**File system**



**Superblock :**

The **superblock** is essentially file system metadata and defines the file system type, size, status, and information about other metadata structures (metadata of metadata). The superblock is very critical to the file system and therefore is stored in multiple redundant copies for each file system.

Following command displays primary and backup superblock location on /dev/sda3:

**# dumpe2fs /dev/hda3 | grep -i superblock**  
Output:

Primary superblock at 0, Group descriptors at 1-1

Backup superblock at 32768, Group descriptors at 32769-32769

Backup superblock at 98304, Group descriptors at 98305-98305

Backup superblock at 163840, Group descriptors at 163841-163841

Backup superblock at 229376, Group descriptors at 229377-229377

Backup superblock at 294912, Group descriptors at 294913-294913

**Linux: Recover Corrupted Partition From A Bad Superblock**

# fsck -b 32768 /dev/sda2  
Sample output:

fsck 1.40.2 (12-Jul-2007)

e2fsck 1.40.2 (12-Jul-2007)

/dev/sda2 was not cleanly unmounted, check forced.

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

Free blocks count wrong for group #241 (32254, counted=32253).

Fix? yes

Free blocks count wrong for group #362 (32254, counted=32248).

Fix? yes

Free blocks count wrong for group #368 (32254, counted=27774).

Fix? yes

..........

/dev/sda2: \*\*\*\*\* FILE SYSTEM WAS MODIFIED \*\*\*\*\*

/dev/sda2: 59586/30539776 files (0.6% non-contiguous), 3604682/61059048 blocks

Now try to mount file system using mount command:  
# mount /dev/sda2 /mnt  
You can also use superblock stored at 32768 to mount partition, enter:  
# mount sb={alternative-superblock} /dev/device /mnt  
# mount sb=32768 /dev/sda2 /mnt  
Try to browse and access file system:  
# cd /mnt  
# mkdir test  
# ls -l  
# cp file /path/to/safe/location

You should always keep backup of all important data including configuration files.

## inode definition

An inode is a data structure on a traditional Unix-style file system such as UFS or ext3. An inode stores basic information about a regular file, directory, or other file system object.

## How do I see file inode number?

You can use ls -i command to see inode number of file  
$ ls -i /etc/passwd  
Sample Output

32820 /etc/passwd

You can also use stat command to find out inode number and its attribute:  
$ stat /etc/passwd

=> File type (executable, block special etc)  
=> Permissions (read, write etc)  
=> Owner  
=> Group  
=> File Size  
=> File access, change and modification time (remember UNIX or Linux never stores file creation time, this is favorite question asked in UNIX/Linux sys admin job interview)  
=> File deletion time  
=> Number of links (soft/hard)  
=> Extended attribute such as append only or [no one can delete file](https://www.cyberciti.biz/tips/linux-password-trick.html) including [root user (immutability)](https://www.cyberciti.biz/tips/howto-write-protect-file-with-immutable-bit.html)  
=> Access Control List (ACLs)

All the above information stored in an inode.

**dentry**

A **dentry** is the glue that holds inodes and files together by relating inode numbers to file names. Dentries also play a role in directory caching which, ideally, keeps the most frequently used files on-hand for faster access. File system traversal is another aspect of the dentry as it maintains a relationship between directories and their files.

Surviving a Linux Filesystem Failures

When you use term filesystem failure, you mean corrupted filesystem [data structures (or objects](https://www.cyberciti.biz/nixcraft/vivek/blogger/2005/11/understanding-unixlinux-file-system.php) such as inode, directories, [superblock](https://www.cyberciti.biz/nixcraft/vivek/blogger/2005/11/understanding-unixlinux-filesystem.php) etc. This can be caused by any one of the following reason:

\* Mistakes by Linux/UNIX Sys admin  
\* Buggy device driver or utilities (especially third party utilities)  
\* Power outage (very rarer on production system) due to UPS failure  
\* Kernel bugs (that is why you don’t run latest kernel on production Linux/UNIX system, most of time you need to use stable kernel release)

Due to filesystem failure:

* File system will refuse to mount
* Entire system get hangs
* Even if filesystem mount operation result into success, users may notice strange behavior when mounted such as system reboot, gibberish characters in directory listings etc

So how the hell you are gonna Surviving a Filesystem Failures? Most of time fsck (front end to ext2/ext3 utility) can fix the problem, first simply run e2fsck – to check a Linux ext2/ext3 file system (assuming /home [/dev/sda3 partition] filesystem for demo purpose), first unmount /dev/sda3 then type following command :  
# e2fsck -f /dev/sda3  
Where,

* -f : Force checking even if the file system seems clean.

