

An introduction to Logical Volume Management

<http://distrowatch.com/weekly.php?issue=20090309>

For users new to Linux, the task of switching operating systems can be quite daunting. While it is quite similar to other UNIX-based operating systems, such as BSD, Solaris and OS X, it *is* very different to Windows. These days most distributions are very easy to use and (for better or worse) abstract away the complex and powerful system underneath. This makes it easier to use but as a result, users often know little about the underlying system. Today we are going to take a look at Logical Volume Management (LVM), which gives users the ability to re-size partitions on the fly.

First, a little background on hard drives and partitions. Your hard drive is where all your data is stored permanently and is not to be confused with your system's memory, which is used to store information temporarily like when you run applications or create files. Those from a Windows background will be aware of the drive letter system they use, such as C:\ and D:\ (C drive and D drive). Most users know that this is where they store their files, but not all are aware what their C:\ actually is. In order to store information on a hard drive, your computer needs to know how to read and write to it. You need to tell the computer which area(s) of the hard drive it is allowed to write to, which we call a partition. This could be the whole drive as one large partition, or multiple smaller partitions. The computer also needs to know how to store the data on each partition, which is done by creating a file system on it. Linux has a great number of file systems you can choose from, including but not limited to, ReiserFS, XFS, JFS, Btrfs, ext2, ext3 and now ext4. Windows mostly uses the NTFS and FAT32 file systems, while OS X uses HFS+. Most consumer devices, such as memory cards, are pre-formatted with FAT32.

The information for the partitions is stored on the hard drive itself, in what is called a partition table. Most personal computers use the MSDOS partition table, although Apple uses GPT which Linux fully supports. A blank hard drive has no partition table by default, but one is created when it is first partitioned. There are many tools you can use to partition your drives under Linux, including fdisk, parted and its graphical front-end, gparted. Under Linux, all devices are addressable through a file under the /dev directory. For example, your first terminal is /dev/tty0. Names of conventional hard drive are related to their position on the computer subsystem. For example, /dev/hda refers to the master drive on the primary IDE port of the mainboard. These days most

computers use SATA controllers, so the first drive is often /dev/sda. Each partition on the drive is then given a number. Your first primary partition is number one, your second is number two, and so on. The MSDOS partition table only supports a maximum of 4 partitions (which we call primary), but does support a partition type called 'extended'. Extended partitions allow users to create an unlimited number of extra partitions (called logical partitions) and under Linux this always starts at the fifth partition. Based on the partition number, you should be able to tell where the partition sits in relation to other partitions on the disk.

In terms of the file system, Linux uses a hierarchy where everything exists underneath / (slash - the root). When installing a distribution, you must create at least one partition, which will then be used as the root file system. If you are a Windows convert then you could loosely (very loosely) relate this to your C drive. Many Linux distributions will create just one such partition for all your data. Below is an example of the partition information for a hard drive. It is an 8 GB drive (/dev/sde) with a single partition (/dev/sde1) which is of the type "Linux".



Illustration 1: GParted - Single Partition

In order to access the data on this device it must be assigned a mount point. A mount point is a directory on the computer which Linux then associates the partition with. Anything that is written to this mount point is physically written to the partition it corresponds to. The /dev/sde1 device is mounted to the location /media/Linux, as seen in this screenshot.

