

# Chapter 9

---

## Logical Volume Manager

### Introduction

Because this is a "how to" book, this chapter focuses on some commonly performed Logical Volume Manager (LVM):

- Logical Volume Manager (LVM) - You will probably use LVM to manage the data on your system. This section provides background information on LVM.
- Example of adding external disk capacity on an Integrity server. This thorough section shows the commands used to recognize the external storage, the commands used to configure the external storage and mount the file systems, and the scripts I use to accomplish some of the configuration automatically.
- A summary of some commonly used LVM procedures. I include only short procedures. I strongly suggest you view LVM documents under [www.hp.com/dspp](http://www.hp.com/dspp) including "When Good Disks Go Bad: Dealing With Disk Failures Under LVM."
- Some additional file system commands.

## Logical Volume Manager Background

Logical Volume Manager is a disk-management subsystem that allows you to manage physical disks as logical volumes which means that a file system can span multiple physical disks. You can view Logical Volume Manager as a flexible way of defining boundaries of disk space that are independent of one another. Not only can you specify the size of a logical volume, but you can also change its size if the need arises. This possibility is a great advancement over dedicating a disk to a files ystem or having fixed-size partitions on a disk. Logical volumes can hold file systems, raw data, or swap space. You can now specify a logical volume to be any size you wish, have logical volumes span multiple physical disks, and change the size of the logical volume if you need to!

So what do you need to know to set up Logical Volume Manager and realize all these great benefits? First, you need to know the terminology, and second, you need to know Logical Volume Manager commands. As with many other system administration tasks, you can use graphical management tools which are covered in Chapter 12. But, as usual, I recommend that you read this section first so you understand the basics of Logical Volume Manager.

For use with the Journaled File System (JFS), Hewlett-Packard has an add-on product called HP OnLineJFS. This product allows you to perform many of the LVM functions without going into single-user mode. For example, when a file system needs to be expanded, the logical volume on which it resides needs to be unmounted before the expansion takes place. Normally, that unmounting would mean shutting the system down into single-user mode so that no user or process could access the volume and it could then be unmounted. With OnLineJFS, the logical volumes and file systems are simply expanded with the system up and running and no interruption to users or processes. Even without OnLineJFS, there are some file systems that can be unmounted when not in single user mode.

With both JFS and OnLineJFS, an *intent log*, or *journal*, is used to keep track of metadata (structural information) that would be written synchronously on a traditional HFS-based system. The journal information is used to complete an operation if a crash occurs, thereby making system recovery with **fsck** much faster.

With both the base JFS and OnLineJFS products, you can specify a file system as *vxf*s rather than *hfs*. With the **newfs** or **mount** commands, for example, you can specify *-F vxf*s. There are many options that you can spec-

ify with *-o* when working with JFS. One of the most common is *-o large-files*, which allows files larger than two GB in size.

## Logical Volume Manager Terms

The following terms are used when working with Logical Volume Manager. They are only some of the terminology associated with Logical Volume Manager, but they'll get you started with Logical Volume Manager. It is a good idea to read the following brief overview of these terms:

**Volume** A device used for a file system, swap, or raw data. Without Logical Volume Manager, a volume would be either a disk partition or an entire disk drive.

**Physical Volume** A disk or LUN that can be used by LVM. An entire disk must be initialized if it is to be used by Logical Volume Manager; that is, you can't initialize only part of a disk for Logical Volume Manager use and the rest for fixed partitioning.

**Volume Group** A collection of logical volumes that are managed by Logical Volume Manager. You would typically define which disks on your system are going to be used by Logical Volume Manager and then define how you wish to group these into volume groups. Each individual disk may be a volume group, or more than one disk may form a volume group. At this point, you have created a pool of disk space called a *volume group*. A disk can belong to only one volume group. A volume group may span multiple physical disks.

