This book describes the LVM logical volume manager, including information on running LVM in a clustered environment. The content of this document is specific to the LVM2 release.
# Table of Contents

Introduction .............................................................................................................. vii  
1. About This Guide .......................................................................................... vii  
2. Audience ...................................................................................................... vii  
3. Software Versions ........................................................................................ vii  
4. Related Documentation ............................................................................... vii  
5. Document Conventions ............................................................................... viii  

1. The LVM Logical Volume Manager ............................................................... 1  
   1. Logical Volumes ......................................................................................... 1  
   2. LVM Architecture Overview .................................................................. 2  
   3. Running LVM in a Cluster ..................................................................... 3  
   4. Document Overview .............................................................................. 4  

2. LVM Components ............................................................................................ 6  
   1. Physical Volumes ...................................................................................... 6  
      1.1. LVM Physical Volume Layout ...................................................... 6  
      1.2. Multiple Partitions on a Disk ...................................................... 7  
   2. Volume Groups ......................................................................................... 8  
   3. LVM Logical Volumes ............................................................................. 8  
      3.1. Linear Volumes ............................................................................ 8  
      3.2. Striped Logical Volumes ............................................................. 11  
      3.3. Mirrored Logical Volumes ......................................................... 12  
      3.4. Snapshot Volumes ..................................................................... 13  

3. LVM Administration Overview ........................................................................ 15  
   1. Creating LVM Volumes in a Cluster ..................................................... 15  
   2. Logical Volume Creation Overview .................................................... 15  
   3. Growing a File System on a Logical Volume ....................................... 16  
   4. Logical Volume Backup ........................................................................ 16  
   5. Logging ................................................................................................... 16  

4. LVM Administration with CLI Commands ...................................................... 18  
   1. Using CLI Commands ........................................................................... 18  
   2. Physical Volume Administration .......................................................... 19  
      2.1. Creating Physical Volumes ......................................................... 19  
      2.2. Displaying Physical Volumes ...................................................... 21  
      2.3. Preventing Allocation on a Physical Volume ............................... 21  
      2.4. Resizing a Physical Volume ....................................................... 22  
      2.5. Removing Physical Volumes ....................................................... 22  
   3. Volume Group Administration ............................................................... 22  
      3.1. Creating Volume Groups ......................................................... 22  
      3.2. Adding Physical Volumes to a Volume Group ............................ 23  
      3.3. Displaying Volume Groups ......................................................... 23  
      3.4. Scanning Disks for Volume Groups to Build the Cache File .......... 24  
      3.5. Removing Physical Volumes from a Volume Group ....................... 24  
      3.6. Changing the Parameters of a Volume Group .............................. 25  
      3.7. Activating and Deactivating Volume Groups ................................. 25  
      3.8. Removing Volume Groups ......................................................... 26  
      3.9. Splitting a Volume Group ............................................................ 26
Introduction

1. About This Guide

This book describes the Logical Volume Manager (LVM), including information on running LVM in a clustered environment. The content of this document is specific to the LVM2 release.

2. Audience

This book is intended to be used by system administrators managing systems running the Linux operating system. It requires familiarity with Red Hat Enterprise Linux 5 and GFS file system administration.

3. Software Versions

<table>
<thead>
<tr>
<th>Software</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RHEL5</td>
<td>refers to RHEL5 and higher</td>
</tr>
<tr>
<td>GFS</td>
<td>refers to GFS for RHEL5 and higher</td>
</tr>
</tbody>
</table>

Table 1. Software Versions

4. Related Documentation

For more information about using Red Hat Enterprise Linux, refer to the following resources:

- *Red Hat Enterprise Linux Deployment Guide* — Provides information regarding the deployment, configuration and administration of Red Hat Enterprise Linux 5.

For more information about Red Hat Cluster Suite for Red Hat Enterprise Linux 5, refer to the following resources:

- *Configuring and Managing a Red Hat Cluster* — Provides information about installing, configuring and managing Red Hat Cluster components.
5. Document Conventions

Certain words in this manual are represented in different fonts, styles, and weights. This highlighting indicates that the word is part of a specific category. The categories include the following:

Courier font

Courier font represents commands, file names and paths, and prompts.

When shown as below, it indicates computer output:

<table>
<thead>
<tr>
<th>Desktop</th>
<th>about.html</th>
<th>logs</th>
<th>paulwesterberg.png</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mail</td>
<td>backupfiles</td>
<td>mail</td>
<td>reports</td>
</tr>
</tbody>
</table>

Bold Courier font

Bold Courier font represents text that you are to type, such as: `service jonas start`

If you have to run a command as root, the root prompt (`#`) precedes the command:

```
# gconftool-2
```

Italic Courier font

Italic Courier font represents a variable, such as an installation directory: `install_dir/bin/

Bold font

Bold font represents application programs and text found on a graphical interface.

When shown like this: OK, it indicates a button on a graphical application interface.

Additionally, the manual uses different strategies to draw your attention to pieces of information. In order of how critical the information is to you, these items are marked as follows:

**Note**

A note is typically information that you need to understand the behavior of the system.
5. Document Conventions

Tip

A tip is typically an alternative way of performing a task.

Important

Important information is necessary, but possibly unexpected, such as a configuration change that will not persist after a reboot.

Caution

A caution indicates an act that would violate your support agreement, such as re-compiling the kernel.

Warning

A warning indicates potential data loss, as may happen when tuning hardware for maximum performance.
Chapter 1. The LVM Logical Volume Manager

This chapter provides a high-level overview of the components of the Logical Volume Manager (LVM).

1. Logical Volumes

Volume management creates a layer of abstraction over physical storage, allowing you to create logical storage volumes. This provides much greater flexibility in a number of ways than using physical storage directly.

A logical volume provides storage virtualization. With a logical volume, you are not restricted to physical disk sizes. In addition, the hardware storage configuration is hidden from the software so it can be resized and moved without stopping applications or unmounting file systems. This can reduce operational costs.

Logical volumes provide the following advantages over using physical storage directly:

- Flexible capacity
  When using logical volumes, file systems can extend across multiple disks, since you can aggregate disks and partitions into a single logical volume.

- Resizeable storage pools
  You can extend logical volumes or reduce logical volumes in size with simple software commands, without reformatting and repartitioning the underlying disk devices.

- Online data relocation
  To deploy newer, faster, or more resilient storage subsystems, you can move data while your system is active. Data can be rearranged on disks while the disks are in use. For example, you can empty a hot-swappable disk before removing it.

- Convenient device naming
  Logical storage volumes can be managed in user-defined groups, which you can name according to your convenience.

- Disk striping
  You can create a logical volume that stripes data across two or more disks. This can dramatically increase throughput.

- Mirroring volumes
  Logical volumes provide a convenient way to configure a mirror for your data.

- Volume Snapshots

Using logical volumes, you can take device snapshots for consistent backups or to test the effect of changes without affecting the real data.

The implementation of these features in LVM is described in the remainder of this document.

2. LVM Architecture Overview

For the RHEL 4 release of the Linux operating system, the original LVM1 logical volume manager was replaced by LVM2, which has a more generic kernel framework than LVM1. LVM2 provides the following improvements over LVM1:

- flexible capacity
- more efficient metadata storage
- better recovery format
- new ASCII metadata format
- atomic changes to metadata
- redundant copies of metadata

LVM2 is backwards compatible with LVM1, with the exception of snapshot and cluster support. You can convert a volume group from LVM1 format to LVM2 format with the `vgconvert` command. For information on converting LVM metadata format, see the `vgconvert(8)` man page.

The underlying physical storage unit of an LVM logical volume is a block device such as a partition or whole disk. This device is initialized as an LVM physical volume (PV).

To create an LVM logical volume, the physical volumes are combined into a volume group (VG). This creates a pool of disk space out of which LVM logical volumes (LVs) can be allocated. This process is analogous to the way in which disks are divided into partitions. A logical volume is used by file systems and applications (such as databases).

Figure 1.1, “LVM Logical Volume Components” shows the components of a simple LVM logical volume:
For detailed information on the components of an LVM logical volume, see Chapter 2, *LVM Components*.

### 3. Running LVM in a Cluster

The Clustered Logical Volume Manager (CLVM) is a set of clustering extensions to LVM. These extensions allow a cluster of computers to manage shared storage (for example, on a SAN) using LVM.

The `clmvmd` daemon is the key clustering extension to LVM. The `clmvmd` daemon runs in each cluster computer and distributes LVM metadata updates in a cluster, presenting each cluster computer with the same view of the logical volumes.

Figure 1.2, “CLVM Overview” shows a CLVM overview in a Red Hat cluster.
Logical volumes created with CLVM on shared storage are visible to all computers that have access to the shared storage.

CLVM allows a user to configure logical volumes on shared storage by locking access to physical storage while a logical volume is being configured. CLVM uses the locking services provided by the high availability symmetric infrastructure.

**Note**

CLVM requires changes to the `lvm.conf` file for cluster-wide locking. For information on configuring the `lvm.conf` file to support CLVM, see Section 1, “Creating LVM Volumes in a Cluster”.

You configure LVM volumes for use in a cluster with the standard set of LVM commands or the LVM graphical user interface, as described in Chapter 4, *LVM Administration with CLI Commands* and Chapter 7, *LVM Administration with the LVM GUI*.

For information on installing LVM in a Red Hat Cluster, see *Configuring and Managing a Red Hat Cluster*.

4. Document Overview
This remainder of this document includes the following chapters:

- Chapter 2, *LVM Components* describes the components that make up an LVM logical volume.
- Chapter 3, *LVM Administration Overview* provides an overview of the basic steps you perform to configure LVM logical volumes, whether you are using the LVM Command Line Interface (CLI) commands or the LVM Graphical User Interface (GUI).
- Chapter 4, *LVM Administration with CLI Commands* summarizes the individual administrative tasks you can perform with the LVM CLI commands to create and maintain logical volumes.
- Chapter 5, *LVM Configuration Examples* provides a variety of LVM configuration examples.
- Chapter 6, *LVM Troubleshooting* provide instructions for troubleshooting a variety of LVM issues.
- Chapter 7, *LVM Administration with the LVM GUI* summarizes the operating of the LVM GUI.
- Appendix A, *The Device Mapper* describes the Device Mapper that LVM uses to map logical and physical volumes.
- Appendix B, *The LVM Configuration Files* describes the LVM configuration files.
- Appendix C, *LVM Object Tags* describes LVM object tags and host tags.
- Appendix D, *LVM Volume Group Metadata* describes LVM volume group metadata, and includes a sample copy of metadata for an LVM volume group.
Chapter 2. LVM Components

This chapter describes the components of an LVM Logical volume.

1. Physical Volumes

The underlying physical storage unit of an LVM logical volume is a block device such as a partition or whole disk. To use the device for an LVM logical volume the device must be initialized as a physical volume (PV). Initializing a block device as a physical volume places a label near the start of the device.

By default, the LVM label is placed in the second 512-byte sector. You can overwrite this default by placing the label on any of the first 4 sectors. This allows LVM volumes to co-exist with other users of these sectors, if necessary.