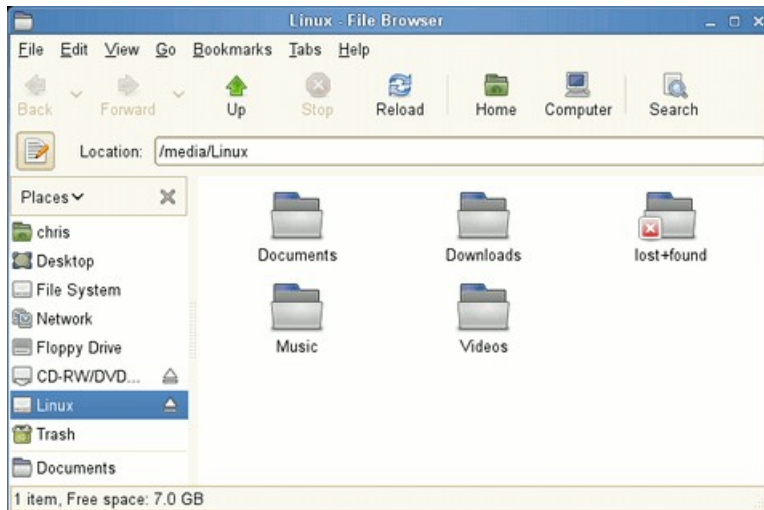


Illustration 2: Partition mounted under /media/Linux

We are all reasonably familiar with this concept when using removable media, such as memory sticks, but this can also be applied to your whole system. One of the great advantages of using a hierarchical system like this is that you can assign extra partitions to lower directories in order to preserve your data. Linux needs at least one partition for root (/) but you can also create a second partition for other locations such as /home, the location where all user data is stored. Why would you want to create a separate partition for this? Well, if you need to re-install your Linux distribution for any particular reason, you do not need to backup your data - all you need to do is wipe the first partition and mount the second as /home without formatting it. Upon completion, all your data and settings will be exactly as they were! Here is an example of a partitioned hard drive with one partition for / and another for /home.

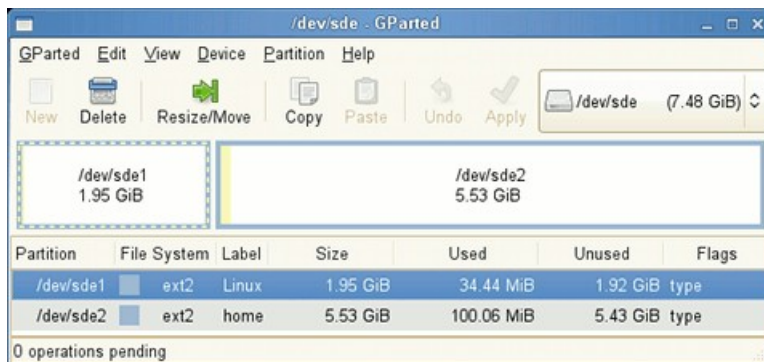


Illustration 3: GParted - dual partitions

This great feature is very handy, but how do you know how much space to assign? And what if you do this and then later run out of room? Well you could delete data or move it off to other partitions, but there is a much more powerful and flexible way. It's called Logical Volume Management. LVM is a way to dynamically create, delete, re-size and expand partitions on your computer. It's not just for servers, it's great for desktops too! How does it work? Instead of your partition information residing on your partition table, LVM writes its own information separately and keeps track of where partitions are, what devices are a part of them and how big they are. It enables you to have a partition for root, as well as another partition for home, but should you run out of space we can simply tell it to extend either partition and presto, we have more space available. Not only this, but you can add extra hard drives to your system and tell LVM to include them too! In short, you'll never run out of space again :-)

Most modern Linux distributions support LVM devices during the installer phase so most of this is managed for you automatically. In the examples above you can see a partition with type *Linux*, which is then formatted with a file system and mounted to a directory on the system. Pretty standard stuff. When using LVM, you need to set this partition type to *Linux LVM* and then use userspace tools to create and manage these devices. Importantly, GParted does not support LVM partitions, so you will need to use other tools. If you're using openSUSE, it has an excellent LVM management tool as a part of YaST. There is a kernel module you will require, as well as userspace tools. Install these via your system's package management tool. Below is an example under Debian.

```
# modprobe dm-mod ; apt-get install lvm2
```

Creating LVM devices

LVM consists of a few things. Firstly, you need a physical partition of type *Linux LVM* which will be the LVM **Physical Volume**. Next, we create a **Logical Group** to which we assign the physical partition. Next, we can create our individual partitions, called **Logical Volumes**. Let's walk through this now as the root user.

Step 1

Create a partition on a new blank drive and set it to type *LVM Linux (8e)*.

Note: My device is `/dev/sde` but yours will most likely be different!

```
# fdisk /dev/sde
n
p
1
[Enter]
[Enter]
t
8e
w
```

The device should look something like this.

```
# fdisk -l /dev/sde

Disk /dev/sde: 8040 MB, 8040480256 bytes
255 heads, 63 sectors/track, 977 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x000bf9ae

Device Boot Start End Blocks Id System
/dev/sde1 1 977 7847721 8e Linux LVM
```

Here you can see a single partition, of type *LVM Linux (8e)*.