Please note that If the [superblock is not found](https://www.cyberciti.biz/nixcraft/vivek/blogger/2005/11/understanding-unixlinux-filesystem.php), e2fsck will terminate with a fatal error. However Linux maintains multiple redundant copies of the superblock in every file system, so you can use -b {alternative-superblock} option to get rid of this problem. The location of the backup superblock is dependent on the filesystem’s blocksize:

* For filesystems with 1k blocksizes, a backup superblock can be found at block 8193
* For filesystems with 2k blocksizes, at block 16384
* For 4k blocksizes, at block 32768.

Tip you can also try any one of the following command(s) to determine alternative-superblock locations:  
# mke2fs -n /dev/sda3  
OR  
# dumpe2fs /dev/sda3|grep -i superblock  
To repair file system by alternative-superblock use command as follows:  
# e2fsck -f -b 8193 /dev/sda3

However it is highly recommended that you make backup before you run fsck command on system, use dd command to create a backup (provided that you have spare space under /disk2)  
# dd if=/dev/sda2 of=/disk2/backup-sda2.img

**/etc/fstab**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| /dev/hda2 | / | ext4 | defaults | 1 1 |
| /dev/hdb1 | /home | ext4 | defaults | 1 2 |
| /dev/cdrom | /media/cdrom | auto | ro,noauto,user,exec | 0 0 |
| /dev/fd0 | /media/floppy | auto | rw,noauto,user,sync | 0 0 |
| proc | /proc | proc | defaults | 0 0 |
| /dev/hda1 | swap | swap | pri=42 | 0 0 |

## 1st and 2nd columns: Device and default mount point

## 3rd column: Filesystem type(ext3,ext4,Ext4, ReiserFS, swap, auto)

## 4th column: Mount options

🡪 **auto and noauto:**With the auto option, the device will be mounted automatically at bootup

🡪 **user and nouser:**These are very useful options. The user option allows normal users to mount the device, whereas**nouser** lets only the root to mount the device. **nouser** is the default.

🡪 **exec and noexec:** **exec** lets you execute binaries that are on that partition, whereas**noexec** doesn't let you do that. **noexec** might be useful for a partition that contains binaries you don't want to execute on your system. Default exec.

**ro:**Mount the filesystem read-only.  
  
**rw:**Mount the filesystem read-write.

**sync and async:**How the input and output to the filesystem should be done. **sync** means it's done synchronously. **async** option in **/etc/fstab**, input and output is done asynchronously. **async** is the default.

**defaults:**Uses the default options that are rw, suid, dev, exec, auto, nouser, and async.

## 5th and 6th columns: Dump and fsck options

Dump checks it and uses the number to decide if a filesystem should be backed up. If it's zero, dump will ignore that filesystem. If you take a look at the example fstab , you'll notice that the 5th column is zero in most cases. **sck** option. **fsck** looks at the number in the 6th column to determine in which order the filesystems should be checked. If it's zero, fsck won't check the filesystem.

**#tune2fs -l /dev/md1**  
tune2fs 1.41.12 (17-May-2010)  
Filesystem volume name:   <none>  
Last mounted on:          <not available>  
Filesystem UUID:          957e1766-4723-420e-b942-a5dafe67c661  
Filesystem magic number:  0xEF53  
Filesystem revision #:    1 (dynamic)  
Filesystem features:      has\_journal ext\_attr resize\_inode dir\_index filetype sparse\_super large\_file  
Filesystem flags:         signed\_directory\_hash   
Default mount options:    (none)  
Filesystem state:         clean  
Errors behavior:          Continue  
Filesystem OS type:       Linux  
Inode count:              30539776  
Block count:              61048976  
Reserved block count:     3052448  
Free blocks:              19734533  
Free inodes:              30508949  
First block:              0  
Block size:               4096  
Fragment size:            4096  
Reserved GDT blocks:      1009  
Blocks per group:         32768  
Fragments per group:      32768  
Inodes per group:         16384  
Inode blocks per group:   512  
Filesystem created:       Tue Sep 27 11:09:10 2011  
Last mount time:          Fri Aug 21 23:28:40 2015  
Last write time:          Fri Aug 21 23:33:36 2015  
Mount count:              1  
Maximum mount count:      32  
Last checked:             Thu Aug 20 23:19:10 2015  
Check interval:           15552000 (6 months)  
Next check after:         Tue Feb 16 22:19:10 2016  
Reserved blocks uid:      0 (user root)  
Reserved blocks gid:      0 (group root)  
First inode:              11  
Inode size:              128  
Journal inode:            8  
Default directory hash:   tea  
Directory Hash Seed:      0aa9310b-d299-4d55-a83b-e4eca6c0a6f2  
Journal backup:           inode blocks

