Linux Disk Management with parted command

Learn how to create, list, format, mount (temporary and permanent), remove and delete MBR & GPT partitions step by step in detail. This tutorial explains how to manage MBR disk partition (Primary, Extended and Logical) and GPT disk partition in Linux from parted command with practical examples.

For this tutorial I assume that you have two separate disks for exercise. If you are using virtual program (such as VMware or Virtual Box) for the practice, I suggest you to add two new virtual hard disks for this practice.

MBR partition scheme and GPT partition scheme both use different method to store partition information in disk. We cannot use both scheme in a single disk. Since in this tutorial we will create both types of partitions, we need two separate disks. We have already added two virtual disks in our system.

*To learn more about MBR partition, GPT partition and how to add virtual disks in VMware see the first part of this tutorial.*

This tutorial is the fourth part of our article “Linux Disk Management Explained in Easy Language with Examples”. You can read other parts of this article here.

[Linux Disk Management Tutorial](https://www.computernetworkingnotes.com/rhce-study-guide/linux-disk-management-tutorial.html)

*This is the first part of this article. This part explains basic concepts of Linux disk management such as BIOS, UEFI, MBR, GPT, SWAP, LVM, RAID, primary partition, extended partition and Linux file system type.*

[Manage Linux Disk Partition with fdisk Command](https://www.computernetworkingnotes.com/rhce-study-guide/manage-linux-disk-partition-with-fdisk-command.html)

*This is the second part of this article. This part explains how to create primary, extended and logical partitions from fdisk command in Linux step by step with examples.*

[Manage Linux Disk Partition with gdisk Command](https://www.computernetworkingnotes.com/rhce-study-guide/manage-linux-disk-partition-with-gdisk-command.html)

*This is the third part of this article. This part explains how to create GPT (GUID partition table) partitions from gdisk command in Linux step by step with examples.*

[How to create SWAP partition in Linux](https://www.computernetworkingnotes.com/rhce-study-guide/how-to-create-swap-partition-in-linux.html)

*This is the fifth part of this article. This part explains how to create swap partition in Linux with examples including basic swap management tasks such as how to increase, mount or clear swap memory.*

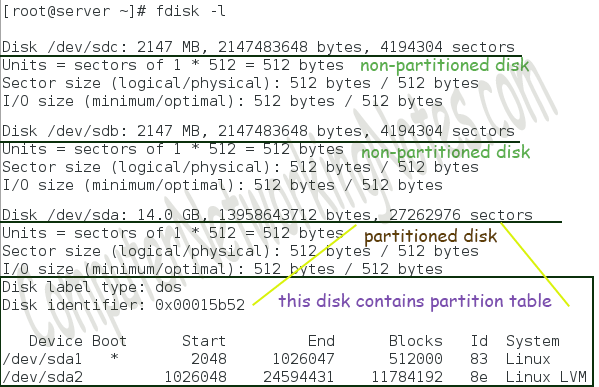
[Learn how to configure LVM in Linux step by step](https://www.computernetworkingnotes.com/rhce-study-guide/learn-how-to-configure-lvm-in-linux-step-by-step.html)

*This is the sixth part of this article. This part explains basic concepts of LVM in detail with examples including how to configure and manage LVM in Linux step by step.*

[How to configure RAID in Linux Step by Step Guide](https://www.computernetworkingnotes.com/rhce-study-guide/how-to-configure-raid-in-linux-step-by-step-guide.html)

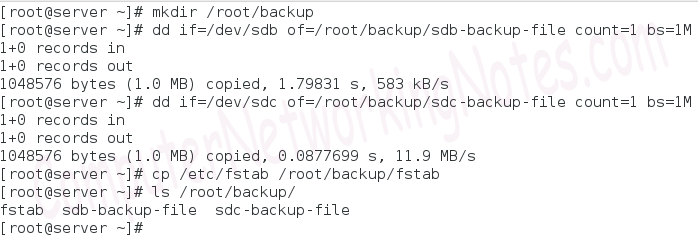
*This is the last part of this article. This part explains basic concepts of RAID in detail with examples including how to configure and manage software RAID in Linux step by step.*

Before we start let’s have an overlook of all attached hard disks with **fdisk –l** command.



As above figure shows there are two un-partitioned disks */dev/sdb* and */dev/sdc*. We will use **/dev/sdb** for the practice of GPT partitions and **/dev/sdc** for the practice of MBR partitions.

Before making any system change we should always take the backup of associate files. Backup copy allows us to revert in original state if anything goes wrong. Run following commands to create a backup copy of files which will change in this practice.



