**Partitioning and Formatting Disks**

1. **Disk Partitioning**

Data is stored on disk drives that are logically divided into partitions at the operating system level. A partition can exist on an entire disk, or it can span multiple disks. Each partition may contain a file system, a raw data space, a swap space, or a dump space.

A filesystem holds files and directories, a raw data space is used by databases and other applications, a swap space supplements physical memory on the system and a dump space stores memory image after a system crash occurs.

RHEL/CentOS offers three solutions for creating and managing disk partitions. These include solutions using **fdisk** partitioning utility, software **RAID** and **Logical Volume Manager**. All three can be used concurrently on a single disk.

During RHEL/CentOS installation, a disk partitioning program called disk druid is invoked to carve up disk based on requirements. This program allows you to use standard partitioning, software RAID partitioning or LVM partitioning technique, or a combination. The disk druid utility is available only during the installation process. You cannot run it after the installation is complete.

The **fdisk** tool is commonly used to carve up disks on RHEL/CentOS systems. This text-based, menu-driven program allows you to display, add, modify, verify and delete partitions. It allows up to three usable primary partitions and one extended partition. The disk partitioning information is stored in the partition table, which is located on the first sector of the disk. The fdisk utility is invoked on a disk device file. To determine how many disks are available on the system and basic information about the disks, run the fdisk command with –l option.

Next, we will use fdisk utility to create a LINUX ext3 partition, on which will lay an ext3 filesystem. Creating an ext3 filesystem involves a device file on which to place the filesystem and knowing the necessary commands. Here are the steps:

**A. Create the device file (the partition):**

[root@Grendel ~]# **fdisk –l -> here we view our harddrive (/dev/sda) with it’s 7 partitions**

Disk /dev/sda: 12.8 GB, 12884901888 bytes

255 heads, 63 sectors/track, 1566 cylinders

Units = cylinders of 16065 \* 512 = 8225280 bytes

Device Boot Start End Blocks Id System

/dev/sda1 \* 1 1239 9952236 83 Linux

/dev/sda2 1240 1272 265072+ 82 Linux swap / Solaris

/dev/sda3 1273 1275 24097+ 8e Linux LVM

/dev/sda4 1276 1566 2337457+ 5 Extended

/dev/sda5 1276 1288 104391 fd Linux raid autodetect

/dev/sda6 1289 1301 104391 fd Linux raid autodetect

/dev/sda7 1302 1314 104391 8e Linux LVM

[root@Grendel ~]# **fdisk /dev/sda -> here we select /dev/sda harddrive**

The number of cylinders for this disk is set to 1566.

There is nothing wrong with that, but this is larger than 1024,

and could in certain setups cause problems with:

1) software that runs at boot time (e.g., old versions of LILO)

2) booting and partitioning software from other OSs

(e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): **m**

Command action

a toggle a bootable flag

b edit bsd disklabel

c toggle the dos compatibility flag

d delete a partition

l list known partition types

m print this menu

**n add a new partition**

o create a new empty DOS partition table

**p print the partition table**

q quit without saving changes

s create a new empty Sun disklabel

t change a partition's system id

u change display/entry units

v verify the partition table

w write table to disk and exit

x extra functionality (experts only)

Command (m for help): **n -> here we add a new partition of 100M starting from cylinder 1315**

First cylinder (1315-1566, default 1315):

Using default value 1315

Last cylinder or +size or +sizeM or +sizeK (1315-1566, default 1566): +100M

Command (m for help): **p -> here we print our current partitions**

Disk /dev/sda: 12.8 GB, 12884901888 bytes

255 heads, 63 sectors/track, 1566 cylinders

Units = cylinders of 16065 \* 512 = 8225280 bytes

Device Boot Start End Blocks Id System

/dev/sda1 \* 1 1239 9952236 83 Linux

/dev/sda2 1240 1272 265072+ 82 Linux swap / Solaris

/dev/sda3 1273 1275 24097+ 8e Linux LVM

/dev/sda4 1276 1566 2337457+ 5 Extended

/dev/sda5 1276 1288 104391 fd Linux raid autodetect

/dev/sda6 1289 1301 104391 fd Linux raid autodetect

/dev/sda7 1302 1314 104391 8e Linux LVM

/dev/sda8 1315 1327 104391 83 Linux

Command (m for help): **t - > here we establish the filesystem type for partition 8**