### Set  volume label for a filesystem

The option -L volume-label is to set the volume label of the filesystem

#tune2fs -L data /dev/md1  
tune2fs 1.41.12 (17-May-2010)  
#tune2fs -l /dev/md1 | grep volume  
Filesystem volume name:   data

### Set number of counts and time interval

The options -c adjust the number of mounts after which the filesystem will be checked, while the option adjust the maximal time between two filesystem checks.

#tune2fs -c 8 -i 15 /dev/md1  
tune2fs 1.41.12 (17-May-2010)  
Setting maximal mount count to 8 #default is 32  
Setting interval between checks to 1296000 seconds #default is 6 month

### Set error behavior

The option -e allows you to change the behavior of the kernel code when errors are detected.  In all cases, a filesystem error will cause fsck to check the filesystem on the next boot.   error-behavior  can  be one of the following:

   continue    Continue normal execution. #default mode  
   remount-ro  Remount filesystem read-only.  
  panic       Cause a kernel panic.

### Enable speeding lookups

The option dir\_index option, which is turned off by default, adds a balanced tree(B-tree) binary hash lookup method for directories. This feature improves scalability of directories with large numbers of files, although it means that the hash needs to be updated each time a directory changes.

To turn it on, using

#tune2fs -O dir\_index

And reboot to create the hash.

## Change an ext2 filesystem to an ext3 filesystem

To change an ext2 filesystem to an ext3 filesystem, you must put a journal on the filesystem, and the kernel must support ext3 filesystems.

Use the option -j to set up a journal on an unmounted filesystem:

#tune2fs -j /dev/md2

Also change the entry in the fstab to reflect its new type.

### To change an ext3 filesystem to an ext2 filesystem

#tune2fs -O ^has\_journal /dev/md2

**-c max-mount-counts**

**Adjust the number of mounts after which the filesystem will be checked by e2fsck(8).**

**If max-mount-counts is 0 or -1, the number of times the filesystem is mounted will be disregarded by e2fsck(8) and the kernel.**

**-C mount-count**

**Set the number of times the filesystem has been mounted. If set to a greater value than the max-mount-counts parameter set by the -c option, e2fsck(8) will check the filesystem at the next reboot.**

**-i interval-between-checks[d|m|w]**

**Adjust the maximal time between two filesystem checks. No postfix or d result in days,m in months, and w in weeks. A value of zero will disable the time-dependent checking.**

**-l List the contents of the filesystem superblock, including the current values of the parameters that can be set via this program.**

**-m reserved-blocks-percentage**

**Set the percentage of the filesystem which may only be allocated by privileged pro-cesses. Reserving some number of filesystem blocks for use by privileged processes is done to avoid filesystem agmentation, and to allow system daemons, such as sys-logd(8), to continue to function correctly after non-privileged processes are prevented from writing to the filesystem. Normally, the default percentage of reserved blocks is 5%.**

**-U UUID Set the universally unique identifier (UUID) of the filesystem to UUID.**

# Converts ext2 to ext3 : tune2fs -j /dev/sdX

**tune2fs -O extents,uninit\_bg,dir\_index /dev/yourpartition**

**Migrate existing Ext3 filesystems to Ext4**

**-o** [^]*mount-option*[,...]

Set or clear the indicated default mount options in the filesystem. Default mount options can be overridden by mount options specified either in **/etc/fstab**

Mount options prefixed with a caret character (’^’) will be cleared in the filesystem’s superblock; mount options without a prefix character or prefixed with a plus character (’+’) will be added to the filesystem.

