting a snapshot volume.

The lvs command displays the snapshot volume. In this case the volume is softvol-snap. The lvremove command is used to remove a logical volume. But before removing a logical volume make sure it is unmounted. Once the lvremove command is successful, the lvs command output will not display the snapshot volume as it is now removed. The following screen displays the commands and their output.

```
[root@Lnxsrv1 data1]# lvs
  LV           VG       Attr       LSize  Pool Origin  Data%  Move Log Cpy%Sync
Convert
  softvol      datavol1 owi-aos---  4.00g

  softvol-snap datavol1 swi-a-s---  1.00g      softvol   0.00

  vol1         datavol1 -wi-a----- 15.00g

[root@Lnxsrv1 data1]# lvremove /dev/datavol1/softvol-snap
Do you really want to remove active logical volume softvol-snap? [y/n]: y
  Logical volume "softvol-snap" successfully removed
[root@Lnxsrv1 data1]# lvs
  LV      VG       Attr       LSize  Pool Origin Data%  Move Log Cpy%Sync Conver
t
  softvol datavol1 -wi-ao----  4.00g

  vol1    datavol1 -wi-a----- 15.00g

[root@Lnxsrv1 data1]#
```

Removing all LVM components

First unmount all the logical volumes. Then using lvremove command remove all the logical volumes. Check the successful deletion using lvs or lvdisplay command. Thefollowing screen displays the commands and their output.

```
[root@Lnxsrv1 data1]# cd
[root@Lnxsrv1 ~]# umount /data1
[root@Lnxsrv1 ~]# lvs
  LV      VG       Attr       LSize  Pool Origin Data%  Move Log Cpy%Sync Conver
t
  softvol datavol1 -wi-a-----  4.00g

  vol1    datavol1 -wi-a----- 15.00g

[root@Lnxsrv1 ~]# lvremove /dev/datavol1/softvol
Do you really want to remove active logical volume softvol? [y/n]: y
  Logical volume "softvol" successfully removed
[root@Lnxsrv1 ~]# lvremove /dev/datavol1/vol1
Do you really want to remove active logical volume vol1? [y/n]: y
  Logical volume "vol1" successfully removed
[root@Lnxsrv1 ~]# lvs
[root@Lnxsrv1 ~]# lvdisplay
[root@Lnxsrv1 ~]#
```

Once all the logical volumes are deleted then delete the volume groups. The following screen displays the commands and output of these commands.

```
[root@Lnxsrv1 ~]# vgs
  VG       #PV #LV #SN Attr   VSize  VFree
  datavol1   4   0   0 wz--n- 38.98g 38.98g
[root@Lnxsrv1 ~]# vgremove datavol1
  Volume group "datavol1" successfully removed
[root@Lnxsrv1 ~]# vgs
  No volume groups found
[root@Lnxsrv1 ~]# vgdisplay
  No volume groups found
[root@Lnxsrv1 ~]#
```

Once all the volume groups are removed then remove the physical volumes. The following screen displays the procedure.

```
[root@Lnxsrv1 ~]# pvs
  PV         VG   Fmt  Attr PSize  PFree
  /dev/sdb1       lvm2 a--  10.00g 10.00g
  /dev/sdc1       lvm2 a--  10.00g 10.00g
  /dev/sdd1       lvm2 a--  10.00g 10.00g
  /dev/sde1       lvm2 a--  15.00g 15.00g
[root@Lnxsrv1 ~]# pvremove /dev/sdb1
  Labels on physical volume "/dev/sdb1" successfully wiped
[root@Lnxsrv1 ~]# pvremove /dev/sdc1
  Labels on physical volume "/dev/sdc1" successfully wiped
[root@Lnxsrv1 ~]# pvremove /dev/sdd1
  Labels on physical volume "/dev/sdd1" successfully wiped
[root@Lnxsrv1 ~]# pvremove /dev/sde1
  Labels on physical volume "/dev/sde1" successfully wiped
[root@Lnxsrv1 ~]# pvs
[root@Lnxsrv1 ~]# pvdisplay
[root@Lnxsrv1 ~]#
```