Partition number (1-8): **8**

Hex code (type L to list codes): L

0 Empty 1e Hidden W95 FAT1 80 Old Minix bf Solaris

1 FAT12 24 NEC DOS 81 Minix / old Lin c1 DRDOS/sec (FAT-

2 XENIX root 39 Plan 9 **82**  Linux swap / So c4 DRDOS/sec (FAT-

3 XENIX usr 3c PartitionMagic **83** Linux c6 DRDOS/sec (FAT-

4 FAT16 <32M 40 Venix 80286 84 OS/2 hidden C: c7 Syrinx

5 Extended 41 PPC PReP Boot 85 Linux extended da Non-FS data

6 FAT16 42 SFS 86 NTFS volume set db CP/M / CTOS / .

7 HPFS/NTFS 4d QNX4.x 87 NTFS volume set de Dell Utility

8 AIX 4e QNX4.x 2nd part 88 Linux plaintext df BootIt

9 AIX bootable 4f QNX4.x 3rd part **8e** Linux LVM e1 DOS access

a OS/2 Boot Manag 50 OnTrack DM 93 Amoeba e3 DOS R/O

b W95 FAT32 51 OnTrack DM6 Aux 94 Amoeba BBT e4 SpeedStor

c W95 FAT32 (LBA) 52 CP/M 9f BSD/OS eb BeOS fs

e W95 FAT16 (LBA) 53 OnTrack DM6 Aux a0 IBM Thinkpad hi ee EFI GPT

f W95 Ext'd (LBA) 54 OnTrackDM6 a5 FreeBSD ef EFI (FAT-12/16/

10 OPUS 55 EZ-Drive a6 OpenBSD f0 Linux/PA-RISC b

11 Hidden FAT12 56 Golden Bow a7 NeXTSTEP f1 SpeedStor

12 Compaq diagnost 5c Priam Edisk a8 Darwin UFS f4 SpeedStor

14 Hidden FAT16 <3 61 SpeedStor a9 NetBSD f2 DOS secondary

16 Hidden FAT16 63 GNU HURD or Sys ab Darwin boot fb VMware VMFS

17 Hidden HPFS/NTF 64 Novell Netware b7 BSDI fs fc VMware VMKCORE

18 AST SmartSleep 65 Novell Netware b8 BSDI swap **fd**  Linux raid auto

1b Hidden W95 FAT3 70 DiskSecure Mult bb Boot Wizard hid fe LANstep

1c Hidden W95 FAT3 75 PC/IX be Solaris boot ff BBT

**Hex code (type L to list codes): 83**

**Notice:** if we were to create a RAID device we would have modified the partition type to **fd**, if we needed and LVM structure, we would have chosen the **8e** partition type. Swap partition type would be **82**.

Command (m for help): **p**

Disk /dev/sda: 12.8 GB, 12884901888 bytes

255 heads, 63 sectors/track, 1566 cylinders

Units = cylinders of 16065 \* 512 = 8225280 bytes

Device Boot Start End Blocks Id System

/dev/sda1 \* 1 1239 9952236 83 Linux

/dev/sda2 1240 1272 265072+ 82 Linux swap / Solaris

/dev/sda3 1273 1275 24097+ 8e Linux LVM

/dev/sda4 1276 1566 2337457+ 5 Extended

/dev/sda5 1276 1288 104391 fd Linux raid autodetect

/dev/sda6 1289 1301 104391 fd Linux raid autodetect

/dev/sda7 1302 1314 104391 8e Linux LVM

**/dev/sda8 1315 1327 104391 83 Linux -> this is our new Linux partition**

Command (m for help): **w -> here we write to disk the new partition table**

The partition table has been altered!

Calling ioctl() to re-read partition table.

The kernel still uses the old table.

The new table will be used at the next reboot.

Syncing disks.