|  |  |
| --- | --- |
| **debug** | Enable debugging code for this filesystem. |
| **bsdgroups** | |
|  | Emulate BSD behaviour when creating new files: they will take the group-id of the directory in which they were created. The standard System V behaviour is the default, where newly created files take on the fsgid of the current process, unless the directory has the setgid bit set, in which case it takes the gid from the parent directory, and also gets the setgid bit set if it is directory itself. |
| **user\_xattr** | |
|  | Enable user-specified extended attributes. |
| **acl** | Enable Posix Access Control Lists. |
| **uid16** | Disables 32-bit UIDs and GIDs. This is for interoperability with older kernels which only store and expect 16-bit values. |
| **journal\_data** | |
|  | When the filesystem is mounted with journalling enabled, all data (not just metadata) is committed into the journal prior to being written into the main filesystem. |
| **journal\_data\_ordered** | |
|  | When the filesystem is mounted with journalling enabled, all data is forced directly out to the main file system prior to its metadata being committed to the journal. |
| **journal\_data\_writeback** | |
|  | When the filesystem is mounted with journalling enabled, data may be written into the main filesystem after its metadata has been committed to the journal. This may increase throughput, however, it may allow old data to appear in files after a crash and journal recovery. |

**-O** [^]*feature*[,...]

Set or clear the indicated filesystem features (options) in the filesystem. More than one filesystem feature can be cleared or set by separating features with commas. Filesystem features prefixed with a caret character (’^’) will be cleared in the filesystem’s superblock; filesystem features without a prefix character or prefixed with a plus character (’+’) will be added to the filesystem.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Tag** | **Description** | | **dir\_index** | | |  | Use hashed b-trees to speed up lookups in large directories. | | **filetype** | | |  | Store file type information in directory entries. | | **has\_journal** | | |  | Use a journal to ensure filesystem consistency even across unclean shutdowns. Setting the filesystem feature is equivalent to using the **-j** option. | | **sparse\_super** | | |  | Limit the number of backup superblocks to save space on large filesystems. | |
| After setting or clearing **sparse\_super** and **filetype** filesystem features, **e2fsck**(8) must be run on the filesystem to return the filesystem to a consistent state. **Tune2fs** will print a message requesting that the system administrator run **e2fsck**(8) if necessary. After setting the **dir\_index** feature, **e2fsck -D** can be run to convert existing directories to the hashed B-tree format. |

|  |  |
| --- | --- |
| **-s** [**0**|**1**] | Turn the sparse super feature off or on. Turning this feature on saves space on really big filesystems. This is the same as using the **-O sparse\_super** option. |

|  |  |
| --- | --- |
| **-u***user* | Set the user who can use the reserved filesystem blocks. *user* can be a numerical uid or a user name. If a user name is given, it is converted to a numerical uid before it is stored in the superblock. |

## Use fsck command to check and repair filesystems on linux

The fsck(FileSystem ChecK) utility verifies the integrity of many types of filesystems and, if possible, repairs problems it finds. Repairs can destroy data, particularly on nonjournaling filesystems, such as ext2, so by default fsck asks you for confirmation before making each repair.

Note:  You can run fsck with option -N(no write) on a mounted filesystem. But DO NOT run fsck on a mounted filesystem, more likely you will harm the filesystem if do so. This is because when fsck is rearranging the underlying structure through the raw device, another process could change a disk block using the block device, resulting in a corrupted filesystem.

Depends on the fsck option in[/etc/fstab](http://www.fibrevillage.com/sysadmin/312-fstab-keeps-track-of-filesystems) and the last time the filesystem was checked, during booting, system runs fsck to check whether the filesystem is in consistent state.

Here are just few examples:

### Check a filesystem

To check a filesystem named /data, you can either run fsck directly to the filesystem.

#fsck /data  
fsck from util-linux-ng 2.17.2  
e2fsck 1.41.12 (17-May-2010)  
/dev/md1 has gone 835 days without being checked, check forced.  
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/md1: 30827/30539776 files (2.7% non-contiguous), 41314443/61048976 blocks

Or to its filesystem device

#fsck /dev/md1

### Check all umounted filesystem

#fsck -AR

The -A option let fsck to check filesystems listed in fstab, the -R option causes fsck not to check the root filesystem.

### Check and fix filesystem without asking.

#fsck /data -y

By default fsck asks for confirmation before making repair, the option -y is to say yes to all repairs.

The opposite option is -n, which is to avoid attempting to repair.