In first command we created a directory to store the backup files.

In second command we took the backup of first megabyte of raw blocks of */dev/sdb* hard disk. Let’s understand this command in detail.

**dd** :- The dd (disk duplicator) command is used to copy or backup the low level data.

**if=/dev/sdb** :- This is the first option. This option accepts source location as argument. In this example I copied from */dev/sdb* disk. *Change the disk name with the disk name which you want to use for the practice*.

**of=/root/backup/sdb-backup-file** :- This is the second option. This option accepts destination location as argument. In this example I copied at /root/backup/sdb-backup-file. The sdb-backup-file is the name of file. You can use any descriptive name here.

**bs=1** :- This is the third option. It is used to define the size of block. 1M stands for one megabyte.

**count=1** :- This is the last option. It is used to define the number of block which we want to copy.

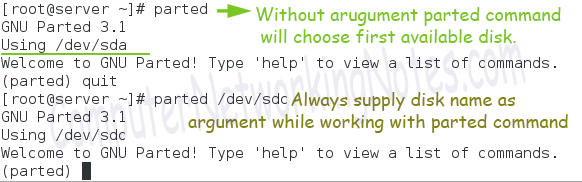
In third command we took the backup of first megabyte of raw blocks of */dev/sdc* hard disk.

In fourth command we took the backup of */etc/fstab* file. This file contains partitions information. This information is used to mount the file system while system boots.

Now we are ready to create the partitions.

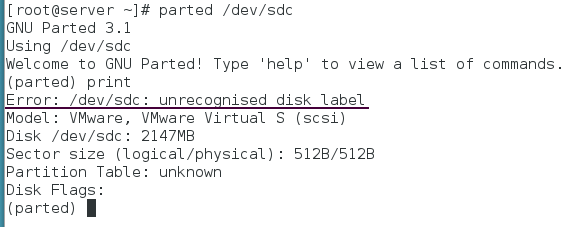
Creating MBR Partition with parted command

The **parted** command needs the name of disk where we want to create the partition as its argument.

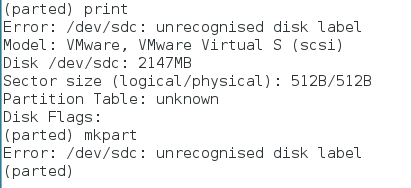


Always use parted command with argument. Without argument it will start with first available disk. The parted command includes several sub-commands which may erase entire disk data without any warning or notification. Always make sure, you are working with correct disk before executing any sub-command in parted. If wrong disk is selected, exit immediately with quit command.

Before we start working with disk let’s have the overview of current disk layout. The **print** command is used to print the current disk layout.



The **print** command provides important information about the hard disk. This information includes manufacturer information, disk model, disk size, Partition Table and disk flag. If this is a new disk, we will get ***unrecognised disk label*** error. Disk label is the information which is used to determine the type of disk. The parted command needs a valid disk label before it can do anything with disk. To see it practically run **mkpart** command. The **mkpart** command is used to create new partition.

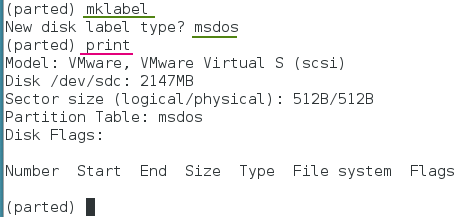


As above output confirmations that we are not allowed to create new partition unless it has a valid disk label. The *msdos* and *gpt* are two valid disk labels which we can use here. The **msdos** label is used to define the disk as MBR disk and the **gpt** label is used to define the disk as GPT disk.

*To understand the difference between MBR and GPT see the first part of this tutorial.*

To assign the label **mklabel** command is used. Let’s assign the msdos label to this disk.

* Type **mklabel** and press Enter key.
* Type **msdos** and press Enter key.
* Verify the action with **print** command.



Do not use this command if disk contains data. This command is very risky. This command will convert disk in un-partitioned blank disk silently. Neither it will generate a warning message if disk contains partitions nor it will ask for confirmation. To add worse, changes are written in disk immediately. Before hitting Enter key make sure that disk is either blank or you really want to delete all partitions from the disk.

Now disk has label. We can create partition in it. To create partition following steps are used.

Type **mkpart** and press Enter key.

As we know in MBR partition scheme we are allowed to create maximum four primary partitions. If more partitions are required, we have to use one primary partition as extended partition. Further Extended partition can be used to create 15 logical partitions.

*To learn more about MBR and GPT partition scheme, see the first part of this tutorial which explains these and other relative topics in detail with examples.*

Type the partition type and press Enter key.