[root@Grendel ~]#

[root@Grendel ~]# **partprobe**  **-> here we force the kernel to reread the partition table since we have made changes to it**

**B. Create the filesystem:**

[root@Grendel ~]# **mkfs -t ext3 /dev/sda8**

mke2fs 1.39 (29-May-2006)

Filesystem label=

OS type: Linux

Block size=1024 (log=0)

Fragment size=1024 (log=0)

26104 inodes, 104388 blocks

5219 blocks (5.00%) reserved for the super user

First data block=1

Maximum filesystem blocks=67371008

13 block groups

8192 blocks per group, 8192 fragments per group

2008 inodes per group

Superblock backups stored on blocks:

8193, 24577, 40961, 57345, 73729

Writing inode tables: done

Creating journal (4096 blocks): done

Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 36 mounts or

180 days, whichever comes first. Use tune2fs -c or -i to override.

**C. Mount the filesystem:**

[root@Grendel ~]# mkdir /mnt/newfilesystem

[root@Grendel ~]#

[root@Grendel ~]# **mount /dev/sda8 /mnt/newfilesystem/**

[root@Grendel ~]#

[root@Grendel ~]# df -h /dev/sda8

Filesystem Size Used Avail Use% Mounted on

/dev/sda8 99M 5.6M 89M 6% /mnt/newfilesystem

After you have created the filesystem you can check it’s parameters by using **tune2fs –l <filesystem-device-file>**, perform a filesystem check by using **e2fsck <filesystem-device-file>** or update **/etc/fstab** file so that the filesystem will be accessible at reboot.

The file /etc/fstab has the 6 columns which will be explained based on the following example:

/dev/sdb2 /media/Music ext3 defaults 1 2

* + Column 1: device file
  + Column 2: mountpoint
  + Column 3: filesystem type
  + Column 4: filesystem options
  + Column 5: enable syncing everything from ram to disk
  + Column 6: the order to check filesystem at reboot (1,2,3..7)

1. **RAID**

**RAID = Redundant Array of Independent Disks**

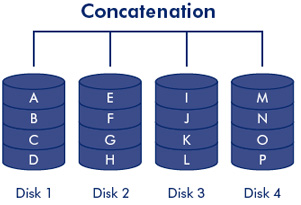
RAID is an arrangement of disk drives that is used to provide fault tolerance in case of a disk failure within the arrangement or to provide increased performance. RAID provides fault tolerance using striping, mirroring and parity. There are two types of RAID – hardware and software. Hardware RAID is provided by storage sub-systems (small to enterprise disk devices). Software RAID, in contrast, is configured at the operating system level. Almost all operating systems todat including RHEL and CentOS, support software RAID.

RAID may be implemented using two or more partitions on a single disk, but it will not provide any benefits from fault tolerance and performance perspectives. It is highly recommended to use partitions from separate disks to create RAID arrays to get true benefits offered by this arrangement.

There are several RAID levels, but RHEL/CentOS 5 supports only 0,1,5 and 6. These are explained below along with advantages associated with each one of them:

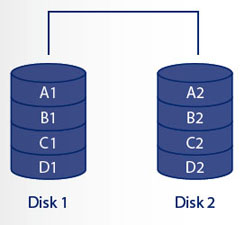
**RAID 0 – Concatenation**

This arrangement allows disks to be added one after another in such a way that a single large volume is formed. This setup requires at least 2 disks. A major disadvantage is that the entire volume stops working if one of the disks fails (no redundancy). In this configuration, data is written on the first disk and then on the second disk, but only after the first one fills up, just like in the below picture.