Note that not all filesystem-specific checkers implement -y option.  In particular fsck.minix(8) and fsck.cramfs(8) does not support the  -y option as of this writing.

### Check only specified filesystem type

#fsck -AR -t ext3 -y

The command above will check and repair all ext2 filesystems listed in /etc/fstab.

Further more, you can specify a list of filesystems to the option -t, like

-t fslist

fslist is a The fslist parameter is a comma-separated list of filesystems and options specifiers. The options specifiers must have the format opts=fs-option

#fsck -AR -t ext3,opts=ro

The example above causes fsck to check rdonly ext3 filesystems in /etc/fstab

### Turn on verbose output

#fsck -V /data

### Force to check a filesystem even it's in clean state

The option -f forces fsck to check a filesystem even it's in clean state.

### Fsck exit code

The exit code returned by fsck is the sum of the following conditions:  
    0    - No errors  
    1    - File system errors corrected  
    2    - System should be rebooted  
    4    - File system errors left uncorrected  
    8    - Operational error  
    16   - Usage or syntax error  
    32   - Fsck canceled by user request  
  128  - Shared library error  
The exit code returned when multiple file systems are checked is the bit-wise OR of the exit codes for each file system that is checked.

How to display who/what is acting on a filesystem?

Ans.            # fuser  -v  mnt\_point

    How to kill all actions on a filesystem?

Ans.                  # fuser  -km  mnt\_point

# Parted’ Commands

#### Check Parted Version : parted

#### List Linux Disk Partitions : print

#### List or Switch to Different Disk : select /dev/sdX

#### Create Primary or Logical Partition in Linux mklabel name

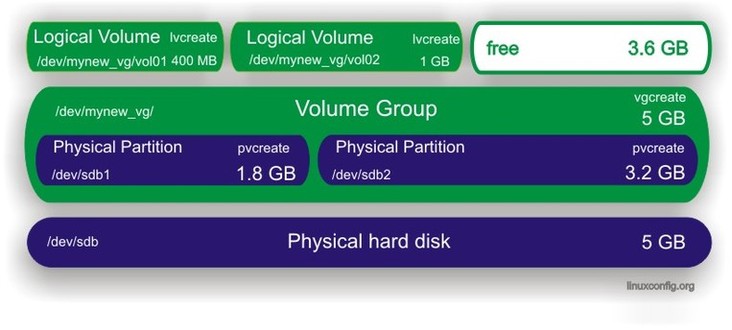
Mkpart 🡪 to create

RESIZE 🡪 resizepart volume number

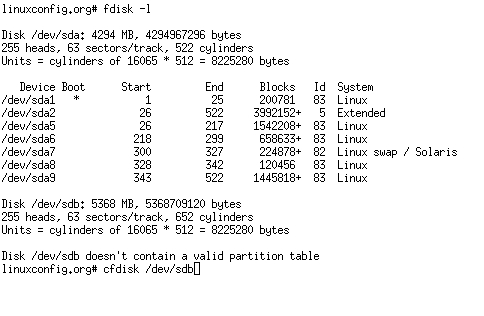
Delete 🡪 rm 1

Rescue 🡪 rescue

**LVM(Logical Volume Management)**



# Create Partitions



# Create physical volumes

Use the pvcreate command to create physical volumes.

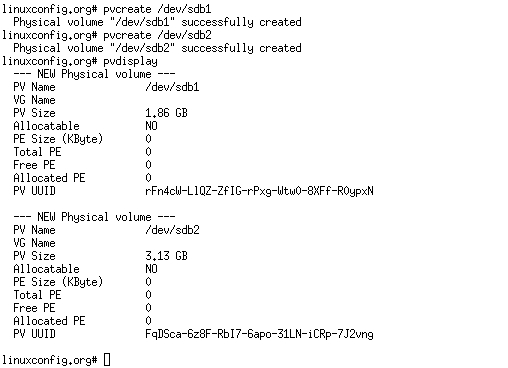
# pvcreate /dev/sdb1  
# pvcreate /dev/sdb2

The pvdisplay command displays all physical volumes on your system.

# pvdisplay

Alternatively the following command should be used:

# pvdisplay /dev/sdb1



# Create Virtual Group

his will create the volume group using 32MB PE size in the name of tecmint\_add\_vg using 3 Physical volumes we created in last steps.