An LVM label provides correct identification and device ordering for a physical device, since devices can come up in any order when the system is booted. An LVM label remains persistent across reboots and throughout a cluster.

The LVM label identifies the device as an LVM physical volume. It contains a random unique identifier (the UUID) for the physical volume. It also stores the size of the block device in bytes, and it records where the LVM metadata will be stored on the device.

The LVM metadata contains the configuration details of the LVM volume groups on your system. By default, an identical copy of the metadata is maintained in every metadata area in every physical volume within the volume group. LVM metadata is small and stored as ASCII.

Currently LVM allows you to store 0, 1 or 2 identical copies of its metadata on each physical volume. The default is 1 copy. Once you configure the number of metadata copies on the physical volume, you cannot change that number at a later time. The first copy is stored at the start of the device, shortly after the label. If there is a second copy, it is placed at the end of the device. If you accidentally overwrite the area at the beginning of your disk by writing to a different disk than you intend, a second copy of the metadata at the end of the device will allow you to recover the metadata.

For detailed information about the LVM metadata and changing the metadata parameters, see Appendix D, LVM Volume Group Metadata.

1.1. LVM Physical Volume Layout

Figure 2.1, “Physical Volume layout” shows the layout of an LVM physical volume. The LVM label is on the second sector, followed by the metadata area, followed by the usable space on the device.

Note

In the Linux kernel (and throughout this document), sectors are considered to be
1.2. Multiple Partitions on a Disk

LVM allows you to create physical volumes out of disk partitions. It is generally recommended that you create a single partition that covers the whole disk to label as an LVM physical volume for the following reasons:

- Administrative convenience

  It is easier to keep track of the hardware in a system if each real disk only appears once. This becomes particularly true if a disk fails. In addition, multiple physical volumes on a single disk may cause a kernel warning about unknown partition types at boot-up.

- Striping performance

  LVM can not tell that two physical volumes are on the same physical disk. If you create a striped logical volume when two physical volumes are on the same physical disk, the stripes could be on different partitions on the same disk. This would result in a decrease in performance rather than an increase.

Although it it is not recommended, there may be specific circumstances when you will need to divide a disk into separate LVM physical volumes. For example, on a system with few disks it may be necessary to move data around partitions when you are migrating an existing system to LVM volumes. Additionally, if you have a very large disk and want to have more than one volume group for administrative purposes then it is necessary to partition the disk. If you do have a disk with more than one partition and both of those partitions are in the same volume group, take care to specify which partitions are to be included in a logical volume when creating...
2. Volume Groups

Physical volumes are combined into volume groups (VGs). This creates a pool of disk space out of which logical volumes can be allocated.

Within a volume group, the disk space available for allocation is divided into units of a fixed-size called extents. An extent is the smallest unit of space that can be allocated. Within a physical volume, extents are referred to as physical extents.

A logical volume is allocated into logical extents of the same size as the physical extents. The extent size is thus the same for all logical volumes in the volume group. The volume group maps the logical extents to physical extents.

3. LVM Logical Volumes

In LVM, a volume group is divided up into logical volumes. There are three types of LVM logical volumes: linear volumes, striped volumes, and mirrored volumes. These are described in the following sections.

3.1. Linear Volumes

A linear volume aggregates multiple physical volumes into one logical volume. For example, if you have two 60GB disks, you can create a 120GB logical volume. The physical storage is concatenated.

Creating a linear volume assigns a range of physical extents to an area of a logical volume in order. For example, as shown in Figure 2.2, "Extent Mapping" logical extents 1 to 99 could map to one physical volume and logical extents 100 to 198 could map to a second physical volume. From the point of view of the application, there is one device that is 198 extents in size.
Figure 2.2. Extent Mapping

The physical volumes that make up a logical volume do not have to be the same size. Figure 2.3, “Linear Volume with Unequal Physical Volumes” shows volume group VG1 with a physical extent size of 4MB. This volume group includes 2 physical volumes named PV1 and PV2. The physical volumes are divided into 4MB units, since that is the extent size. In this example, PV1 is 100 extents in size (400MB) and PV2 is 200 extents in size (800MB). You can create a linear volume any size between 1 and 300 extents (4MB to 1200MB). In this example, the linear volume named LV1 is 300 extents in size.
3.2. Striped Logical Volumes

You can configure more than one linear logical volume of whatever size you desire from the pool of physical extents. Figure 2.4, “Multiple Logical Volumes” shows the same volume group as in Figure 2.3, “Linear Volume with Unequal Physical Volumes”, but in this case two logical volumes have been carved out of the volume group: LV1, which is 250 extents in size (1000MB) and LV2 which is 50 extents in size (200MB).

Figure 2.3. Linear Volume with Unequal Physical Volumes

Figure 2.4. Multiple Logical Volumes
3.2. Striped Logical Volumes

When you write data to an LVM logical volume, the file system lays the data out across the underlying physical volumes. You can control the way the data is written to the physical volumes by creating a striped logical volume. For large sequential reads and writes, this can improve the efficiency of the data I/O.

Striping enhances performance by writing data to a predetermined number of physical volumes in round-round fashion. With striping, I/O can be done in parallel. In some situations, this can result in near-linear performance gain for each additional physical volume in the stripe.

The following illustration shows data being striped across three physical volumes. In this figure:

- the first stripe of data is written to PV1
- the second stripe of data is written to PV2
- the third stripe of data is written to PV3
- the fourth stripe of data is written to PV1

In a striped logical volume, the size of the stripe cannot exceed the size of an extent.

![Figure 2.5. Striping Data Across Three PVs](image)

Striped logical volumes can be extended by concatenating another set of devices onto the end of the first set. In order extend a striped logical volume, however, there must be enough free
space on the underlying physical volumes that make up the volume group to support the stripe. For example, if you have a two-way stripe that uses up an entire volume group, adding a single physical volume to the volume group will not enable you to extend the stripe. Instead, you must add at least two physical volumes to the volume group. For more information on extending a striped volume, see Section 4.9, “Extending a Striped Volume”.

3.3. Mirrored Logical Volumes

A mirror maintains identical copies of data on different devices. When data is written to one device, it is written to a second device as well, mirroring the data. This provides protection for device failures. When one leg of a mirror fails, the logical volume becomes a linear volume and can still be accessed.

LVM supports mirrored volumes. When you create a mirrored logical volume, LVM ensures that data written to an underlying physical volume is mirrored onto a separate physical volume. With LVM, you can create mirrored logical volumes with multiple mirrors.

An LVM mirror divides the device being copied into regions that are typically 512KB in size. LVM maintains a small log which it uses to keep track of which regions are in sync with the mirror or mirrors. This log can be kept on disk, which will keep it persistent across reboots, or it can be maintained in memory.

Figure 2.6, “Mirrored Logical Volume” shows a mirrored logical volume with one mirror. In this configuration, the log is maintained on disk.

Figure 2.6. Mirrored Logical Volume
Note

Mirrored logical volumes are not currently supported in a cluster.

For information on creating and modifying mirrors, see Section 4.1.3, “Creating Mirrored Volumes”.

3.4. Snapshot Volumes

The LVM snapshot feature provides the ability to create virtual images of a device at a particular instant without causing a service interruption. When a change is made to the original device (the origin) after a snapshot is taken, the snapshot feature makes a copy of the changed data area as it was prior to the change so that it can reconstruct the state of the device.

Note

LVM snapshots are not supported across the nodes in a cluster.

Because a snapshot copies only the data areas that change after the snapshot is created, the snapshot feature requires a minimal amount of storage. For example, with a rarely updated origin, 3-5% of the origin's capacity is sufficient to maintain the snapshot.

Note

Snapshot copies of a file system are virtual copies, not actual media backup for a file system. Snapshots do not provide a substitute for a backup procedure.

If a snapshot runs full, the snapshot is dropped. This is to be sure that there is enough space for the origin file system. You should regularly monitor the size of the snapshot. Snapshots are fully resizeable, however, so if you have the storage capacity you can increase the size of the snapshot volume to prevent it from getting dropped. Conversely, if you find that the snapshot volume is larger than you need, you can reduce the size of the volume to free up space that is needed by other logical volumes.

When you create a snapshot file system, full read and write access to the origin stays possible. If a chunk on a snapshot is changed, that chunk is marked and never gets copied from the original volume.

There are several uses for the snapshot feature:

- Most typically, a snapshot is taken when you need to perform a backup on a logical volume without halting the live system that is continuously updating the data.
• You can execute the `fsck` command on a snapshot file system to check the file system integrity and determine whether the original file system requires file system repair.

• Because the snapshot is read/write, you can test applications against production data by taking a snapshot and running tests against the snapshot, leaving the real data untouched.

• You can create volumes for use with the Xen virtual machine monitor. You can use the snapshot feature to create a disk image, snapshot it, and modify the snapshot for a particular domU instance. You can then create another snapshot and modify it for another domU instance. Since the only storage used is chunks that were changed on the origin or snapshot, the majority of the volume is shared.
Chapter 3. LVM Administration
Overview

This chapter provides an overview of the administrative procedures you use to configure LVM logical volumes. This chapter is intended to provide a general understanding of the steps involved. For specific step-by-step examples of common LVM configuration procedures, see Chapter 5, LVM Configuration Examples.

For descriptions of the CLI commands you can use to perform LVM administration, see Chapter 4, LVM Administration with CLI Commands. Alternately, you can use the LVM GUI, which is described in Chapter 7, LVM Administration with the LVM GUI.

1. Creating LVM Volumes in a Cluster

Creating LVM logical volumes in a cluster environment is identical to creating LVM logical volumes on a single node. There is no difference in the LVM commands themselves, or in the LVM GUI interface. In order to enable the LVM volumes you are creating in a cluster, the cluster infrastructure must be running and the cluster must be quorate.

For information on how to set up the cluster infrastructure, see Configuring and Managing a Red Hat Cluster.

2. Logical Volume Creation Overview

The following is a summary of the steps to perform to create an LVM logical volume.

1. Initialize the partitions you will use for the LVM volume as physical volumes (this labels them).

2. Create a volume group.

3. Create a logical volume.

After creating the logical volume you can create and mount the file system. The examples in this document use GFS file systems.

1. Create a GFS file system on the logical volume with the `gfs_mkfs` command.

2. Create a new mount point with the `mkdir` command. In a clustered system, create the mount point on all nodes in the cluster.

3. Mount the file system. You may want to add a line to the `fstab` file for each node in the system.

Alternately, you can create and mount the GFS file system with the LVM GUI.

Creating the LVM volume is machine independent, since the storage area for LVM setup information is on the physical volumes and not the machine where the volume was created. Servers
that use the storage have local copies, but can recreate that from what is on the physical volumes. You can attach physical volumes to a different server if the LVM versions are compatible.

3. Growing a File System on a Logical Volume

To grow a file system on a logical volume, perform the following steps:

1. Make a new physical volume.
2. Extend the volume group that contains the logical volume with the file system you are growing to include the new physical volume.
3. Extend the logical volume to include the new physical volume.
4. Grow the file system.