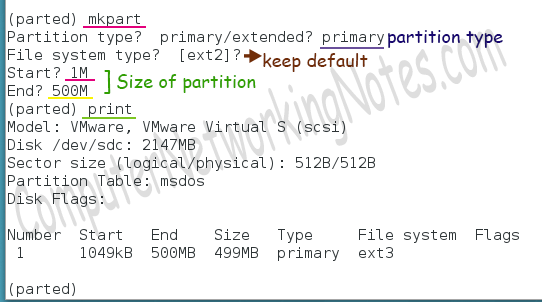
Type the starting point for partition and press Enter key.

The parted utility supports very limited file system types. To add worse, most of them are out dated nowadays. We will make file system individually later. By now just keep the default file system type.

Type the ending point for partition. We can use KB, MB or GB units to define the size of partition.

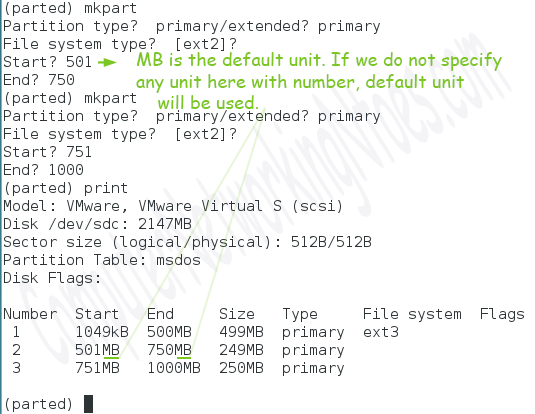
We can verify the creation of partition with **print** command

Following figure illustrates above steps



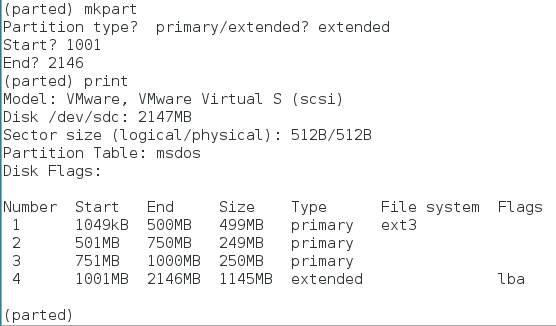
Always start first partition with 1MB even only first 512 bytes are used to store the MBR data. This way we can avoid several entry point errors.

Following same process create two more partitions



We have created three primary partitions from maximum (four) allowed. Now we can create one more primary partition or can create one extended partition.

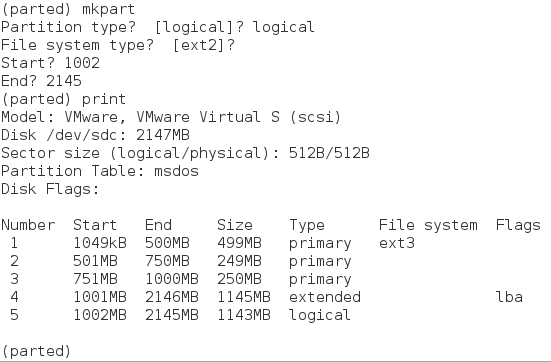
Let’s create one allowed extended partition. Since extended partition is used to create all allowed logical partitions, we should allocate the maximum disk space in this partition. We will assign all remaining disk space in this partition.



All logical partitions will be created in extended partition. The sum (of size) of all logical partitions cannot go beyond the extended partition. In this example we created an extended partition of 1145MB, so we can create logical partitions from 1145MB. We are allowed to create up to 15 logical partitions in extended partition. We can create any number of partitions from range of one to fifteen. The only limit is that the sum of all partitions must be below the size of extended partition.

*This concept is explained in detail with example in second part of this tutorial.*

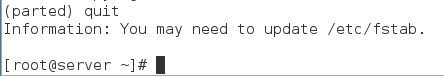
Let’s create a single logical partition of all available space.



We have used all space of extended partition in one logical partition. So even we are allowed to create up to 15 logical partitions and free space is available outside the extended partition, we cannot create the another logical partition. To create another logical partition we need free space inside the extended partition not outside the partition.

So far we have created all possible partitions including primary, extended and logical partitions.

To exit from parted use **quit** command.