- Logical Volume** This is space that is defined within a volume group. A volume group is divided up into logical volumes. This is similar to a disk partition, which is of a fixed size, but you have the flexibility to change its size. A logical volume is contained within a volume group, but the volume group may span multiple physical disks. You can have a logical volume that is bigger than a single disk.
- Physical Extent** A set of contiguous disk blocks on a physical volume. If you define a disk to be a physical volume, the contiguous blocks within that disk form a physical extent. Logical Volume Manager uses the physical extent as the unit for allocating disk space to logical volumes. If you use a small physical extent size, such as 1 MB, you have a fine granularity for defining logical volumes. If you use a large physical extent size, such as 256 MB, then you have a coarse granularity for defining logical volumes. The default size is 4 MB. The physical extent is a function of the size of the disk by default.
- Logical Extents** A logical volume is made up of logical extents. Logical extents and physical extents are the same size within a volume group. Although logical and physical extents are the same size, this doesn't mean that two logical extents map to two contiguous physical extents. It may be that you have two logical extents that end up being mapped to physical extents on different disks! With mirroring, one logical extent may have more than one physical extent.
- `/etc/lvmtab`** This file has in it the device file associated with each disk in a volume group. `/sbin/lvmrc` starts each volume group by reading the contents of this file at boot time. This file can be rebuilt with

**vgscan**. This is not an ascii file, so **strings /etc/lvmtab** is used to see its contents (non standard ASCII characters may be displayed so please be careful with this comand).

PV Links            Physical Volume Links (PV Links) provide dual SCSI or Fibre Links to the same disk. If one of the links were to fail, the other link automatically takes over routing I/O to the disk.

The following figure graphically depicts some of the logical volume terms just covered. In this figure, you can see clearly that logical extents are not mapped to contiguous physical extents, because some of the physical extents are not used.