If you have sufficient unallocated space in the volume group, you can use that space to extend the logical volume instead of performing steps 1 and 2.

4. Logical Volume Backup

Metadata backups and archives are automatically created on every volume group and logical volume configuration change unless disabled in the `lvm.conf` file. By default, the metadata backup is stored in the `/etc/lvm/backup` file and the metadata archives are stored in the `/etc/lvm/archive` file. How long the metadata archives stored in the `/etc/lvm/archive` file are kept and how many archive files are kept is determined by parameters you can set in the `lvm.conf` file. A daily system backup should include the contents of the `/etc/lvm` directory in the backup.

Note that a metadata backup does not back up the user and system data contained in the logical volumes.

You can manually back up the metadata to the `/etc/lvm/backup` file with the `vgcfgbackup` command. You can restore metadata with the `vgcfgrestore` command. The `vgcfgbackup` and `vgcfgrestore` commands are described in Section 3.11, “Backing Up Volume Group Metadata”.

5. Logging

All message output passes through a logging module with independent choices of logging levels for:

- standard output/error
- syslog
- log file
5. Logging

- external log function

The logging levels are set in the \texttt{/etc/lvm/lvm.conf} file, which is described in Appendix B, \textit{The LVM Configuration Files}.
Chapter 4. LVM Administration with CLI Commands

This chapter summarizes the individual administrative tasks you can perform with the LVM Command Line Interface (CLI) commands to create and maintain logical volumes.

1. Using CLI Commands

There are several general features of all LVM CLI commands.

When sizes are required in a command line argument, units can always be specified explicitly. If you do not specify a unit, then a default is assumed, usually KB or MB. LVM CLI commands do not accept fractions.

When specifying units in a command line argument, LVM is case-insensitive; specifying M or m is equivalent, for example, and powers of 2 (multiples of 1024) are used. However, when specifying the --units argument in a command, lower-case indicates that units are in multiples of 1024 while upper-case indicates that units are in multiples of 1000.

Where commands take volume group or logical volume names as arguments, the full path name is optional. A logical volume called lvol0 in a volume group called vg0 can be specified as vg0/lvol0. Where a list of volume groups is required but is left empty, a list of all volume groups will be substituted. Where a list of logical volumes is required but a volume group is given, a list of all the logical volumes in that volume group will be substituted. For example, the lvdisplay vg0 command will display all the logical volumes in volume group vg0.

All LVM commands accept a -v argument, which can be entered multiple times to increase the output verbosity. For example, the following examples shows the default output of the lvcreate command.

```
# lvcreate -L 50MB new_vg
Rounding up size to full physical extent 52.00 MB
Logical volume "lvol0" created
```

The following command shows the output of the lvcreate command with the -v argument.

```
# lvcreate -v -L 50MB new_vg
Finding volume group "new_vg"
Rounding up size to full physical extent 52.00 MB
Creating logical volume lvol0
Creating volume group backup "/etc/lvm/backups/new_vg" (seqno 5).
Found volume group "new_vg"
Creating new_vg-lvol0
Loading new_vg-lvol0 table
Resuming new_vg-lvol0 (253:2)
Clearing start of logical volume "lvol0"
Creating volume group backup "/etc/lvm/backups/new_vg" (seqno 5).
Logical volume "lvol0" created
```

You could also have used the -vv, -vvv or the -vvvv argument to display increasingly more de-
tails about the command execution. The \texttt{-vvvv} argument provides the maximum amount of information at this time. The following example shows only the first few lines of output for the \texttt{lvcreate} command with the \texttt{-vvvv} argument specified.

```
# lvcreate -vvvv -L 50MB new_vg
#lvcmdline.c:913 Processing: lvcreate -vvvv -L 50MB new_vg
#lvcmdline.c:916 O_DIRECT will be used
#config/config.c:864 Setting global/locking_type to 1
#locking/locking.c:138 File-based locking selected.
#config/config.c:841 Setting global/locking_dir to /var/lock/lvm
#activate/activate.c:358 Getting target version for linear
#ioctl/libdm-iface.c:1569 dm version OF [16384]
#ioctl/libdm-iface.c:1569 dm versions OF [16384]
#activate/activate.c:358 Getting target version for striped
#ioctl/libdm-iface.c:1569 dm versions OF [16384]
#config/config.c:864 Setting activation/mirror_region_size to 512
...
```

You can display help for any of the LVM CLI commands with the \texttt{--help} argument of the command.

```
commandname --help
```

To display the man page for a command, execute the \texttt{man} command:

```
man commandname
```

The \texttt{man lvm} command provides general online information about LVM.

All LVM objects are referenced internally by a UUID, which is assigned when you create the object. This can be useful in a situation where you remove a physical volume called /dev/sdf which is part of a volume group and, when you plug it back in, you find that it is now /dev/sdk. LVM will still find the physical volume because it identifies the physical volume by its UUID and not its device name. For information on specifying the UUID of a physical volume when creating a physical volume, see see Section 4, “Recovering Physical Volume Metadata”.

## 2. Physical Volume Administration

This section describes the commands that perform the various aspects of physical volume administration.

### 2.1. Creating Physical Volumes

The following subsections describe the commands used for creating physical volumes.

#### 2.1.1. Setting the Partition Type

If you are using a whole disk device for your physical volume, the disk must have no partition table. For DOS disk partitions, the partition id should be set to 0x8e using the \texttt{fdisk} or \texttt{cfdisk} command or an equivalent. For whole disk devices only the partition table must be erased, which will effectively destroy all data on that disk. You can remove an existing partition table by zeroing the first sector with the following command:
2.2. Displaying Physical Volumes

2.1.2. Initializing Physical Volumes

Use the `pvcreate` command to initialize a block device to be used as a physical volume. Initialization is analogous to formatting a file system.

The following command initializes `/dev/sdd1`, `/dev/sde1`, and `/dev/sdf1` for use as LVM physical volumes.

```
pvcreate /dev/sdd1 /dev/sde1 /dev/sdf1
```

To initialize partitions rather than whole disks: run the `pvcreate` command on the partition. The following example initializes `/dev/hdb1` as an LVM physical volume for later use as part of an LVM logical volume.

```
pvcreate /dev/hdb1
```

2.1.3. Scanning for Block Devices

You can scan for block devices that may be used as physical volumes with the `lvmdiskscan` command, as shown in the following example.

```
# lvmdiskscan
/dev/ram0          [ 16.00 MB]
/dev/sda           [ 17.15 GB]
/dev/root          [ 13.69 GB]
/dev/ram           [ 16.00 MB]
/dev/sdal          [ 17.14 GB] LVM physical volume
/dev/VolGroup00/LogVol01 [ 512.00 MB]
/dev/ram2          [ 16.00 MB]
/dev/new_vg/lvol0   [ 52.00 MB]
/dev/ram3          [ 16.00 MB]
/dev/pkl_new_vg/sparkie_lv [ 7.14 GB]
/dev/ram4          [ 16.00 MB]
/dev/ram5          [ 16.00 MB]
/dev/ram6          [ 16.00 MB]
/dev/ram7          [ 16.00 MB]
/dev/ram8          [ 16.00 MB]
/dev/ram9          [ 16.00 MB]
/dev/ram10         [ 16.00 MB]
/dev/ram11         [ 16.00 MB]
/dev/ram12         [ 16.00 MB]
/dev/ram13         [ 16.00 MB]
/dev/ram14         [ 16.00 MB]
/dev/ram15         [ 16.00 MB]
/dev/sdb           [ 17.15 GB]
/dev/sdb1          [ 17.14 GB] LVM physical volume
/dev/sdc           [ 17.15 GB]
/dev/sdc1          [ 17.14 GB] LVM physical volume
/dev/sdd           [ 17.15 GB]
/dev/sdd1          [ 17.14 GB] LVM physical volume
```

7 disks
17 partitions
0 LVM physical volume whole disks
4 LVM physical volumes
2.2. Displaying Physical Volumes

There are three commands you can use to display properties of LVM physical volumes: `pvs`, `pvdisplay`, and `pvscan`.

The `pvs` command provides physical volume information in a configurable form, displaying one line per physical volume. The `pvs` command provides a great deal of format control, and is useful for scripting. For information on using the `pvs` command to customize your output, see Section 9, “Customized Reporting for LVM”.

The `pvdisplay` command provides a verbose multi-line output for each physical volume. It displays physical properties (size, extents, volume group, etc.) in a fixed format.

The following example shows the output of the `pvdisplay` command for a single physical volume.

```
# pvdisplay
--- Physical volume ---
PV Name /dev/sdc1
VG Name new_vg
PV Size 17.14 GB / not usable 3.40 MB
Allocatable yes
PE Size (KByte) 4096
Total PE 4388
Free PE 4375
Allocated PE 13
PV UUID Joqlch-yWSj-kuEn-IdwM-01S9-XO8M-mcpsVe
```

The `pvscan` command scans all supported LVM block devices in the system for physical volumes.

The following command shows all physical devices found:

```
# pvscan
PV /dev/sdb2 VG vg0 lvm2 [964.00 MB / 0 free]
PV /dev/sdc1 VG vg0 lvm2 [964.00 MB / 428.00 MB free]
PV /dev/sdc2 lvm2 [964.84 MB]
```

You can define a filter in the `lvm.conf` so that this command will avoid scanning specific physical volumes. For information on using filters to control which devices are scanned, see Section 6, “Controlling LVM Device Scans with Filters”.

2.3. Preventing Allocation on a Physical Volume

You can prevent allocation of physical extents on the free space of one or more physical volumes with the `pvchange` command. This may be necessary if there are disk errors, or if you will be removing the physical volume.

The following command disallows the allocation of physical extents on `/dev/sdc1`.

```
pvchange -x n /dev/sdc1
```

You can also use the `-xy` arguments of the `pvchange` command to allow allocation where it had
previously been disallowed.

### 2.4. Resizing a Physical Volume

If you need to change the size of an underlying block device for any reason, use the `pvresize` command to update LVM with the new size. You can execute this command while LVM is using the physical volume.

### 2.5. Removing Physical Volumes

If a device is no longer required for use by LVM, you can remove the LVM label with the `pvremove` command. Executing the `pvremove` command zeroes the LVM metadata on an empty physical volume.

If the physical volume you want to remove is currently part of a volume group, you must remove it from the volume group with the `vgreduce` command, as described in Section 3.5, “Removing Physical Volumes from a Volume Group”.

```bash
# pvremove /dev/ram15
  Labels on physical volume "/dev/ram15" successfully wiped
```

### 3. Volume Group Administration

This section describes the commands that perform the various aspects of volume group administration.

#### 3.1. Creating Volume Groups

To create a volume group from one or more physical volumes, use the `vgcreate` command. The `vgcreate` command creates a new volume group by name and adds at least one physical volume to it.

The following command creates a volume group named `vg1` that contains physical volumes `/dev/sdd1` and `/dev/sde1`.

`vgcreate vg1 /dev/sdd1 /dev/sde1`

When physical volumes are used to create a volume group, its disk space is divided into 4MB extents, by default. This extent is the minimum amount by which the logical volume may be increased or decreased in size. Large numbers of extents will have no impact on I/O performance of the logical volume.