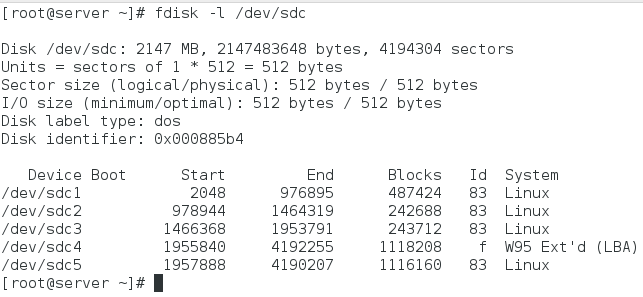


The parted cannot update the in-memory kernel partition table. Kernel reads partition table when system boots. So the kernel will be updated about this change at next reboot automatically. Meanwhile if we want to update the kernel immediately, we can use **partprobe**command which force kernel to reread the partition table.



*Replace /dev/sdc with your disk name.*

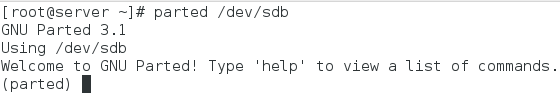
We have successfully created MBR partitions. To view the partitions use **fdisk –l** command.



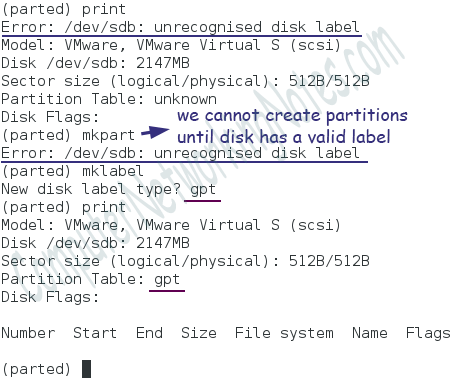
Creating GPT partition with parted command

GPT is the new partition scheme. It allows us to create up to 128 partitions. It does not divide partitions in primary, extended and logical. First part of this article explains basic concepts of GPT in detail with examples.

We will create GPT partition in /dev/sdb disk. Let’s initiate parted again for /dev/sdb disk



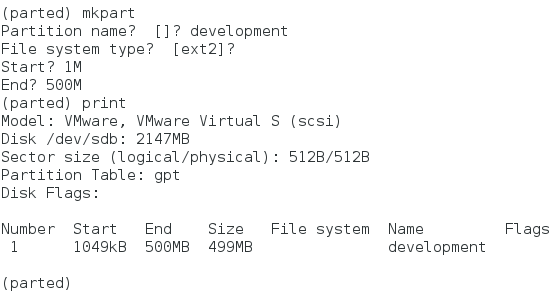
As we know, we cannot create partitions until disk has a valid label. Assign the **gpt** label in disk



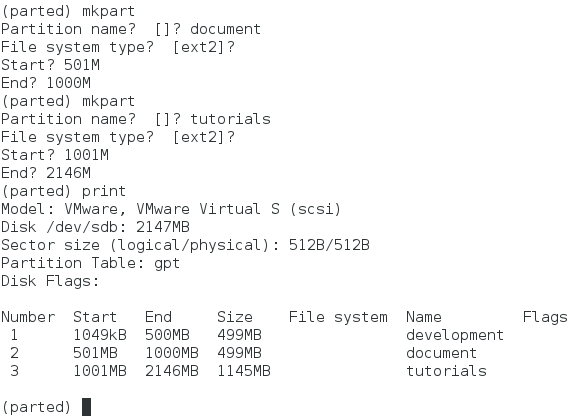
Now disk has a valid label. We can create partition in it. Creating partition is a six steps process.

* Type **mkpart** and press Enter key
* Set desired descriptive name for partition
* Accept default file system type
* Enter starting point of partition
* Enter ending point of partition
* Verify the partition with **print** command

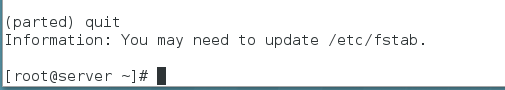
Following figure illustrates above steps



Following same steps create two more partitions



Exit from parted with quit command



Finally update kernel about this change



So far we have created MBR and GPT partitions in /dev/sdc and /dev/sdb respectively.

These partitions are useless unless we put a file system in them. File system is a logical container which allows us to store files and directories.