Step 2

Now we need to tell LVM to use this partition as a Physical Volume.

```
# pvcreate /dev/sde1

No physical volume label read from /dev/sde1
```

Physical volume `"/dev/sde1"` successfully created

This device has now been added to the LVM pool.

Step 3

Create a Volume Group on the Physical Volume, which we will call 'system'.

```
# vgcreate system /dev/sde1
Volume group "system" successfully created
```

Step 4

Now that we have created a Physical Volume and a Volume Group, it's time to create a Logical Volume. We will create one for root, called *linux* and another for home.

```
# lvcreate -n linux -L 2G system
Logical volume "linux" created

# lvcreate -n home -L 3G system
Logical volume "home" created
```

That's it! Now we have two new partitions which can be formatted just like regular partitions. If we look under `/dev` we should be able to see our new devices.

```
# ls -l /dev/system/
total 0
lrwxrwxrwx 1 root root 23 Mar 9 17:18 home -> /dev/mapper/system-home
lrwxrwxrwx 1 root root 24 Mar 9 17:17 linux -> /dev/mapper/system-linux
```

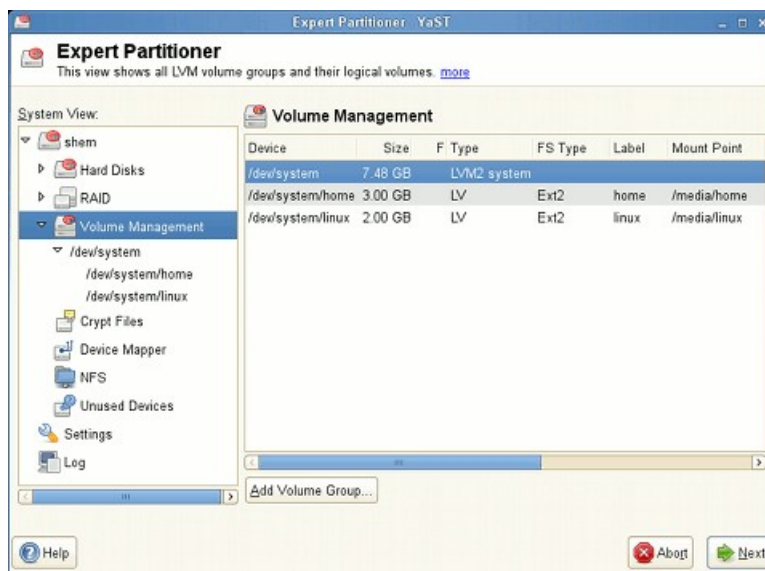


Illustration 4: openSUSE YaST partitioner - LVM

Display tools

Before we move on, let's take a look at some other LVM tools for displaying the status of our devices. Remember we have three different components that make up a complete LVM partition, Physical Volume (PV), Volume Group (VG) and Logical Volume (LV).

Let's take a look at PV.

```
# pvdisk
--- Physical volume ---
PV Name /dev/sde1
VG Name system
PV Size 7.48 GB / not usable 3.79 MB
Allocatable yes
PE Size (KByte) 4096
Total PE 1915
Free PE 1915
Allocated PE 0
PV UUID 7vkgGI-e402-K3hE-XGJz-k14C-nI7o-oFqwA8
```

Here you can see the Physical Volume name (which is the physical partition we created), the Volume Group that the partition has been assigned to (we called it *system*), as well as other information related to the size of the volume.

Let's take a look at VG.

```
# vgdisk
--- Volume group ---
VG Name system
System ID
Format lvm2
Metadata Areas 1
Metadata Sequence No 3
VG Access read/write
VG Status resizable
MAX LV 0
Cur LV 2
Open LV 0
Max PV 0
Cur PV 1
Act PV 1
VG Size 7.48 GB
PE Size 4.00 MB
Total PE 1915
Alloc PE / Size 1280 / 5.00 GB
Free PE / Size 635 / 2.48 GB
VG UUID Z6TSX0-0DQ3-7Jiz-67k2-dEKY-dYR2-RNJE85
```

Here you can see the name of the Volume Group (we called it *system*), the type it is (lvm2), how much total space there is and how much of it has already been assigned (remember we created two Logical volumes, root and home).