You can specify the extent size with the `vgcreate` command if the default is not suitable with the `-s` argument. You can put limits on the number of physical or logical volumes the volume group can have by using the `-p` and `-l` arguments of the `vgcreate` command.

By default, a volume group allocates physical extents according to common-sense rules such as not placing parallel stripes on the same physical volume. This is the normal allocation policy. You can use the `--alloc` argument of the `vgcreate` command to specify an allocation policy of contiguous, anywhere, or cling.
The contiguous policy requires that new extents are adjacent to existing extents. If there are sufficient free extents to satisfy an allocation request but a normal allocation policy would not use them, the anywhere allocation policy will, even if that reduces performance by placing two stripes on the same physical volume. The cling policy places new extents on the same physical volume as existing extents in the same stripe of the logical volume. These policies can be changed using the vgchange command.

In general, allocation policies other than normal are required only in special cases where you need to specify unusual or nonstandard extent allocation.

LVM volume groups and underlying logical volumes are included in the device special file directory tree in the /dev directory with the following layout:

/dev/vg/lv/

For example, if you create two volume groups myvg1 and myvg2, each with three logical volumes named lv01, lv02, and lv03, this create six device special files:

/dev/myvg1/lv01
/dev/myvg1/lv02
/dev/myvg1/lv03
/dev/myvg2/lv01
/dev/myvg2/lv02
/dev/myvg2/lv03

The maximum device size with LVM is 8 Exabytes on 64-bit CPUs.

3.2. Adding Physical Volumes to a Volume Group

To add additional physical volumes to an existing volume group, use the vgextend command. The vgextend command increases a volume group’s capacity by adding one or more free physical volumes.

The following command adds the physical volume /dev/sdf1 to the volume group vg1.

vgextend vg1 /dev/sdf1

3.3. Displaying Volume Groups

There are two commands you can use to display properties of LVM volume groups: vgs and vg-display.

The vgscan command will also display the volume groups, although its primary purpose is to scan all the disks for volume groups and rebuild the LVM cache file. For information on the vg-scan command, see Section 3.4, “Scanning Disks for Volume Groups to Build the Cache File”.

The vgs Command provides volume group information in a configurable form, displaying one line per volume group. The vgs command provides a great deal of format control, and is useful for scripting. For information on using the vgs command to customize your output, see Section 9, “Customized Reporting for LVM”.

23
The `vgdisplay` command displays volume group properties (such as size, extents, number of physical volumes, etc.) in a fixed form. The following example shows the output of a `vgdisplay` command for the volume group `new_vg`. If you do not specify a volume group, all existing volume groups are displayed.

```bash
# vgdisplay new_vg
--- Volume group ---
VG Name new_vg
System ID
Format lvml
Metadata Areas 3
Metadata Sequence No 11
VG Access read/write
VG Status resizable
MAX LV 0
Cur LV 1
Open LV 0
Max PV 0
Cur PV 3
Act PV 3
VG Size 51.42 GB
PE Size 4.00 MB
Total PE 13164
Alloc PE / Size 13 / 52.00 MB
Free PE / Size 13151 / 51.37 GB
VG UUID jxQJ0a-ZKk0-OpMO-0118-nlwO-wwqd-fD5D32
```

### 3.4. Scanning Disks for Volume Groups to Build the Cache File

The `vgscan` command scans all supported disk devices in the system looking for LVM physical volumes and volume groups. This builds the LVM cache in the `/etc/lvm/.cache` file, which maintains a listing of current LVM devices.

LVM runs the `vgscan` command automatically at system startup and at other times during LVM operation, such as when you execute a `vgcreate` command or when LVM detects an inconsistency. You may need to run the `vgscan` command manually when you change your hardware configuration, causing new devices to be visible to the system that were not present at system bootup. This may be necessary, for example, when you add new disks to the system on a SAN or hotplug a new disk that has been labeled as a physical volume.

You can define a filter in the `lvm.conf` file to restrict the scan to avoid specific devices. For information on using filters to control which devices are scanned, see Section 6, "Controlling LVM Device Scans with Filters".

The following example shows the output of a `vgscan` command.

```bash
# vgscan
Reading all physical volumes. This may take a while...
Found volume group "new_vg" using metadata type lvml
Found volume group "officevg" using metadata type lvml
```

### 3.5. Removing Physical Volumes from a Volume Group

To remove unused physical volumes from a volume group, use the `vgreduce` command. The
vgreduce command shrinks a volume group’s capacity by removing one or more empty physical volumes. This frees those physical volumes to be used in different volume groups or to be removed from the system.

Before removing a physical volume from a volume group, you can make sure that the physical volume is not used by any logical volumes by using the pvdisplay command.

```
# pvdisplay /dev/hda1
-- Physical volume ---
PV Name /dev/hda1
VG Name myvg
PV Size 1.95 GB / NOT usable 4 MB [LVM: 122 KB]
PV# 1
PV Status available
Allocatable yes (but full)
Cur LV 1
PE Size (KByte) 4096
Total PE 499
Free PE 0
Allocated PE 499
PV UUID Sd44tK-9IRw-SrMC-MOkn-76iP-iftz-OVSen7
```

If the physical volume is still being used you will have to migrate the data to another physical volume using the pvmove command. Then use the vgreduce command to remove the physical volume:

```
The following command removes the physical volume /dev/hda1 from the volume group my_volume_group.
```

```
# vgreduce my_volume_group /dev/hda1
```

3.6. Changing the Parameters of a Volume Group

There are several volume group parameters that you can change for an existing volume group with the vgchange command. Primarily, however, this command is used to deactivate and activate volume groups, as described in Section 3.7, “Activating and Deactivating Volume Groups”,

```
The following command changes the maximum number of logical volumes of volume group vg00 to 128.
```

```
vgchange -l 128 /dev/vg00
```

For a description of the volume group parameters you can change with the vgchange command, see the vgchange(8) man page.

3.7. Activating and Deactivating Volume Groups

When you create a volume group it is, by default, activated. This means that the logical volumes in that group are accessible and subject to change.

There are various circumstances for which you need to make a volume group inactive and thus unknown to the kernel. To deactivate or activate a volume group, use the -a (--available) argument of the vgchange command.
The following example deactivates the volume group `my_volume_group`.

```
vgchange -a n my_volume_group
```

If clustered locking is enabled, add ‘e’ to activate or deactivate a volume group exclusively on one node or ‘l’ to activate or deactivate a volume group only on the local node. Logical volumes with single-host snapshots are always activated exclusively because they can only be used on one node at once.

You can deactivate individual logical volumes with the `lvchange` command, as described in Section 4.4, “Changing the Parameters of a Logical Volume Group”. For information on activating logical volumes on individual nodes in a cluster, see Section 8, “Activating Logical Volumes on Individual Nodes in a Cluster”.

### 3.8. Removing Volume Groups

To remove a volume group that contains no logical volumes, use the `vgremove` command.

```
# vgremove officevg
Volume group "officevg" successfully removed
```

### 3.9. Splitting a Volume Group

To split the physical volumes of a volume group and create a new volume group, use the `vgsplit` command.

Logical volumes cannot be split between volume groups. Each existing logical volume must be entirely on the physical volumes forming either the old or the new volume group. If necessary, however, you can use the `pvmove` command to force the split.

The following example splits off the new volume group `smallvg` from the original volume group `bigvg`.

```
# vgsplit bigvg smallvg /dev/ram15
Volume group "smallvg" successfully split from "bigvg"
```

### 3.10. Combining Volume Groups

Two combine two volume groups into a single volume group, use the `vgmerge` command. You can merge an inactive "source" volume with an active or an inactive "destination" volume if the physical extent sizes of the volume are equal and the physical and logical volume summaries of both volume groups fit into the destination volume groups limits.

The following command merges the inactive volume group `my_vg` into the active or inactive volume group `databases` giving verbose runtime information.

```
vgmerge -v databases my_vg
```

### 3.11. Backing Up Volume Group Metadata
Metadata backups and archives are automatically created on every volume group and logical volume configuration change unless disabled in the lvm.conf file. By default, the metadata backup is stored in the /etc/lvm/backup file and the metadata archives are stored in the /etc/lvm/archives file. You can manually back up the metadata to the /etc/lvm/backup file with the vgcfgbackup command.

The vgcfrestore command restores the metadata of a volume group from the archive to all the physical volumes in the volume groups.

For an example of using the vgcfrestore command to recover physical volume metadata, see Section 4, “Recovering Physical Volume Metadata”.

### 3.12. Renaming a Volume Group

Use the vgrename command to rename an existing volume group.

Either of the following commands renames the existing volume group vg02 to my_volume_group

```
vgrename /dev/vg02 /dev/my_volume_group

vgrename vg02 my_volume_group
```

### 3.13. Moving a Volume Group to Another System

You can move an entire LVM volume group to another system. It is recommended that you use the vgexport and vgimport commands when you do this.

The vgexport command makes an inactive volume group inaccessible to the system, which allows you to detach its physical volumes. The vgimport command makes a volume group accessible to a machine again after the vgexport command has made it inactive.

To move a volume group from one system to another, perform the following steps:

1. Make sure that no users are accessing files on the active volumes in the volume group, then unmount the logical volumes.
2. Use the -a n argument of the vgchange command to mark the volume group as inactive, which prevents any further activity on the volume group.
3. Use the vgexport command to export the volume group. This prevents it from being accessed by the system from which you are removing it.

After you export the volume group, the physical volume will show up as being in an exported volume group when you execute the pvscan command, as in the following example.

```
[root@tng3-1]# pvscan
PV /dev/sda1 is in exported VG myvg [17.15 GB / 7.15 GB free]
PV /dev/sdc1 is in exported VG myvg [17.15 GB / 15.15 GB free]
PV /dev/sdd1 is in exported VG myvg [17.15 GB / 15.15 GB free]
...```
When the system is next shut down, you can unplug the disks that constitute the volume group and connect them to the new system.

4. When the disks are plugged into the new system, use the `vgimport` command to import the volume group, making it accessible to the new system.

5. Activate the volume group with the `-a y` argument of the `vgchange` command.

6. Mount the file system to make it available for use.


To recreate a volume group directory and logical volume special files, use the `vgmknodes` command. This command checks the LVM2 special files in the `/dev` directory that are needed for active logical volumes. It creates any special files that are missing removes unused ones.

You can incorporate the `vgmknodes` command into the `vgscan` command by specifying the `-mknodes` argument to the command.

### 4. Logical Volume Administration

This section describes the commands that perform the various aspects of logical volume administration.

#### 4.1. Creating Logical Volumes

To create a logical volume, use the `lvcreate` command. You can create linear volumes, striped volumes, and mirrored volumes, as described in the following subsections.

If you do not specify a name for the logical volume, the default name `lvol#` is used where `#` is the internal number of the logical volume.

The following sections provide examples of logical volume creation for the three types of logical volumes you can create with LVM.

#### 4.1.1. Creating Linear Volumes

When you create a logical volume, the logical volume is carved from a volume group using the free extents on the physical volumes that make up the volume group. Normally logical volumes use up any space available on the underlying physical volumes on a next-free basis. Modifying the logical volume frees and reallocates space in the physical volumes.

The following command creates a logical volume 10 gigabytes in size in the volume group `vg1`.