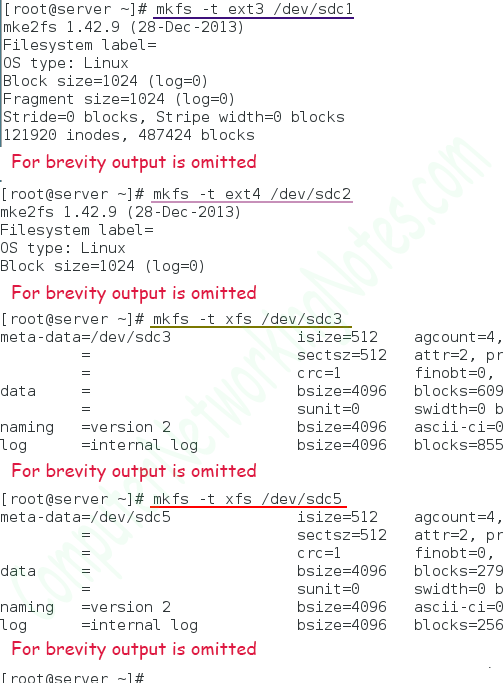
How to create file system

To create a file system in partition we format it with supported file system. To format a partition following command is used.

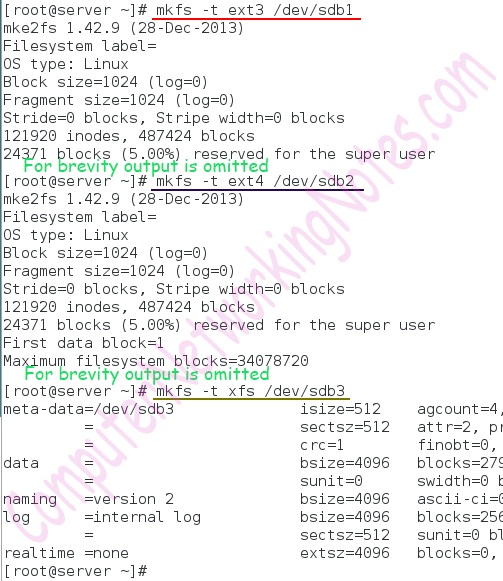
#mkfs –t [supported file system type] [partition]

Some common supported file systems are ext3, ext4 and xfs. Let’s format the recently created partitions with these file systems.

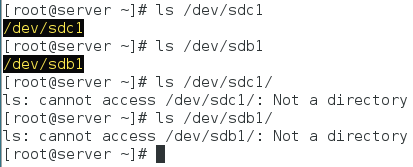
Format MBR partitions



Format GPT partitions



Now partitions have file system, can we use them for data storage? No, even though partitions are ready to use, we cannot use them until they are connected with primary directory structure (Main Linux file system).



Linux file system (primary directory structure) starts with root (/) directory and everything goes under it or its subdirectories. We have to mount partitions somewhere under this directory tree. We can mount partitions temporary or permanently.

Mounting partitions temporary

Following command is used to mount partition temporary.

#mount [what to mount] [where to mount]

*Mount command accepts several options and arguments which I will explain separately in another tutorial. For this tutorial this basic syntax is sufficient.*

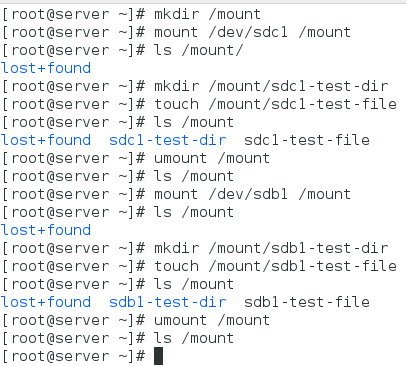
**what to mount** :- This is the partition.

**where to mount** :- This is the directory which will be used to access the mounted resource.

Once mounted, whatever action we will perform in mounted directory will be performed in mounted resources. Let’s understand it practically.

* Create a mount directory in / directory
* Mount /dev/sdc1 partition
* List the content
* Create a test directory and file
* List the content again
* Un-mount the /dev/sdc1 partition and list the content again
* Now mount the /dev/sdb1 partition and list the content
* Again create a test directory and file. Use different name for file and directory
* List the content
* Un-mount the /dev/sdb1 partition and list the content again

Following figure illustrates this exercise step by step



As above figure shows whatever action we performed in mount directory was actually performed in respective partition.

Temporary mount option is good for partitions which we access occasionally. If we access partition on regular basis then this approach will not be helpful. Each time we reboot the system all temporary mounted resources are get un-mounted automatically. So if you have a partition which is going to be used regularly, mount it permanently.

Mounting partitions permanently

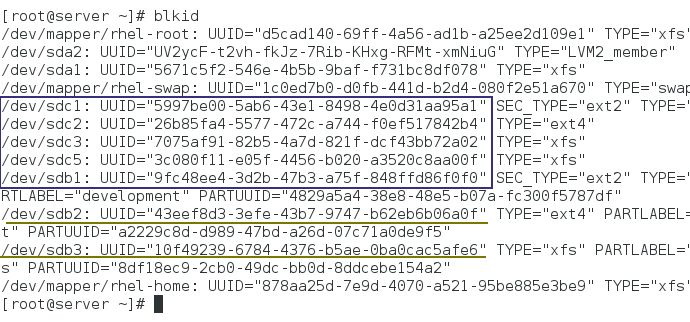
Each resource in file system has a unique ID called UUID. When mounting a partition permanently we should use UUID instead of its name. From version 7, RHEL also uses UUID instead of device name.