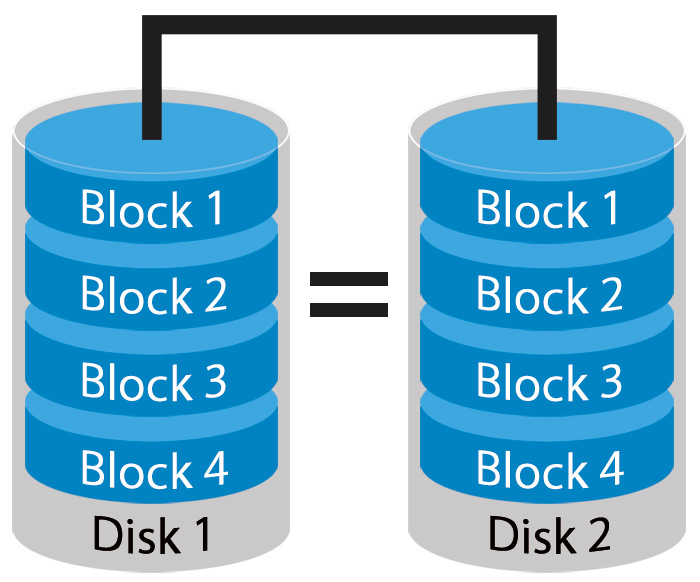
**RAID 0 – Striping**

This arrangement allows disks to be added one after another in such a way that a single large volume is formed. This setup requires at least two disks. A major disadvantage is that the entire volume stops working if one of the disks fails, but at the same time, this arrangement offers better speed. In this configuration, data is equally and alternately read and written to on all the disks in the arrangement, just like in the below figure:



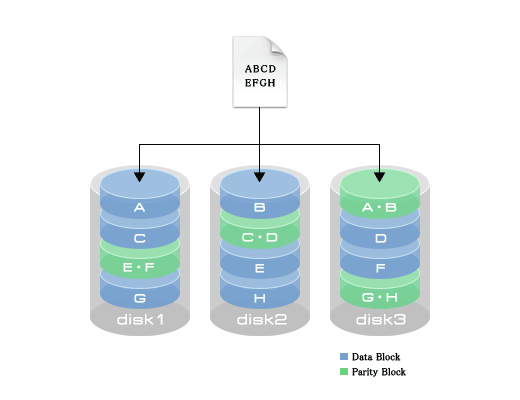
**RAID 1 – Mirroring**

RAID 1 facilitates duplication of data on two disks. If one of the disks fails, the other disk continues to function normally. The disadvantage of using RAID 1 is that you need twice as much disk capacity, but the advantage is 100% redundancy. This RAID level offers slower write performance as compared to RAID 0 since it has to write the same data on two separate disks. The read performance is better. It is highly recommended to use disks located on separate controllers for improved performance and increased redundancy. You can see it’s functionality in the following picture:



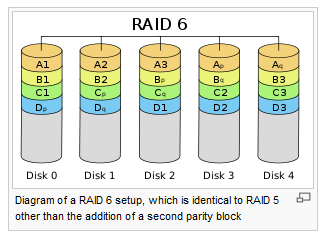
**RAID 5 – striping with parity**

RAID 5 supports both stripping as well as redundancy in terms of data parity. It adds an additional disk to RAID 0 striped volume. RAID 5 survives a single disk failure by using the parity bit to recalculate and rebuild lost data on the replaced disk. This configuration requires at least 3 disks to setup. It distributes the parity bit evenly across all disks in the configuration. RAID 5 offers good read and write performance. In this configuration, 25% of the combined disk space is used to store the parity information. In other words, you have 75% of the total disk capacity available for use. See below picture for an overview of RAID 5 mechanism:



**RAID 6**

RAID 6 is an enhanced form of RAID 5. It requires at least 4 disks to be configured and uses double parity bits. This RAID level survives a concurrent failure of two disks, but does not perform as fast as RAID 5 because I has to write two parity bits, which require some extra overhead. The mechanism is show in the below figure:



**Creating a RAID Array**

Creating a RAID array requires the needed number of hard disks (or partitions) for the RAID level that you wish to implement. Although you can create any RAID level on a single physical disk, it defeats the purpose of using RAID. Moreover, each disk in the configuration should be on a separate physical controller card.

**MDADM** is the command used to create RAID arrays. You can check the manual for this command to inspect it’s functionality. In the next section we will create a RAID 1 device and a ext3 filesystem on it which will be mounted under /mnt/Music.