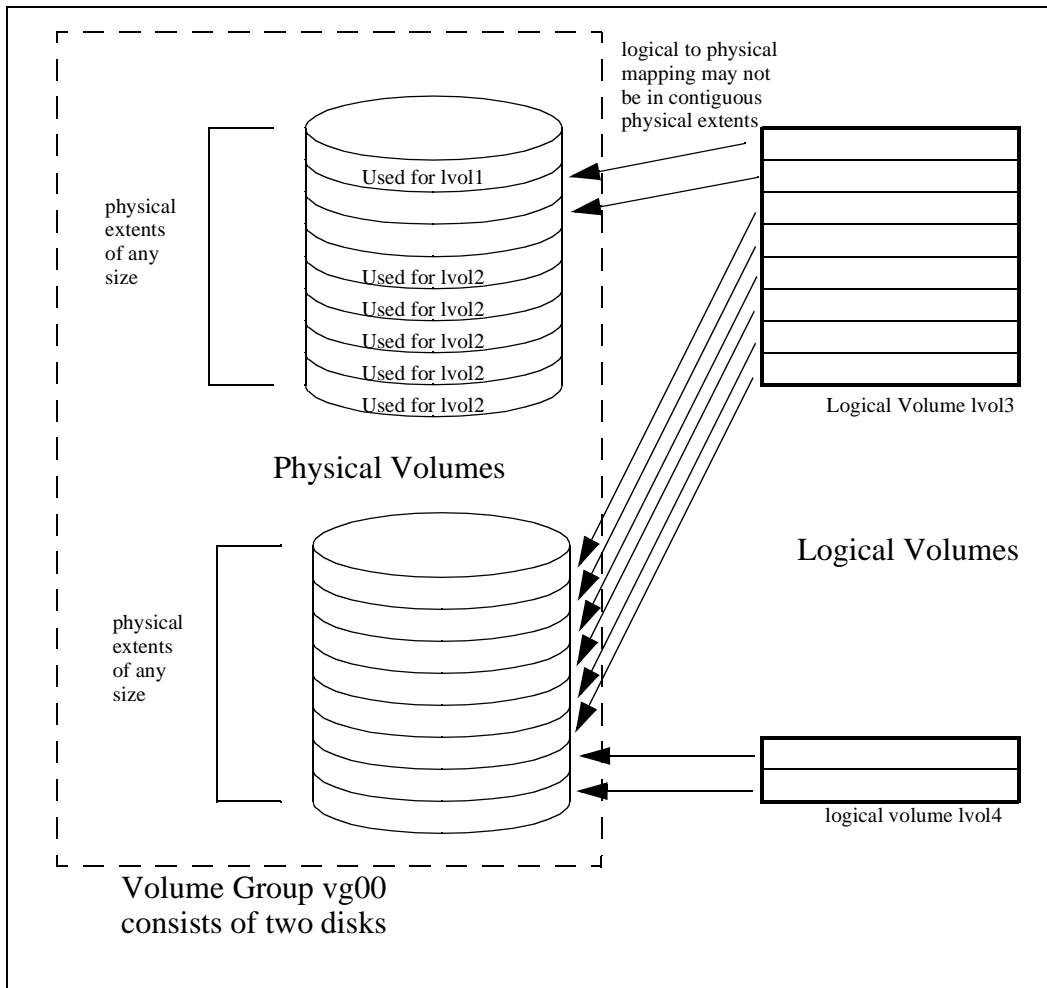


Figure 9-1 Logical Volume Manager Partial Logical to Physical Mapping

## Disk Mirroring

Logical volumes can be mirrored one or more times, which creates an identical image of the logical volume. This means that a logical extent can map to more than one physical extent if mirrored.

Mirroring is done with the *MirrorDisk/UX* product. Mirroring is done on a logical volume basis, not a disk basis. You must specify the logical volumes you want mirrored. This is demonstrated in several examples later in this chapter. Recall Figure 9-1, which showed logical extents mapped to physical extents. With mirroring, you map each logical extent to two or more physical extents, meaning that your data physically resides in two or more places.

You may have an environment where you want to mirror some or all of the logical volumes. You can configure your mirroring manually (as shown in some examples later in this chapter), or graphical management tools (covered in Chapter 12) can be used to set up disk mirroring for you. You must first, however, decide the characteristics of your mirroring. There is a mirroring policy called *strict*. You define one of the following three strict policies when you create the logical volume using the following options:

- n*                    No, this is not a strict allocation policy, meaning that mirrored copies of a logical extent can share the same physical volume. This means that your original data and mirrored data may indeed be on the same physical disk. If you encounter a disk mechanism problem of some type, you may lose both your original and mirrored data.
  
- y*                    Yes, this is a strict allocation policy, meaning that mirrored copies of a logical extent may not share the same physical volume. This is safer than allowing mirrored copies of data to share the same physical volume. If you have a problem with a disk in this scenario, you are guaranteed that your original data is on a different physical disk from your mirrored data. Original data and mirrored data are always part of the same volume group even if you want them on different physical volumes.
  
- g*                    Mirrored data will not be on the same Physical Volume Group (PVG) as the original data. This policy is called a PVG-strict allocation policy.

The strict allocation policy depends on your environment. Most installations that employ mirroring buy sufficient disk drives to mirror all data. In an environment such as this, I would create two physical volume groups, one for the original data and one for the mirrored data, and use the *strict -g* option when creating logical volumes so that the original data is on one volume group and the mirrored data on the other.

## Logical Volume Manager Commands

The following definitions are of some of the more common LVM commands. This section describes these commands so that when you see them, you have an idea of each command's use. Although these are not all of the Logical Volume Manager commands, these are the ones I use most often and are the commands you should have knowledge of when using Logical Volume Manager. The commands are grouped by physical volume (pv) commands, volume group (vg) commands, and logical volume (lv) commands. There are *usage* summaries of some of the commands included with the descriptions, and details can be found in the online manual pages. Some of the commands, such as **vgdisplay**, **pvdisplay**, and **lvdisplay** were issued so that you could see examples of these. The following output of **bdf** will be helpful to you when you view the output of Logical Volume Manager commands that are issued. The output of **bdf** shows several logical volumes mounted (**lv01**, **lv03**, **lv04**, **lv05**, **lv06**, **lv07**, and **lv08**), all of which are in volume group **vg00** (see the **bdf** command overview later in this chapter).

\$ **bdf**

File system	kbytes	used	avail	% used	Mounted on
/dev/vg00/lv03	47829	18428	24618	43%	/
/dev/vg00/lv01	67733	24736	36223	41	/stand
/dev/vg00/lv08	34541	8673	22413	28%	/var
/dev/vg00/lv07	299157	149449	119792	56%	/usr

File system	kbytes	used	avail	%used	Mounted on
/dev/vg00/lvol4	23013	48	20663	0%	/tmp
/dev/vg00/lvol6	99669	32514	57188	36%	/opt
/dev/vg00/lvol5	19861	9	17865	0%	/home
/dev/dsk/c0t6d0	802212	552120	169870	76%	/mnt/9.x

For these logical volumes to be mounted, you must manually mount them with the **mount** command or have them automatically mounted by placing entries in **/etc/fstab**.