Finally, let's look at LV.

```
# lvdisplay
--- Logical volume ---
LV Name /dev/system/linux
VG Name system
LV UUID L0qrZu-bwCp-rnEu-uJry-4j3n-XBLB-0WsXVx
LV Write Access read/write
LV Status available
# open 0
LV Size 2.00 GB
Current LE 512
Segments 1
Allocation inherit
Read ahead sectors auto
- currently set to 256
Block device 253:0

--- Logical volume ---
LV Name /dev/system/home
VG Name system
LV UUID ASchE0-q5sJ-F8eH-bYRy-3URL-Nt7m-0UFduW
LV Write Access read/write
LV Status available
# open 0
LV Size 3.00 GB
Current LE 768
Segments 1
Allocation inherit
Read ahead sectors auto
- currently set to 256
Block device 253:1
```

Here you can see the two partitions we have created, home and linux. Notice that like PV and VG they also have a unique identifier and this is what Linux uses to detect and control the devices.

Formatting

Now that we have our two partitions, we can format them just like any other physical device.

```
# mke2fs /dev/system/linux -L linux
mke2fs 1.41.1 (01-Sep-2008)
Filesystem label=linux
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
131072 inodes, 524288 blocks
26214 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=536870912
16 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
32768, 98304, 163840, 229376, 294912

Writing inode tables: 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.
```

They can also be mounted just like any other device.

Expanding partitions

Now that you're using LVM, if you run out of space on a partition, all we need to do is tell LVM to assign more space to that particular device and re-size the file system. While you *can* shrink partitions, it is much safer to grow them. For this reason I recommend never assigning the full size of your physical volume to your Logical Volumes, but to start small and increase as you need. If you are using an ext file system, this can be performed while your partitions are still mounted (some others can too).

This is what it looks like currently, where you can see full 100% usage.

```
# df -h
/dev/mapper/system-linux 2.0G 2.0G 0 100% /media/linux
```

First, extend the Logical Volume by 1GB.

```
# lvresize -L +1G /dev/system/linux
Extending logical volume linux to 3.00 GB
Logical volume linux successfully resized
```

Now that the device has been extended, we need to re-size the file system

```
# resize2fs /dev/system/linux
Filesystem at /dev/system/linux is mounted on /media/linux; on-line resizing
required
Resizing the filesystem on /dev/system/linux to 786432 (4k) blocks.
The filesystem on /dev/system/linux is now 786432 blocks long.
```

After extending and resizing the file system, this is what it now looks like.

```
# df -h
/dev/mapper/system-linux 3.0G 2.0G 855M 71% /media/linux
```

If you want to add another hard drive to your machine, simply install it into your computer and repeat steps 1 and 2. As the Volume Group already exists, you do not need to create it as in step 3. Instead, add it to the existing group.

```
# vgextend system /dev/sdf1
Volume group "system" successfully extended
```

Now you will have the entire new disk available as additional LVM space, ready to assign to any Logical Volume you want.

And that's how you can use LVM to expand and re-assign more space on the fly, without having to shift any data or shut down your computer!

Booting from LVM

If you want to use LVM for your entire system, you can. The only issue is that you will require an additional partition for /boot and you need to use an initial RAM disk (initrd). Your boot partition only needs to be about 100MB and will store the Linux kernel and initrd. The initrd will contain the LVM userspace tools required to detect and activate your LVM devices on boot-up, in order to load the rest of the system. The distribution installers will take care of this for you automatically.

Conclusion

Well that's a behind-the-scenes look at LVM, which is a great way to gain extra control over your hard drives on a computer. It allows users to infinitely expand their individual partitions to make optimal use of all the space they have. There are lots of other features of LVM, such as snapshots and striping, but we haven't covered any of those. While the examples given here are mostly command-line driven, your distribution may have a graphical manager to make life easier. Either way, a good way to test it out is under a virtual machine where you won't damage any of your precious data.