```
 lvcreate -L 10G vg1
```

The following command creates a 1500 megabyte linear logical volume named `testlv` in the volume group `testvg`, creating the block device `/dev/testvg/testlv`.

```
 lvcreate -L1500 -ntestlv testvg
```
The following command creates a 50 gigabyte logical volume named `gfs1v` from the free extents in volume group `vg0`.

```
lvcreate -L 50G -n gfs1v vg0
```

You can use the `-l` argument of the `lvcreate` command to specify the size of the logical volume in extents. You can also use this argument to specify the percentage of the volume group to use for the logical volume. The following command creates a logical volume called `mylv` that uses 60% of the total space in volume group `testvol`.

```
lvcreate -l 60%VG -n mylv testvol
```

You can also use the `-l` argument of the `lvcreate` command to specify the percentage of the remaining free space in a volume group as the size of the logical volume. The following command creates a logical volume called `yourlv` that uses all of the unallocated space in the volume group `testvol`.

```
lvcreate -l 100%FREE -n yourlv testvol
```

You can use the `-l` argument of the `lvcreate` command to create a logical volume that uses the entire volume group. Another way to create a logical volume that uses the entire volume group is to use the `vgdisplay` command to find the "Total PE" size and to use those results as input to the `lvcreate` command.

The following commands create a logical volume called `mylv` that fills the volume group named `testvg`.

```
# vgdisplay testvg | grep "Total PE"
Total PE 10230
# lvcreate -l 10230 testvg -n mylv
```

The underlying physical volumes used to create a logical volume can be important if the physical volume needs to be removed, so you may need to consider this possibility when you create the logical volume. For information on removing a physical volume from a volume group, see Section 3.5, “Removing Physical Volumes from a Volume Group”.

To create a logical volume to be allocated from a specific physical volume in the volume group, specify the physical volume or volumes at the end at the `lvcreate` command line. The following command creates a logical volume named `testlv` in volume group `testvg` allocated from the physical volume `/dev/sdg1`.

```
lvcreate -L 1500 -n testlv testvg /dev/sdg1
```

You can specify which extents of a physical volume are to be used for a logical volume. The following example creates a linear logical volume out of extents 0 through 25 of physical volume `/dev/sda1` and extents 50 through 125 of physical volume `/dev/sdb1` in volume group `testvg`.
lvcreate -l 100 -n testlv testvg /dev/sda1:0-25 /dev/sdb1:50-125

The following example creates a linear logical volume out of extents 0 through 25 of physical volume /dev/sda1 and then continues laying out the logical volume at extent 100.

lvcreate -l 100 -n testlv testvg /dev/sda1:0-25:100-

The default policy for how the extents of a logical volume are allocated is inherit, which applies the same policy as for the volume group. These policies can be changed using the lvchange command. For information on allocation policies, see Section 3.1, “Creating Volume Groups”.

### 4.1.2. Creating Striped Volumes

For large sequential reads and writes, creating a striped logical volume can improve the efficiency of the data I/O. For general information about striped volumes, see Section 3.2, “Striped Logical Volumes”.

When you create a striped logical volume, you specify the number of stripes with the -i argument of the lvcreate Command. This determines over how many physical volumes the logical volume will be striped. The number of stripes cannot be greater than the number of physical volumes in the volume group (unless the --alloc anywhere argument is used).

The stripe size should be tuned to a power of 2 between 4kB and 512kB, and matched to the application's I/O that is using the striped volume. The -I argument of the lvcreate command specifies the stripe size in kilobytes.

If the underlying physical devices that make up a striped logical volume are different sizes, the maximum size of the striped volume is determined by the smallest underlying device. For example, in a two-legged stripe, the maximum size is twice the size of the smaller device. In a three-legged stripe, the maximum size is three times the size of the smallest device.

The following command creates a striped logical volume across 2 physical volumes with a stride of 64kB. The logical volume is 50 gigabytes in size, is named gfs1v, and is carved out of volume group vg0.

```
lvcreate -L 50G -i2 -I64 -n gfs1v vg0
```

As with linear volumes, you can specify the extents of the physical volume that you are using for the stripe. The following command creates a striped volume 100 extents in size that stripes across two physical volumes, is named stripelv and is in volume group testvg. The stripe will use sectors 0-50 of /dev/sda1 and sectors 50-100 of /dev/sdb1.

```
# lvcreate -l 100 -i2 -nstripelv testvg /dev/sda1:0-50 /dev/sdb1:50-100
Using default stripesize 64.00 KB
Logical volume "stripelv" created
```

### 4.1.3. Creating Mirrored Volumes

When you create a mirrored volume, you specify the number of copies of the data to make with the -m argument of the lvcreate command. Specifying -ml creates one mirror, which yields two
copies of the file system: a linear logical volume plus one copy. Similarly, specifying \texttt{-m2} creates two mirrors, yielding three copies of the file system.

The following command creates a mirrored logical volume with a single mirror. The volume is 50 gigabytes in size, is named \texttt{mirrorlv}, and is carved out of volume group \texttt{vg0}:

\begin{verbatim}
 lvcreate -L 50G -m1 -n gfslv vg0
\end{verbatim}

An LVM mirror divides the device being copied into regions that, by default, are 512KB in size. You can use the \texttt{-R} argument to specify the region size in MB. LVM maintains a small log which it uses to keep track of which regions are in sync with the mirror or mirrors. By default, this log is kept on disk, which keeps it persistent across reboots. You can specify instead that this log be kept in memory with the \texttt{--corelog} argument; this eliminates the need for an extra log device, but it requires that the entire mirror be resynchronized at every reboot.

The following command creates a mirrored logical volume from the volume group \texttt{bigvg}. The logical is named \texttt{ondiskmirvol} and has a single mirror. The volume is 12MB in size and keeps the mirror log in memory.

\begin{verbatim}
 # lvcreate -L 12MB -m1 --corelog -n ondiskmirvol bigvg
 Logical volume "ondiskmirvol" created
\end{verbatim}

When a mirror is created, the mirror regions are synchronized. For large mirror components, the sync process may take a long time. When you are creating a new mirror that does not need to be revived, you can specify the \texttt{nosync} argument to indicate that an initial synchronization from the first device is not required.

You can specify which devices to use for the mirror logs and log, and which extents of the devices to use. To force the log onto a particular disk, specify exactly one extent on the disk on which it will be placed. LVM does not necessarily respect the order in which devices are listed in the command line. If any physical volumes are listed that is the only space on which allocation will take place. Any physical extents included in the list that are already allocated will get ignored.

The following command creates a mirrored logical volume with a single mirror. The volume is 500 megabytes in size, it is named \texttt{mirrorlv}, and it is carved out of volume group \texttt{vg0}. The first leg of the mirror is on device \texttt{/dev/sda1}, the second leg of the mirror is on device \texttt{/dev/sdb1}, and the mirror log is on \texttt{/dev/sdc1}.

\begin{verbatim}
 lvcreate -L 500M -m1 -n mirrorlv vg0 /dev/sda1 /dev/sdb1 /dev/sdc1
\end{verbatim}

The following command creates a mirrored logical volume with a single mirror. The volume is 500 megabytes in size, it is named \texttt{mirrorlv}, and it is carved out of volume group \texttt{vg0}. The first leg of the mirror is on extents 0 through 499 of device \texttt{/dev/sda1}, the second leg of the mirror is on extents 0 through 499 of device \texttt{/dev/sdb1}, and the mirror log starts on extent 0 of device \texttt{/dev/sdc1}. These are 1MB extents. If any of the specified extents have already been allocated, they will be ignored.

\begin{verbatim}
 lvcreate -L 500M -m1 -n mirrorlv vg0 /dev/sda1:0-499 /dev/sdb1:0-499 /dev/sdc1:0
\end{verbatim}
4.1.4. Changing Mirrored Volume Configuration

You can convert a logical volume from a mirrored volume to a linear volume or from a linear volume to a mirrored volume with the `lvconvert` command. You can also use this command to reconfigure other mirror parameters of an existing logical volume, such as `corelog`.

When you convert a logical volume to a mirrored volume, you are basically creating mirror legs for an existing volume. This means that your volume group must contain the devices and space for the mirror legs and for the mirror log.

If you lose a leg of a mirror, LVM converts the volume to a linear volume so that you still have access to the volume, without the mirror redundancy. After you replace the leg, you can use the `lvconvert` command to restore the mirror. This procedure is provided in Section 3, “Recovering from LVM Mirror Failure”.

The following command converts the linear logical volume `vg00/lvol1` to a mirrored logical volume.

```
 lvconvert -m1 vg00/lvol1
```

The following command converts the mirrored logical volume `vg00/lvol1` to a linear logical volume, removing the mirror leg.

```
 lvconvert -m0 vg00/lvol1
```

4.2. Persistent Device Numbers

Major and minor device numbers are allocated dynamically at module load. Some applications work best if the block device always is activated with the same device (major and minor) number. You can specify these with the `lvcreate` and the `lvchange` commands by using the following arguments:

```
 --persistent y --major major --minor minor
```

Use a large minor number to be sure that it hasn't already been allocated to another device dynamically.

If you are exporting a file system using NFS, specifying the `fsid` parameter in the exports file may avoid the need to set a persistent device number within LVM.

4.3. Resizing Logical Volumes

To change the size of a logical volume, use the `lvreduce` command. If the logical volume contains a file system, be sure to reduce the file system first (or use the LVM GUI) so that the logical volume is always at least as large as the file system expects it to be.

The following command reduces the size of logical volume `lvol1` in volume group `vg00` by 3 logical extents.
4.4. Changing the Parameters of a Logical Volume Group

To change the parameters of a logical volume, use the `lvchange` command. For a listing of the parameters you can change, see the `lvchange(8)` man page.

You can use the `lvchange` command to activate and deactivate logical volumes. To activate and deactivate all the logical volumes in a volume group at the same time, use the `vgchange` command, as described in Section 3.6, “Changing the Parameters of a Volume Group”.

The following command changes the permission on volume `lvoll` in volume group `vg00` to be read-only.

```
 lvchange -pr vg00/lvol1
```

4.5. Renaming Logical Volumes

To rename an existing logical volume, use the `lvrename` command.

Either of the following commands renames logical volume `lvold` in volume group `vg02` to `lvnew`.

```
 lvrename /dev/vg02/lvold /dev/vg02/lvnew
 lvrename vg02 lvold lvnew
```

For more information on activating logical volumes on individual nodes in a cluster, see Section 8, “Activating Logical Volumes on Individual Nodes in a Cluster”.

4.6. Removing Logical Volumes

To remove an inactive logical volume, use the `lvremove` command. You must close a logical volume with the `umount` command before it can be removed. In addition, in a clustered environment you must deactivate a logical volume before it can be removed.

If the logical volume is currently mounted, unmount the volume before removing it.

The following command removes the logical volume `/dev/testvg/testlv` from the volume group `testvg`. Note that in this case the logical volume has not been deactivated.