The **/usr/sbin/mount** (1M) command has many options so consult the man page for it. The **-a** option, which I use in the command file in the upcoming section, mounts all files in **/etc/fstab**. I also show an example of **/etc/fstab** in this section. The **-r** option to **mount** mounts the file system as read only. The **-f *FStype*** allows you to specify the type of file system to mount, such as vxfs, cdfs, and others.

## Physical Volume Commands

Here is a list of the **pv** commands:

**pvchange** Changes a physical volume in some way. For example, you may want to allow additional physical extents to be added to a physical volume if they are not permitted, or prohibit additional physical extents from being added to a physical volume if, indeed, they are allowed:

```
/usr/sbin/pvchange [-A autobackup] -s pv_path
/usr/sbin/pvchange [-A autobackup] -S autoswitch pv_path
/usr/sbin/pvchange [-A autobackup] -x extensibility pv_path
/usr/sbin/pvchange [-A autobackup] -t IO_timeout pv_path
/usr/sbin/pvchange [-A autobackup] -z sparepv pv_path
```

**pvccreate** Creates a physical volume that will be part of a volume group. Remember that a volume group may consist of several physical volumes. The physical volumes are the disks on your system:

```
/usr/sbin/pvccreate [-b] [-B] [-d soft_defects] [-s disk_size]
                  [-f] [-t disk_type] pv_path
```

**pvdisplay** Shows information about the physical volumes you specify. You can get a lot of information about the logical to physical mapping with this command if you use the verbose (-v) option. With -v **pvdisplay** shows you the mapping of logical to physical extents for the physical volumes specified.

```
/usr/sbin/pvdisplay [-v] [-b BlockList] pv_path ...
```

You get a lot of other useful data from this command, such as the name of the physical volume; the name of the volume group to which the physical volume belongs; the status of the physical volume; the size of physical extents on the physical volume; the total number of physical extents; and the number of free physical extents.

**pvmove** You can move physical extents from one physical volume to other physical volumes with this command. By specifying the source physical volume and one or more destination physical volumes, you can spread data around to the various physical volumes you wish with this command. In this example, if the lvol had been left off, the entire disk would have been moved.

```
/usr/sbin/pvmove [-A autobackup] [-n lv_path] source_pv_path
                [dest_pv_path ... | dest_pvg_name ...]
```

```
# pvmove -n /dev/vg01/lvol1 /dev/dsk/c0t2d0 /dev/dsk/c0t4d0
```

**mkknod** Although this is not an LVM command exclusively, it is used often when creating volume groups, as shown in this example:

```
# cd /dev
# mkdir vg01
# cd vg01
# mkknod group c 64 0x010000
```

## Volume Group Commands

**vgcfgbackup** Saves the configuration information for a volume group. Remember that a volume group is made up of one or more physical volumes:

```
/usr/sbin/vgcfgbackup [-f vg_conf_path] [-u] vg_name
```

**vgcfgrestore** Restores the configuration information for a volume group:

```
/usr/sbin/vgcfgrestore -n vg_name -l
```

```
/usr/sbin/vgcfgrestore [-R] [-F] -n vg_name [-o old_pv_path]
pv_path
```

```
/usr/sbin/vgcfgrestore -f vg_conf_path -l
```

```
/usr/sbin/vgcfgrestore [-R] [-F] -f vg_conf_path
[-o old_pv_path] pv_path
```

**vgchange** Makes a volume group active or inactive. With the *-a* option, you can deactivate (*-a n*) a volume group or activate (*-a y*) a volume group:

```
/usr/sbin/vgchange -a availability [-l] [-p] [-q quorum] [-s]
[-P resync_daemon_count] [vg_name...]
```

**vgcreate** You can create a volume group and specify all of its parameters with this command. You specify a volume group name and all the associated parameters for the volume group when creating it. You can specify many physical volume block devices on the same command line if you want.