The UUID stands for Universally Unique Identifier. It is a 128-bit number, expressed in hexadecimal (base 16) format.

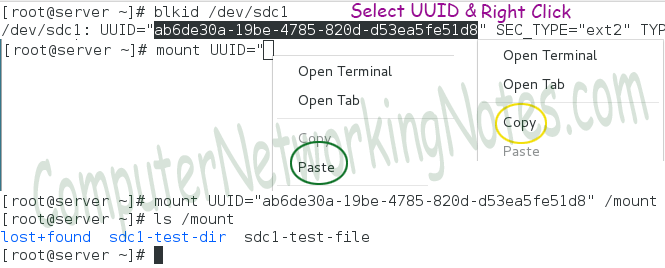
If you have a static environment, you may use device name. But if you have dynamic environment, you should always use UUID. In dynamic environment device name may change each time when system boot. For example we attached an additional SCSI disk in system; it will be named as /dev/sdb. We mounted this disk permanently with its device name. Now suppose someone else removed this disk and attached new SCSI disk in the same slot. New disk will also be named as /dev/sdb. Since name of old disk and new disk is same, new disk will be mounted at the place of old disk. This way, device name could create a serious problem in dynamic environment. But issue can solve with UUID. No matter how we attach the resource with system, its UUID will remain always fix.

*If you have static environment, you may consider device name to mount the partition. But if you have dynamic environment, you should always use UUID.*

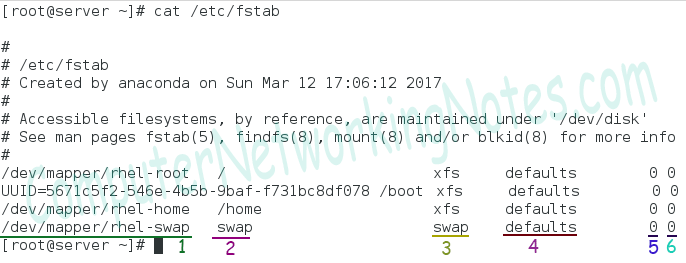
To know the UUID of all partitions we can use **blkid** command. To know the UUID of a specific partition we have to use its name as argument with this command.



Once we know the UUID, we can use it instead of device name. We can also use copy and paste option to type the UUID. Following figure illustrates the temporary mounting process using UUID.



When system boots, it looks in ***/etc/fstab*** file to find out the partitions which need to be mount automatically. By default this file has entry for those partitions which were created during the installation. To mount any additional partition automatically we have to make an entry for that partition in this file. Each entry in this file has six fields.

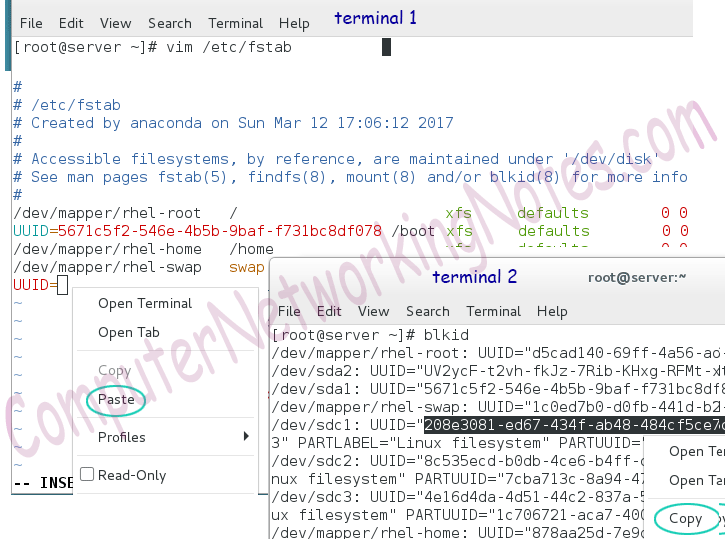


| Number | Filed | Description |
| --- | --- | --- |
| 1 | What to mount | Device which we want to mount. We can use device name, UUID and label in this filed to represent the device. |
| 2 | Where to mount | The directory in main Linux File System where we want to mount the device. |
| 3 | File system | File system type of device. |
| 4 | Options | Just like mount command we can also use supported options here to control the mount process. For this tutorial we will use default. |
| 5 | Dump support | To enable dump on this device use 1. Use 0 to disable dump. |
| 6 | Automatic check | Whether this device should be checking while mounting or not. To disable use 0, to enable use 1 (for root partition) or 2 (for all partitions except root partition). |