# vgcreate -s 32M tecmint\_add\_vg /dev/sda1 /dev/sdb1 /dev/sdc1

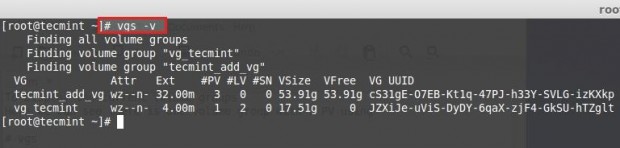
# Verify Volume Groups

nderstanding vgs command output:

1. Volume Group name.
2. Physical Volumes used in this Volume Group.
3. Shows free space available in this volume group.
4. Total Size of the Volume Group.
5. Logical Volumes inside this volume group, Here we have not yet created so there is 0.
6. SN = Number of Snapshots the volume group contains. (Later we can create a snapshot).
7. Status of the Volume group as Writeable, readable, resizeable, exported, partial and clustered, Here it is wz–n- that means w = Writable, z = resizeable..
8. Number of Physical Volume (PV) used in this Volume Group.

7. To Display more information about volume group use command.

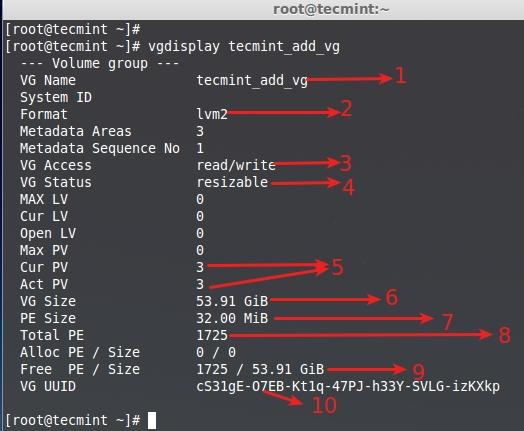
# vgs -v

[](http://www.tecmint.com/wp-content/uploads/2014/07/Create-Logical-Volume-Storage-12.jpg)

*Check Volume Group Information*

8. To get more information about newly created volume groups, run the following command.

# vgdisplay tecmint\_add\_vg

[](http://www.tecmint.com/wp-content/uploads/2014/07/Create-Logical-Volume-Storage-13.jpg)

*List New Volume Groups*

1. Volume group name
2. LVM Architecture used.
3. It can be read and write state, ready to use.
4. This volume group can be resizeable.
5. No of Physical disk used and they are active.
6. Volume Group total size.
7. A Single PE size was 32 here.
8. Total number of PE available in this volume group.
9. Currently we have not created any LV inside this VG so its totally free.
10. UUID of this volume group.

# vgcreate mynew\_vg /dev/sdb1

To include both partitions at once you can use this command:

# vgcreate mynew\_vg /dev/sdb1 /dev/sdb2

# vgcreate –s 8912 /dev/sdb1 /dev/sdb2 with pe size 8mb

# Create Virtual Group

# Extend volume Group

# virtual group extend

# Create Logical Volumes

To create a logical volume, named "vol01", with a size of 400 MB from the virtual group "mynew\_vg" use the following command:

* create a logical volume of size 400 MB -L 400
* create a logical volume of size 4 GB -L 4G

# lvcreate -L 400 -n vol01 mynew\_vg

# lvcreate

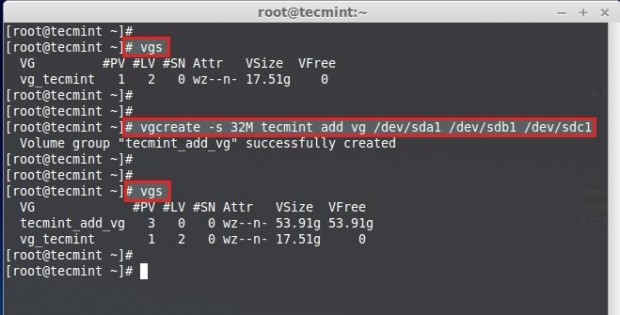
# lvcrete free space

Alternatively instead of specifying the size in megabytes, or gigabytes, you can also specify the number of physical extents to be used in that logical volume with the help of -l (lower case letter) option. Lets see a couple of similar example.