**[root@Grendel ~]# rpm -qa | grep mdadm -> check if mdadm package is available**

mdadm-2.6.9-3.el5

**[root@Grendel ~]# fdisk –l -> create two fd partitions (here I have already created them)**

Disk /dev/sda: 12.8 GB, 12884901888 bytes

255 heads, 63 sectors/track, 1566 cylinders

Units = cylinders of 16065 \* 512 = 8225280 bytes

Device Boot Start End Blocks Id System

/dev/sda1 \* 1 1239 9952236 83 Linux

/dev/sda2 1240 1272 265072+ 82 Linux swap / Solaris

/dev/sda3 1273 1275 24097+ 8e Linux LVM

/dev/sda4 1276 1566 2337457+ 5 Extended

**/dev/sda5 1276 1288 104391 fd Linux raid autodetect**

**/dev/sda6 1289 1301 104391 fd Linux raid autodetect**

/dev/sda7 1302 1314 104391 8e Linux LVM

/dev/sda8 1315 1327 104391 83 Linux

Disk /dev/md1: 106 MB, 106823680 bytes

2 heads, 4 sectors/track, 26080 cylinders

Units = cylinders of 8 \* 512 = 4096 bytes

Disk /dev/md1 doesn't contain a valid partition table

[root@Grendel ~]#

[root@Grendel ~]#

**[root@Grendel ~]# df -h**

Filesystem Size Used Avail Use% Mounted on

/dev/sda1 9.2G 2.0G 6.8G 23% /

tmpfs 252M 0 252M 0% /dev/shm

**Next the raid 1 device will be created:**

**-C : create**

**-v : verbose**

**--raid-devices or –n : the number of raid devices needed in your configuration**

**--level or –l : the raid level you need to configure (possibilities: 0,1,5,6,linear,stripe)**

**--detail or –D : show detailed information about the RAID device**

**[root@Grendel ~]# mdadm -vC /dev/md0 --raid-devices=2 /dev/sda5 /dev/sda6 --level=1**

Continue creating array? y

mdadm: array /dev/md0 started.

[root@Grendel ~]#

[root@Grendel ~]#

[root@Grendel ~]#

**[root@Grendel ~]# mdadm -D /dev/md0**

/dev/md0:

Version : 0.90

Creation Time : Sun Apr 3 18:27:02 2011

**Raid Level : raid1**

Array Size : 104320 (101.89 MiB 106.82 MB)

Used Dev Size : 104320 (101.89 MiB 106.82 MB)

Raid Devices : 2

Total Devices : 2

Preferred Minor : 0

Persistence : Superblock is persistent

Update Time : Sun Apr 3 18:27:05 2011

State : clean

Active Devices : 2

Working Devices : 2

Failed Devices : 0

Spare Devices : 0

UUID : ed455d6d:3b792638:72655bfd:16e86e9c

Events : 0.2

Number Major Minor RaidDevice State

0 8 5 0 active sync /dev/sda5

1 8 6 1 active sync /dev/sda6

**[root@Grendel ~]# cat /proc/mdstat**

Personalities : [raid1]

md0 : active raid1 sda6[1] sda5[0]

104320 blocks [2/2] [UU]

unused devices: <none>

**[root@Grendel ~]# ps -ef | grep -i md**

UID PID PPID C STIME TTY TIME CMD

**root 2465 7 0 18:27 ? 00:00:00 [md0\_raid1] -> notice the process that holds and monitors my raid device md0**

root 2478 2325 0 18:30 pts/1 00:00:00 grep -i md

**[root@Grendel ~]# mkfs -t ext3 /dev/md0**

mke2fs 1.39 (29-May-2006)

Filesystem label=

OS type: Linux

Block size=1024 (log=0)

Fragment size=1024 (log=0)

26104 inodes, 104320 blocks

5216 blocks (5.00%) reserved for the super user

First data block=1

Maximum filesystem blocks=67371008

13 block groups

8192 blocks per group, 8192 fragments per group

2008 inodes per group

Superblock backups stored on blocks:

8193, 24577, 40961, 57345, 73729

Writing inode tables: done

Creating journal (4096 blocks): done

Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 26 mounts or

180 days, whichever comes first. Use tune2fs -c or -i to override.

[root@Grendel ~]#

[root@Grendel ~]#

**[root@Grendel ~]# mount -t ext3 -o defaults /dev/md0 /mnt/Music**

**[root@Grendel ~]# df -h /mnt/Music**

Filesystem Size Used Avail Use% Mounted on

/dev/md0 99M 5.6M 89M 6% /mnt/Music

**Other usefull things you can do with MDADM:**

1. Declare a disk from your array as faulty:

**mdadm <RAID DEVICE> –vf <DISK>**

1. Remove the faulty/failed disk from your array

**mdadm <RAID DEVICE> -vr <DISK>**

1. Add a new disk to the array

**mdadm <RAID DEVICE> -va <DISK>**

1. Stop a RAID array

**mdadm <RAID DEVICE> -vS <DISK>**