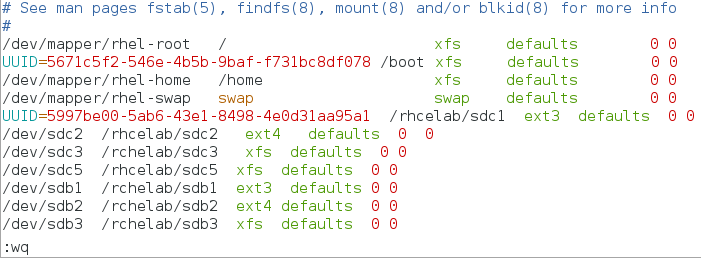
Let’s make some directories to mount the partitions which we have created recently



Now open the *fstab* file and make entries for partitions. We can also use copy and paste operation here to type the UUID. To use copy and paste operation, open another terminal and run **blkid** command. Now copy UUID of partition from second terminal and paste in **fstab** file which is open in first terminal. Once you are done close the second terminal.

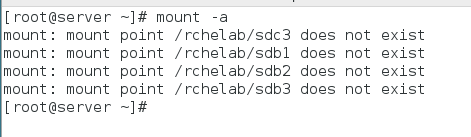


As I mentioned earlier we can also use device name or label here. Let’s use device name for remaining partitions and save the file.

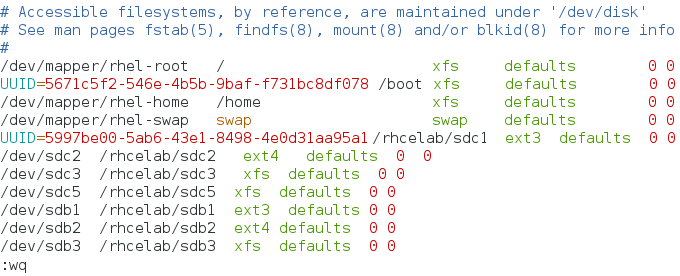


After saving always check the entries with **mount –a** command. This command will mount everything listed in */etc/fstab* file. So if we made any mistake while updating this file, we will get an error as the output of this file.

*To explain it more practically I intentionally made some typo in file.*



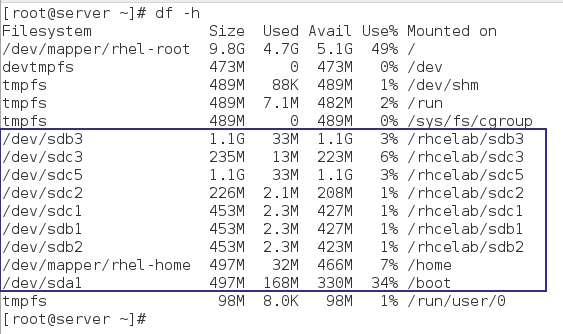
If you get any error as the output of **mount –a** command, open */etc/fstab* file again and correct the corresponding errors.



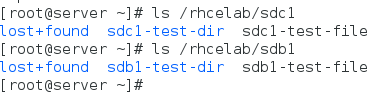
Run **mount –a** command again and if there is no error, reboot the system.

reboot command

The df –h command is used to check the available space in all mounted partition. We can use this command to verify that all partitions are mounted correctly.



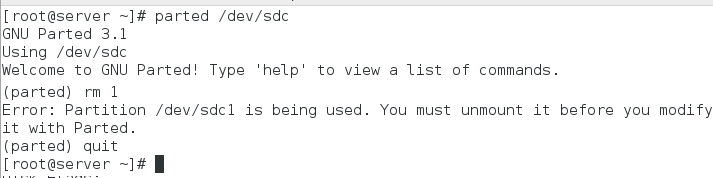
Above output confirms that we have successfully mounted all partitions. We created few files and folder in */dev/sdb1* and in */dev/sdc1*partitions while doing the practice of temporary mount. Let’s check that files and folder are still there.



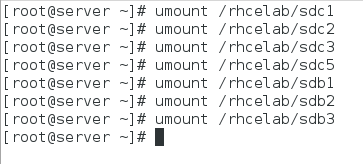
We have successfully completed this practice. Now it’s time to clean up the system for next practice.