*ex2* below shows creating a PV link. The second path in the **vgcreate** command is the *alternate* link:

```
/usr/sbin/vgcreate [-f] [-A autobackup] [-x extensibility]
                  [-e max_pe] [-l max_lv] [-p max_pv]
                  [-s pe_size] [-g pvg_name] vg_name
                  pv_path ...
```

```
ex1: vgcreate /dev/vg01 /dev/dsk/c0t2d0
```

```
ex2: pvcreeate /dev/rdisk/c0t1d0
vgcreate /dev/vg01 /dev/dsk/c0t1d0 /dev/dsk/c2t1d0
```

**vgdisplay** Displays all information related to the volume group if you use the verbose (-v) option, including the volume group name; the status of the volume group; the maximum, current, and open logical volumes in the volume group; the maximum, current, and active physical volumes in the volume group; and physical extent-related information:

```
/usr/sbin/vgdisplay [-v] [vg_name ...]
```

**vgexport** Removes a logical volume group from the system, but does not modify the logical volume information on the physical volumes. These physical volumes can then be imported to another system using **vgimport**. Use -s for sharable option in ServiceGuard environments:

```
/usr/sbin/vgexport [-m mapfile] [-p] [-v] [-f outfile] vg_name
```

```
/usr/sbin/vgexport -m mapfile -s -p -v vg_name
```

```
# vgchange -a n /dev/vg01
# vgexport -v -m /etc/lvmconf/vg01.map /dev/vg01
```

**vgextend** Physical volumes can be added to a volume group with this command by specifying the physical volume to be added to the volume

group. After the summary of the command is an example:

```
/usr/sbin/vgextend [-f] [-A autobackup] [-g pvg_name]
                  [-x extensibility] [-z sparepv] vg_name pv_path ...
```

```
# vgextend /dev/vg01 /dev/dsk/c0t2d0
```

**vgimport** Can be used to import a physical volume to another system. Note in the example below that several disks could have been specified on the **vgimport** line.

```
/usr/sbin/vgimport [-m mapfile] [-p] [-v] [-f infile]
                  vg_name pv_path ...
```

```
/usr/sbin/vgimport -m mapfile -s -v vg_name
```

```
# mkdir /dev/vg01
# mknod /dev/vg01/group c 64 0x010000
# vgimport -v -m /dev/lvmconf/vg01.map /dev/vg01 /dev/dsk/c0t2d0
# vgchange -a y /dev/vg01
# vgcfgbackup vg01
```

**vgreduce** The size of a volume group can be reduced with this command by specifying which physical volume(s) to remove from a volume group. Make sure that the physical volume to be removed has no data on it before doing this:

```
/usr/sbin/vgreduce [-A autobackup] vg_name pv_path ...
```

```
/usr/sbin/vgreduce [-A autobackup] [-l] vg_name pv_path
```

```
/usr/sbin/vgreduce [-A autobackup] [-f] vg_name
```

**vgremove** A volume group definition can be completely removed from the system with this command. The entry for the volume group in **/dev** is not removed. An example of removing a volume group is shown below also:

```
/usr/sbin/vgremove vg_name ...
```

```
# vgchange -a n /dev/vg01
# lvremove /dev/vg01/lvol1 ;run for all lvols in vg
# vgremove /dev/vg01
```

**vgscan** In the event of a catastrophe of some type, use this command to scan your system in an effort to rebuild the `/etc/lvmtab` file. The `-p` option performs a *preview* of **vgscan**:

```
/usr/sbin/vgscan [-a] [-p] [-v]
```

**vgsync** There are times when mirrored data in a volume group becomes "stale" or out-of-date. **vgsync** synchronizes the physical extents in each mirrored logical volume in a volume group:

```
/usr/sbin/vgsync vg_name ...
```

## Logical Volume Commands

Here is a list of **lv** commands:

**lvcreate** Creates a new logical volume. A logical volume is created within a volume group. A logical volume may span multiple disks, but must exist within a volume group. Many options exist for this command, and two that you would often use are `-L` to define the size of the logical volume and `-n` to define the name of the logical volume:

```
/usr/sbin/lvcreate [-A autobackup] [-c mirror_consistency]
  [-C contiguous] [-d schedule] [-D distributed]
  [-i stripes -I stripe_size] [-l le_number | -L lv_size]
  [-m mirror_copies] [-M mirror_write_cache] [-n lv_name]
  [-p permission] [-r relocate] [-s strict] vg_name
```

An interesting nuance to **lvcreate** is that you can't specify a physical volume on which to create the logical volume. If you want a logical volume and its mirror on specific disks, you first run **lvcreate** specifying no size, then two **lvextend**