[?](http://www.slashroot.in/advanced-guide-lvm-logical-volume-management-linux-part-1)

|  |
| --- |
|  |

#lvcreate –l 200 –n vol03 mynew\_vg



# Create File system on logical volumes

mkfs.ext3 -m 0 /dev/mynew\_vg/vol01

the -m option specifies the percentage reserved for the super-user, set this to 0 if you wish not to waste any space, the default is 5%.

# Edit /etc/fstab

Add an entry for your newly created logical volume into /etc/fstab

## Mount logical volumes

Before you mount do not forget to create a mount point.

# mkdir /home/foobar

# mount logical volumes

# Extend logical volume

The biggest advantage of logical volume manager is that you can extend your logical volumes any time you are running out of the space. To increase the size of a logical volume by another 800 MB you can run this command:

# lvextend -L +800 /dev/mynew\_vg/vol01

# Extend logical volume

resize2fs /dev/mynew\_vg/vol01

# Problem with extending a logical volume

# Remove logical volume

lvremove /dev/mynew\_vg/vol02

# remove logical volume

#### Reducing Logical Volume

1. unmount the file system for reducing.
2. Check the file system after unmount.
3. Reduce the file system.
4. Reduce the Logical Volume size than Current size.
5. Recheck the file system for error.
6. Remount the file-system back to stage.

**# vgchange -a y vg00      --->>activating change.**

  1 logical volume(s) in volume group "vg00" now active

* **Linear Logical Volumes**

lvcreate -L1000 -n example examplevg

creates a linear logical volume of 1000 MB with the name of **example**, on the volume group **examplevg**. If no unit (M,K,G) is specified along with the -L option in the above command, M is assumed.

* **Stripped logical volume in LVM**

lvcreate -L 10G -i2 -I64 -n example examplevg

Creating a striped logical volume can be done by using **-i** option in lvcreate command.

The above command will create a striped logical volume with the stripe size of 64 kb, and a total size of 10G, and will use 2 physical volume for the stripes, and will be named **example**, on the volume group named **examplevg**.

* **Mirrored Logical Volume**

lvcreate -L 10G -m1 -n mirrorexample examplevg

**m1** option will create two copies of the file system, one is the linear volume and the other is the mirror of the volume. Hence if you want two mirrors then you need to specify -**m2**option with **lvcreate**command.

|  |
| --- |
| **lvconvert -m0 /dev/vg00/lv0** |

With the help of the above command, you can remove mirroring from a logical volume named **/dev/vg00/lv0.**

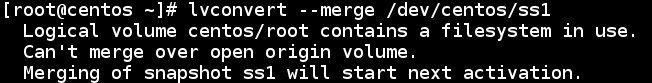
## Creating the snapshot

You can create a new snapshot using this command:

# lvcreate --size 1G --snapshot --name nameofthesnapshot /dev/mapper/nameoflvm

For example:

lvm create



lvs -o+devices

# dmsetup deps /dev/tecmint\_vg/tecmint\_lv

vdisplay -v   
lvdisplay --maps      display mirror volumes  
  
lvs -v  
lvs -a -o +devices   
  
## lvs commands for mirror volumes   
lvs -a -o +devices  
lvs -a -o +seg\_pe\_ranges --segments

## Stripe size   
lvs -v --segments  
lvs -a -o +stripes,stripesize

## use complex command  
lvs -a -o +devices,stripes,stripesize,seg\_pe\_ranges --segments

 attempt to bring the volume group online  
vgchange -a y VolData00  
  
# Restore the LVM configation  
vgcfgrestore VolData00  
  
# attempt to bring the volume grou online  
vgchange -a y VolData00  
  
# file system check  
e2fsck /dev/VolData00/data01

## dmsetup

**dmsetup** **create** *device***\_***name* *table***\_***file* *[uuid]*

**dmsetup** **remove** *device***\_***name*

**dmsetup** **rename** *device***\_***name* *new***\_***name*

**dmsetup** **suspend** *device***\_***name*

**dmsetup** **resume** *device***\_***name*

**dmsetup** **reload** *device***\_***name* *table***\_***file*

**dmsetup** **info** *device***\_***name*

**dmsetup** **deps** *device***\_***name*

**dmsetup** **status** *device***\_***name*

**dmsetup** **table** *device***\_***name*

**dmsetup** **wait** *device***\_***name*

**dmsetup** **remove\_all** *device***\_***name*

**dmsetup** **version**

# **dmsetup ls --tree**