How to delete MBR partitions

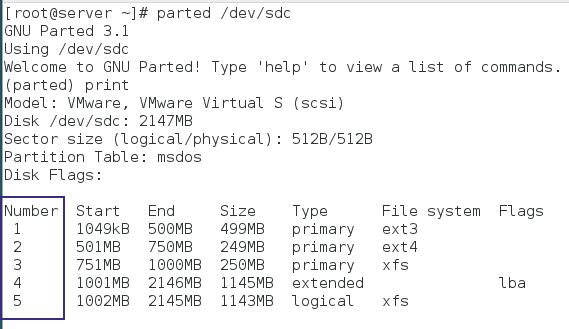
To delete a partition **rm** sub-command is used. Before we delete a partition it must be un-mounted. We cannot delete a mounted partition. Let’s see it practically.



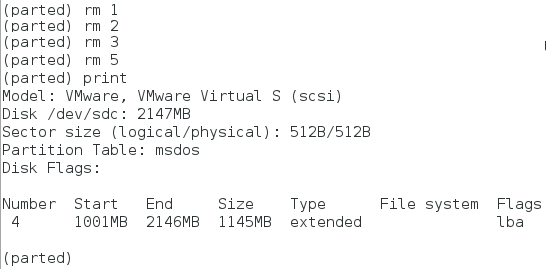
Un-mount all partitions which we created in this exercise



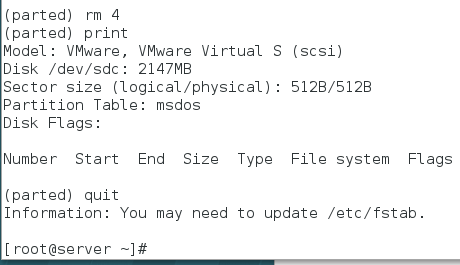
Run *parted /dev/sdc* command again



The **rm** sub-command needs partition number. We can view partition number from **print** command.

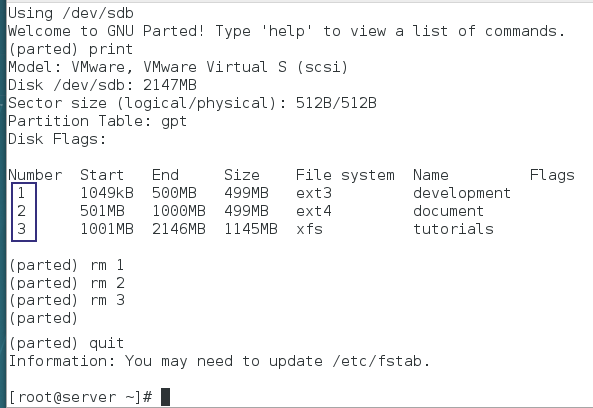


We have removed all primary and logical partitions. Now remove extended partition and exit from utility



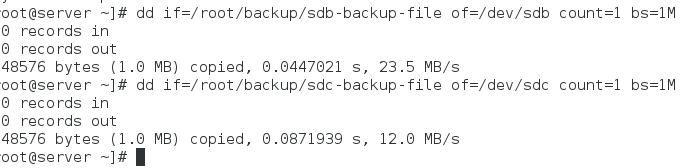
Delete GPT partitions

Deleting a GPT partition is as same as deleting a MBR partition. Initiate parted command with GPT disk (/dev/sdb) and use **print**command to view the number of partition and use **rm** sub command to delete that partition.



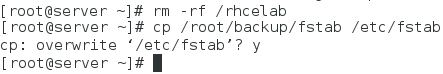
We have deleted all partitions from both disks but they still contain MBR and GPT records. If we use any disk management utility such as fdisk, gdisk or parted, the existing record will be used.

To remove current record we have to overwrite the area in disk which contains these records with raw data. For this purpose we took the backup of first 1Mb from both disks. Let’s restore that data back.

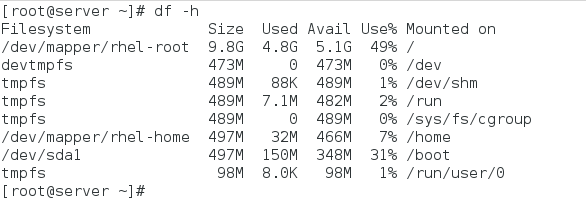


Finally remove the mount directory and copy the original fstab file back.

*If you haven’t taken the backup of original fstab file, remove all entries from this file which you made.*



Now reboot the system and use **df –h** command again to verify that all partitions which we created in this exercise are gone.



That’s all for this part. In next part we will learn how to create and manage swap partition in Linux.