```
 [root@tng3-1 lvm]# lvremove /dev/testvg/testlv
 Do you really want to remove active logical volume "testlv"? [y/n]: y
 Logical volume "testlv" successfully removed
```

You could explicitly deactivate the logical volume before removing it with the `lvchange -an` command, in which case you would not see the prompt verifying whether you want to remove an active logical volume.

4.7. Displaying Logical Volumes
There are three commands you can use to display properties of LVM logical volumes: `lvs`, `lvdisplay`, and `lvscan`.

The `lvs` command provides logical volume information in a configurable form, displaying one line per logical volume. The `lvs` command provides a great deal of format control, and is useful for scripting. For information on using the `lvs` command to customize your output, see Section 9, “Customized Reporting for LVM”.

The `lvdisplay` command displays logical volume properties (such as size, layout, and mapping) in a fixed format.

The following command shows the attributes of `lvol2` in `vg00`. If snapshot logical volumes have been created for this original logical volume, this command shows a list of all snapshot logical volumes and their status (active or inactive) as well.

```
lvdisplay -v /dev/vg00/lvol2
```

The `lvscan` command scans for all logical volumes in the system and lists them, as in the following example.

```
# lvscan
ACTIVE '/dev/vg0/gfs1v' [1.46 GB] inherit
```

### 4.8. Growing Logical Volumes

To increase the size of a logical volume, use the `lvextend` command.

After extending the logical volume, you will need to increase the size of the associated file system to match.

When you extend the logical volume, you can indicate how much you want to extend the volume, or how large you want it to be after you extend it.

The following command extends the logical volume `/dev/myvg/homevol` to 12 gigabytes.

```
# lvextend -L12G /dev/myvg/homevol
lvextend -- extending logical volume "/dev/myvg/homevol" to 12 GB
lvextend -- doing automatic backup of volume group "myvg"
lvextend -- logical volume "/dev/myvg/homevol" successfully extended
```

The following command adds another gigabyte to the logical volume `/dev/myvg/homevol`.

```
# lvextend -L+1G /dev/myvg/homevol
lvextend -- extending logical volume "/dev/myvg/homevol" to 13 GB
lvextend -- doing automatic backup of volume group "myvg"
lvextend -- logical volume "/dev/myvg/homevol" successfully extended
```

As with the `lvcreate` command, you can use the `-l` argument of the `lvextend` command to specify the number of extents by which to increase the size of the logical volume. You can also use this argument to specify a percentage of the volume group, or a percentage of the remaining free space in the volume group. The following command extends the logical volume called

---

34
testlv to fill all of the unallocated space in the volume group myvg.

```
[root@tng3-1 ~]# lvextend -l +100%FREE /dev/myvg/testlv
Extending logical volume testlv to 68.59 GB
Logical volume testlv successfully resized
```

After you have extended the logical volume it is necessary to increase the file system size to match.

By default, most file system resizing tools will increase the size of the file system to be the size of the underlying logical volume so you do not need to worry about specifying the same size for each of the two commands.

**4.9. Extending a Striped Volume**

In order to increase the size of a striped logical volume, there must be enough free space on the underlying physical volumes that make up the volume group to support the stripe. For example, if you have a two-way stripe that uses up an entire volume group, adding a single physical volume to the volume group will not enable you to extend the stripe. Instead, you must add at least two physical volumes to the volume group.

For example, consider a volume group vg that consists of two underlying physical volumes, as displayed with the following vgs command.

```
# vgs
VG   #PV #LV #SN Attr   VSize  VFree
vg   2  0  0 wz--n-  271.31G 271.31G
```

You can create a stripe using the entire amount of space in the volume group.

```
# lvcreate -n stripe1 -L 271.31G -i 2 vg
Using default stripesize 64.00 KB
Rounding up size to full physical extent 271.31 GB
Logical volume "stripe1" created
```

```
# lvs -a -o +devices
LV  VG  Attr  LSize Origin Snap% Move Log Copy% Devices
stripe1 vg -wi-a- 271.31G /dev/sda1(0),/dev/sdb1(0)
```

Note that the volume group now has no more free space.

```
# vgs
VG   #PV #LV #SN Attr   VSize  VFree
vg   2  1  0 wz--n-  271.31G 0
```

The following command adds another physical volume to the volume group, which then has 135G of additional space.

```
# vgextend vg /dev/sdc1
Volume group "vg" successfully extended
```

```
# vgs
VG   #PV #LV #SN Attr   VSize  VFree
vg   3  1  0 wz--n-  406.97G 135.66G
```
At this point you cannot extend the striped logical volume to the full size of the volume group, because two underlying devices are needed in order to stripe the data.

```bash
lvextend vg/stripe1 -L 406G

Using stripesize of last segment 64.00 KB
Extending logical volume stripe1 to 406.00 GB
Insufficient suitable allocatable extents for logical volume stripe1: 34480 more required
```

To extend the striped logical volume, add another physical volume and then extend the logical volume. In this example, having added two physical volumes to the volume group we can extend the logical volume 5A to the full size of the volume group.

```bash
vgextend vg /dev/sdd1

Volume group "vg" successfully extended
```

```bash
vgs

<table>
<thead>
<tr>
<th>VG</th>
<th>PV</th>
<th>LV</th>
<th>SN</th>
<th>Attr</th>
<th>VSize</th>
<th>VFree</th>
</tr>
</thead>
<tbody>
<tr>
<td>vg</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>wz--n-</td>
<td>542.62G</td>
<td>271.31G</td>
</tr>
</tbody>
</table>
```

```bash
lvextend vg/stripe1 -L 542G

Using stripesize of last segment 64.00 KB
Extending logical volume stripe1 to 542.00 GB
Logical volume stripe1 successfully resized
```

If you do not have enough underlying physical devices to extend the striped logical volume, it is possible to extend the volume anyway if it does not matter that the extension is not striped, which may result in uneven performance. When adding space to the logical volume, the default operation is to use the same striping parameters of the last segment of the existing logical volume, but you can override those parameters. The following example extends the existing striped logical volume to use the remaining free space after the initial `lvextend` command fails.

```bash
lvextend vg/stripe1 -L 406G

Using stripesize of last segment 64.00 KB
Extending logical volume stripe1 to 406.00 GB
Insufficient suitable allocatable extents for logical volume stripe1: 34480 more required
```

```bash
lvextend -il -l+100%FREE vg/stripe1
```

### 4.10. Shrinking Logical Volumes

To reduce the size of a logical volume, first unmount the file system. You can then use the `lvreduce` command to shrink the volume. After shrinking the volume, remount the file system.

**Caution**

It is important to reduce the size of the file system or whatever is residing in the volume before shrinking the volume itself, otherwise you risk losing data.

Shrinking a logical volume frees some of the volume group to be allocated to other logical volumes in the volume group.

The following example reduces the size of logical volume `lvol1` in volume group `vg00` by 3 logical-
5. Creating Snapshot Volumes

Use the `--s` argument of the `lvcreate` command to create a snapshot volume. A snapshot volume is writeable.

LVM snapshots are not cluster-aware, so they require exclusive access to a volume. For information on activating logical volumes on individual nodes in a cluster, see Section 8, “Activating Logical Volumes on Individual Nodes in a Cluster”.

The following command creates a snapshot logical volume that is 100 megabytes in size named `/dev/vg00/snap`. This creates a snapshot of the origin logical volume named `/dev/vg00/lvol1`. If the original logical volume contains a file system, you can mount the snapshot logical volume on an arbitrary directory in order to access the contents of the file system to run a backup while the original file system continues to get updated.

```
 lvcreate --size 100M --snapshot --name snap /dev/vg00/lvol1
```

After you create a snapshot logical volume, specifying the origin volume on the `lvdisplay` command yields output that includes a list of all snapshot logical volumes and their status (active or inactive).

The following example shows the status of the logical volume `/dev/new_vg/lvol0`, for which a snapshot volume `/dev/new_vg/newvgsnap` has been created.

```
# lvdisplay /dev/new_vg/lvol0
--- Logical volume ---
LV Name /dev/new_vg/lvol0
VG Name new_vg
LV UUID LBy1Tz-sr23-OjsI-LT03-nHLC-y8XW-EhCl78
LV Write Access read/write
LV snapshot status source of
   /dev/new_vg/newvgsnap1 [active]
LV Status available
# open 0
LV Size 52.00 MB
Current LE 13
Segments 1
Allocation inherit
Read ahead sectors 0
Block device 253:2
```

The `lvs` command, by default, displays the origin volume and the current percentage of the snapshot volume being used for each snapshot volume. The following example shows the default output for the `lvs` command for a system that includes the logical volume `/dev/new_vg/lvol0`, for which a snapshot volume `/dev/new_vg/newvgsnap` has been created.

```
# lvs
LV     VG  Attr  LSize Origin Snap%  Move Log Copy%
lvol0  new_vg owi-a  52.00M
newvgsnap1 new_vg svi-a  8.00M lvol0  0.20
```
Because the snapshot increases in size as the origin volume changes, it is important to monitor the percentage of the snapshot volume regularly with the `lvs` command to be sure it does not fill. A snapshot that is 100% full is lost completely, as write to unchanged parts of the origin in would be unable to succeed without corrupting the snapshot.

6. Controlling LVM Device Scans with Filters

At startup, the `vgscan` command is run to scan the block devices on the system looking for LVM labels, to determine which of them are physical volumes and to read the metadata and build up a list of volume groups. The names of the physical volumes are stored in the cache file of each node in the system, `/etc/lvm/.cache`. Subsequent commands may read that file to avoiding rescanning.

You can control which devices LVM scans by setting up filters in the `lvm.conf` configuration file. The filters consist of a series of simple regular expressions that get applied to the device names in the `/dev` directory to decide whether to accept or reject each block device found.

The examples below show the use of filters to control which devices LVM scans. Note that some of these examples do not necessarily represent best practice, as the regular expressions are matched freely against the complete pathname. For example, `a/loop/` is equivalent to `a/.*loop.*/` and would match `/dev/solooperation/lvol1`.

The following filter adds all discovered devices, which is the default behavior as there is no filter configured in the configuration file:

```bash
filter = [ "a/.*/" ]
```

The following filter removes the `cdrom` device in order to avoid delays if the drive contains no media:

```bash
filter = [ "r|/dev/cdrom|" ]
```

The following filter adds all loop and removes all other block devices:

```bash
filter = [ "a/loop.*", "r/.*" ]
```

The following filter adds all loop and IDE and removes all other block devices:

```bash
filter = [ "a/loop.*", "a|/dev/hd.*", "r/.*" ]
```

The following filter adds just partition 8 on the first IDE drive and removes all other block devices:
7. Online Data Relocation

You can move data while the system is in use with the `pvmove` command.

The `pvmove` command breaks up the data to be moved into sections and creates a temporary mirror to move each section. For more information on the operation of the `pvmove` command, see the `pvmove(8)` man page.

Because the `pvmove` command uses mirroring, it is not cluster-aware and needs exclusive access to a volume. For information on activating logical volumes on individual nodes in a cluster, see Section 8, “Activating Logical Volumes on Individual Nodes in a Cluster”.

The following command moves all allocated space off the physical volume `/dev/sdc1` to other free physical volumes in the volume group:

```
pvmove /dev/sdc1
```

The following command moves just the extents of the logical volume `MyLV`.

```
pvmove -n MyLV /dev/sdc1
```

Since the `pvmove` command can take a long time to execute, you may want to run the command in the background to avoid display of progress updates in the foreground. The following command moves all extents allocated to the physical volume `/dev/sdc1` over to `/dev/sdf1` in the background.

```
pvmove -b /dev/sdc1 /dev/sdf1
```

The following command reports the progress of the move as a percentage at five second intervals.

```
pvmove -i5 /dev/sdd1
```

8. Activating Logical Volumes on Individual Nodes in a Cluster

If you have LVM installed in a cluster environment, you may at times need to activate logical volumes exclusively on one node. For example, the `pvmove` command is not cluster-aware and needs exclusive access to a volume. LVM snapshots require exclusive access to a volume as well.
To activate logical volumes exclusively on one node, use the `lvchange -aey` command. Alternatively, you can use `lvchange -aly` command to activate logical volumes only on the local node but not exclusively. You can later activate them on additional nodes concurrently.

You can also activate logical volumes on individual nodes by using LVM tags, which are described in Appendix C, *LVM Object Tags*. You can also specify activation of nodes in the configuration file, which is described in Appendix B, *The LVM Configuration Files*.

### 9. Customized Reporting for LVM

You can produce concise and customizable reports of LVM objects with the `pvs`, `lvs`, and `vgs` commands. The reports that these commands generate include one line of output for each object. Each line contains an ordered list of fields of properties related to the object. There are five ways to select the objects to be reported: by physical volume, volume group, logical volume, physical volume segment, and logical volume segment.

The following sections provide:

- A summary of command arguments you can use to control the format of the generated report.
- A list of the fields you can select for each LVM object.
- A summary of command arguments you can use to sort the generated report.
- Instructions for specifying the units of the report output.

#### 9.1. Format Control

Whether you use the `pvs`, `lvs`, or `vgs` command determines the default set of fields displayed and the sort order. You can control the output of these commands with the following arguments:

- You can change what fields are displayed to something other than the default by using the `-o` argument. For example, the following output is the default display for the `pvs` command (which displays information about physical volumes).

```bash
# pvs
PV VG Fmt Attr PSize PFree
/dev/sdb1 new_vg lvm2 a- 17.14G 17.14G
/dev/sdc1 new_vg lvm2 a- 17.14G 17.09G
/dev/sdd1 new_vg lvm2 a- 17.14G 17.14G
```

The following command displays only the physical volume name and size.

```bash
# pvs -o pv_name,pv_size
PV PSize
/dev/sdb1 17.14G
/dev/sdc1 17.14G
/dev/sdd1 17.14G
```

- You can append a field to the output with the plus sign (+), which is used in combination with
the -o argument.

The following example displays the UUID of the physical volume in addition to the default fields.

```
# pvs -o +pv_uuid
PV  VG  Fmt  Attr  PSize  PFree  PV UUID
/dev/sdb1 new_vg lvm2 a-  17.14G 17.14G onFF2w-1fLC-ughJ-D9eB-M7iv-6XqA-dqGeXY
/dev/sdc1 new_vg lvm2 a-  17.14G 17.09G Joqlch-yWSj-kuEn-IdwM-01S9-XO8M-mcpsVe
/dev/sdd1 new_vg lvm2 a-  17.14G 17.14G yvfvZK-Cf31-j75k-dECm-0RZ3-0dGW-tUqkCS
```

• Adding the -v argument to a command includes some extra fields. For example, the pvs -v command will display the DevSize and PV UUID fields in addition to the default fields.

```
# pvs -v
Scanning for physical volume names
PV  VG  Fmt  Attr  PSize  PFree  DevSize  PV UUID
/dev/sdb1 new_vg lvm2 a-  17.14G 17.14G 17.14G onFF2w-1fLC-ughJ-D9eB-M7iv-6XqA-dqGeXY
/dev/sdc1 new_vg lvm2 a-  17.14G 17.09G 17.14G Joqlch-yWSj-kuEn-IdwM-01S9-XO8M-mcpsVe
/dev/sdd1 new_vg lvm2 a-  17.14G 17.14G 17.14G yvfvZK-Cf31-j75k-dECm-0RZ3-0dGW-tUqkCS
```

• The --noheadings argument suppresses the headings line. This can be useful for writing scripts.

The following example uses the --noheadings argument in combination with the pv_name argument, which will generate a list of all physical volumes.

```
# pvs --noheadings -o pv_name
/dev/sdb1
/dev/sdc1
/dev/sdd1
```

• The --separator separator argument uses separator to separate each field. This can be useful in a script if you are running a grep command on the output.

The following example separates the default output fields of the pvs command with an equals sign (=).

```
# pvs --separator =
PV=VG=Fmt=Attr=PSize=PFree
/dev/sdb1=new_vg=lvm2=a-=17.14G=17.14G
/dev/sdc1=new_vg=lvm2=a-=17.14G=17.09G
/dev/sdd1=new_vg=lvm2=a-=17.14G=17.14G
```

To keep the fields aligned when using the separator argument, use the --aligned argument.

```
# pvs --separator = --aligned
PV =VG =Fmt =Attr =PSize =PFree
/dev/sdb1 =new_vg =lvm2 =a- =17.14G =17.14G
/dev/sdc1 =new_vg =lvm2 =a- =17.14G =17.09G
/dev/sdd1 =new_vg =lvm2 =a- =17.14G =17.14G
```

You can use the -P argument of the lvs or vgs command to display information about a failed
volume that would otherwise not appear in the output. For information on the output this argument yields, see Section 2, “Displaying Information on Failed Devices”.

For a full listing of display arguments, see the `pvs(8)`, `vgs(8)` and `lvs(8)` man pages.

Volume group fields can be mixed with either physical volume (and physical volume segment) fields or with logical volume (and logical volume segment) fields, but physical volume and logical volume fields cannot be mixed. For example, the following command will display one line of output for each physical volume.

```
# vgs -o +pv_name
VG #PV #LV #SN Attr VSize VFree PV
new_vg 3 1 0 wz--n- 51.42G 51.37G /dev/sdc1
new_vg 3 1 0 wz--n- 51.42G 51.37G /dev/sdd1
new_vg 3 1 0 wz--n- 51.42G 51.37G /dev/sdb1
```

### 9.2. Object Selection

This section provides a series of tables that list the information you can display about the LVM objects with the `pvs`, `vgs`, and `lvs` commands.

For convenience, a field name prefix can be dropped if it matches the default for the command. For example, with the `pvs` command, `name` means `pv_name`, but with the `vgs` command, `name` is interpreted as `vg_name`.

Executing the following command is the equivalent of executing `pvs -o pv_free`.

```
# pvs -o free
PFree
17.14G
17.09G
17.14G
```

### The pvs Command

Table 4.1, “pvs Display Fields” lists the display arguments of the `pvs` command, along with the field name as it appears in the header display and a description of the field.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dev_size</td>
<td>DevSize</td>
<td>The size of the underlying device on which the physical volume was created</td>
</tr>
<tr>
<td>pe_start</td>
<td>1st PE</td>
<td>Offset to the start of the first physical extent in the underlying device</td>
</tr>
<tr>
<td>pv_attr</td>
<td>Attr</td>
<td>Status of the physical volume: (a)locatable or e(x)ported.</td>
</tr>
<tr>
<td>pv_fmt</td>
<td>Fmt</td>
<td>The metadata format of the physical volume <code>(</code>lvm2<code>or</code>lvm1`)</td>
</tr>
<tr>
<td>pv_free</td>
<td>PFree</td>
<td>The free space remaining on the physical volume</td>
</tr>
<tr>
<td>pv_name</td>
<td>PV</td>
<td>The physical volume name</td>
</tr>
<tr>
<td>pv_pe_alloc_count</td>
<td>Alloc</td>
<td>Number of used physical extents</td>
</tr>
</tbody>
</table>
9.2. Object Selection

<table>
<thead>
<tr>
<th>Argument</th>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pv_pe_count</td>
<td>PE</td>
<td>Number of physical extents</td>
</tr>
<tr>
<td>pvseg_size</td>
<td>SSize</td>
<td>The segment size of the physical volume</td>
</tr>
<tr>
<td>pvseg_start</td>
<td>Start</td>
<td>The starting physical extent of the physical volume segment</td>
</tr>
<tr>
<td>pv_size</td>
<td>PSize</td>
<td>The size of the physical volume</td>
</tr>
<tr>
<td>pv_tags</td>
<td>PV Tags</td>
<td>LVM tags attached to the physical volume</td>
</tr>
<tr>
<td>pv_used</td>
<td>Used</td>
<td>The amount of space currently used on the physical volume</td>
</tr>
<tr>
<td>pv_uuid</td>
<td>PV UUID</td>
<td>The UUID of the physical volume</td>
</tr>
</tbody>
</table>

Table 4.1. pvs Display Fields

The `pvs` command displays the following fields by default: `pv_name`, `vg_name`, `pv_fmt`, `pv_attr`, `pv_size`, `pv_free`. The display is sorted by `pv_name`.

```
# pvs
PV     VG     Fmt    Attr PSize PFree
/dev/sdb1 new_vg lvm2 a- 17.14G 17.14G
/dev/sdc1 new_vg lvm2 a- 17.14G 17.09G
/dev/sdd1 new_vg lvm2 a- 17.14G 17.13G
```

Using the `--v` argument with the `pvs` command adds the following fields to the default display: `dev_size`, `pv_uuid`.

```
# pvs --v
Scanning for physical volume names
PV     VG     Fmt    Attr PSize DevSize PFree Dev UUID
/dev/sdb1 new_vg lvm2 a- 17.14G 17.14G 17.14G onFF2w-1fLC-ughJ-D9eB-M7iv-6XqA-dqGeXY
/dev/sdc1 new_vg lvm2 a- 17.14G 17.09G 17.14G Jqglch-yWSj-kuEn-IdwM-01S9-XO8M-mcpsVe
/dev/sdd1 new_vg lvm2 a- 17.14G 17.13G 17.14G yvfvZK-Cf31-j75k-dECm-0RZ3-0dGW-tUqkCS
```

You can use the `--segments` argument of the `pvs` command to display information about each physical volume segment. A segment is a group of extents. A segment view can be useful if you want to see whether your logical volume is fragmented.

The `pvs --segments` command displays the following fields by default: `pv_name`, `vg_name`, `pv_fmt`, `pv_attr`, `pv_size`, `pv_free`, `pvseg_start`, `pvseg_size`. The display is sorted by `pv_name` and `pvseg_size` within the physical volume.

```
# pvs --segments
PV     VG     Fmt    Attr PSize Start SSize
/dev/hda2 VolGroup00 lvm2 a- 37.16G 32.00M 0 1172
/dev/hda2 VolGroup00 lvm2 a- 37.16G 32.00M 1172 16
/dev/hda2 VolGroup00 lvm2 a- 37.16G 32.00M 1188 1
/dev/sda1 vg     lvm2 a- 17.14G 16.75G 0 26
/dev/sda1 vg     lvm2 a- 17.14G 16.75G 26 24
/dev/sda1 vg     lvm2 a- 17.14G 16.75G 50 26
/dev/sda1 vg     lvm2 a- 17.14G 16.75G 76 24
/dev/sda1 vg     lvm2 a- 17.14G 16.75G 100 26
```
You can use the `pvs -a` command to see devices detected by LVM that have not been initialized as LVM physical volumes.

```
# pvs -a
PV     VG Fmt Attr PSize PFree
/dev/VolGroup00/LogVol01 -- 0 0
/dev/new_vg/lvol0 -- 0 0
/dev/ram -- 0 0
/dev/ram0 -- 0 0
/dev/ram2 -- 0 0
/dev/ram3 -- 0 0
/dev/ram4 -- 0 0
/dev/ram5 -- 0 0
/dev/ram6 -- 0 0
/dev/root -- 0 0
/dev/sda -- 0 0
/dev/sdb -- 0 0
/dev/sdb1 new_vg lvm2 a- 17.14G 17.14G
/dev/sdc -- 0 0
/dev/sdc1 new_vg lvm2 a- 17.14G 17.09G
/dev/sdd -- 0 0
/dev/sdd1 new_vg lvm2 a- 17.14G 17.14G
```

### The vgs Command

Table 4.2, “vgs Display Fields” lists the display arguments of the `vgs` command, along with the field name as it appears in the header display and a description of the field.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lv_count</td>
<td>#LV</td>
<td>The number of logical volumes the volume group contains</td>
</tr>
<tr>
<td>max_lv</td>
<td>MaxLV</td>
<td>The maximum number of logical volumes allowed in the volume group (0 if unlimited)</td>
</tr>
<tr>
<td>max_pv</td>
<td>MaxPV</td>
<td>The maximum number of physical volumes allowed in the volume group (0 if unlimited)</td>
</tr>
<tr>
<td>pv_count</td>
<td>#PV</td>
<td>The number of physical volumes that define the volume group</td>
</tr>
<tr>
<td>snap_count</td>
<td>#SN</td>
<td>The number of snapshots the volume group contains</td>
</tr>
<tr>
<td>vg_attr</td>
<td>Attr</td>
<td>Status of the volume group: (w)riteable, (r)eadonly, resi(z)etable, e(x)ported, (p)artial and (c)lustered.</td>
</tr>
<tr>
<td>vg_extent_count</td>
<td>#Ext</td>
<td>The number of physical extents in the volume group</td>
</tr>
<tr>
<td>vg_extent_size</td>
<td>Ext</td>
<td>The size of the physical extents in the volume group</td>
</tr>
</tbody>
</table>
Table 4.2. vgs Display Fields

The `vgs` command displays the following fields by default: `vg_name`, `pv_count`, `lv_count`, `snap_count`, `vg_attr`, `vg_size`, `vg_free`. The display is sorted by `vg_name`.  

```
vgs
VG  #PV  #LV  #SN  Attr  VSize  VFree
new_vg  3   1   1  wz--n-  51.42G  51.36G
```

Using the `-v` argument with the `vgs` command adds the following fields to the default display: `vg_extent_size`, `vg_uuid`.

```
vgs -v
Finding all volume groups
Finding volume group "new_vg"
VG  Attr  Ext  #PV  #LV  #SN  VSize  VFree  VG UUID
new_vg  wz--n-  4.00M  3   1   1  51.42G  51.36G  jxQJ0a-ZKk0-OpMO-0118-nlwO-wwqd-fD5D32
```

The lvs Command

Table 4.3, “lvs Display Fields” lists the display arguments of the `lvs` command, along with the field name as it appears in the header display and a description of the field.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>chunksize</td>
<td>Chunk</td>
<td>Unit size in a snapshot volume</td>
</tr>
<tr>
<td>chunk_size</td>
<td>Chunk</td>
<td>Unit size in a snapshot volume</td>
</tr>
<tr>
<td>copy_percent</td>
<td>Copy%</td>
<td>The synchronization percentage of a mirrored logical volume; also used when physical extents are being moved with the <code>pv_move</code> command</td>
</tr>
<tr>
<td>devices</td>
<td>Devices</td>
<td>The underlying devices that make up the logical volume: the</td>
</tr>
<tr>
<td>Argument</td>
<td>Header</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>lv_attr</td>
<td>Attr</td>
<td>The status of the logical volume. The logical volume attribute bits are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 1: Volume type: (m)irrored, (M)irrored without initial sync, (o)rigin, (p)vmove, (s)napshot, invalid (S)napshot, (v)irtual</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 2: Permissions: (w)ritable, (r)ead-only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 3: Allocation policy: (c)ontiguous, (n)ormal, (a)nywhere, (i)nherited. This is capitalized if the volume is currently locked against allocation changes, for example while executing the pvmove command.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 4: fixed (m)inor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 5 State: (a)ctive, (s)uspended, (l)invalid snapshot, invalid (S)uspended snapshot, mapped (d)evice present without tables, mapped device present with (i)nactive table</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 6: device (o)pen</td>
</tr>
<tr>
<td>lv_kernel_major</td>
<td>KMaj</td>
<td>Actual major device number of the logical volume (-1 if inactive)</td>
</tr>
<tr>
<td>lv_kernel_minor</td>
<td>KMIN</td>
<td>Actual minor device number of the logical volume (-1 if inactive)</td>
</tr>
<tr>
<td>lv_major</td>
<td>Maj</td>
<td>The persistent major device number of the logical volume (-1 if not specified)</td>
</tr>
<tr>
<td>lv_minor</td>
<td>Min</td>
<td>The persistent minor device number of the logical volume (-1 if not specified)</td>
</tr>
<tr>
<td>lv_name</td>
<td>LV</td>
<td>The name of the logical volume</td>
</tr>
<tr>
<td>lv_size</td>
<td>LSize</td>
<td>The size of the logical volume</td>
</tr>
<tr>
<td>lv_tags</td>
<td>LV Tags</td>
<td>LVM tags attached to the logical volume</td>
</tr>
<tr>
<td>lv_uuid</td>
<td>LV UUID</td>
<td>The UUID of the logical volume</td>
</tr>
<tr>
<td>mirror_log</td>
<td>Log</td>
<td>Device on which the mirror log resides</td>
</tr>
<tr>
<td>modules</td>
<td>Modules</td>
<td>Corresponding kernel device-mapper target necessary to use this logical volume</td>
</tr>
<tr>
<td>move_pv</td>
<td>Move</td>
<td>Source physical volume of a temporary logical volume created with the pvmove command</td>
</tr>
<tr>
<td>origin</td>
<td>Origin</td>
<td>The origin device of a snapshot volume</td>
</tr>
<tr>
<td>region_size</td>
<td>Region</td>
<td>The unit size of a mirrored logical volume</td>
</tr>
<tr>
<td>seg_count</td>
<td>#Seg</td>
<td>The number of segments in the logical volume</td>
</tr>
</tbody>
</table>

9.2. Object Selection
<table>
<thead>
<tr>
<th>Argument</th>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>seg_size</td>
<td>SSize</td>
<td>The size of the segments in the logical volume</td>
</tr>
<tr>
<td>seg_start</td>
<td>Start</td>
<td>Offset of the segment in the logical volume</td>
</tr>
<tr>
<td>seg_tags</td>
<td>Seg Tags</td>
<td>LVM tags attached to the segments of the logical volume</td>
</tr>
<tr>
<td>segtype</td>
<td>Type</td>
<td>The segment type of a logical volume (for example: mirror, striped, linear)</td>
</tr>
<tr>
<td>snap_percent</td>
<td>Snap%</td>
<td>Current percentage of a snapshot volume that is in use</td>
</tr>
<tr>
<td>stripes</td>
<td>#Str</td>
<td>Number of stripes or mirrors in a logical volume</td>
</tr>
<tr>
<td>stripesize</td>
<td>Stripe</td>
<td>Unit size of the stripe in a striped logical volume</td>
</tr>
</tbody>
</table>

Table 4.3. lvs Display Fields

The `lvs` command displays the following fields by default: `lv_name`, `vg_name`, `lv_attr`, `lv_size`, `origin`, `snap_percent`, `move_pv`, `mirror_log`, `copy_percent`. The default display is sorted by `vg_name` and `lv_name` within the volume group.

```sh
# lvs
LV VG Attr LSize Origin Snap% Move Log Copy%
lvol0 new_vg owi-a- 52.00M
newvgsnap1 new_vg swi-a- 8.00M lvol0 0.20
```

Using the `-v` argument with the `lvs` command adds the following fields to the default display: `seg_count`, `lv_major`, `lv_minor`, `lv_kernel_major`, `lv_kernel_minor`, `lv_uuid`.

```sh
# lvs -v
Finding all logical volumes
LV VG #Seg Attr LSize Maj Min KMaj KMin Origin Snap% Move Log Copy%
lvol0 new_vg 1 owi-a- 52.00M -1 -1 253 3
newvgsnap1 new_vg 1 swi-a- 8.00M -1 -1 253 5 lvol0 0.20
```

You can use the `--segments` argument of the `lvs` command to display information with default columns that emphasize the segment information. When you use the `segments` argument, the `seg` prefix is optional. The `lvs --segments` command displays the following fields by default: `lv_name`, `vg_name`, `lv_attr`, `stripes`, `segtype`, `seg_size`. The default display is sorted by `vg_name`, `lv_name` within the volume group, and `seg_start` within the logical volume. If the logical volumes were fragmented, the output from this command would show that.

```sh
# lvs --segments
LV VG Attr #Str Type SSize
LogVol100 VolGroup00 -wi-a- 1 linear 36.62G
LogVol101 VolGroup00 -wi-a- 1 linear 512.00M
lv vg -wi-a- 1 linear 104.00M
lv vg -wi-a- 1 linear 104.00M
lv vg -wi-a- 1 linear 104.00M
lv vg -wi-a- 1 linear 88.00M
```
Using the `-v` argument with the `lvs --segments` command adds the following fields to the default display: `seg_start`, `stripesize`, `chunksize`.

```
# lvs -v --segments
Fing all logical volumes
LV    VG   Attr    Start  SSize  #Str  Type  Stripe  Chunk
lvol0 new_vg  owi-a-  0  52.00M  1  linear  0  0
newvgsnap1 new_vg  swi-a-  0  8.00M  1  linear  0  8.00K
```

The following example shows the default output of the `lvs` command on a system with one logical volume configured, followed by the default output of the `lvs` command with the `segments` argument specified.

```
# lvs
LV    VG   Attr    LSize  Origin Snap%  Move  Log  Copy%
lvol0 new_vg  -wi-a-  52.00M

# lvs --segments
LV    VG   Attr    #Str  Type  SSize
lvol0 new_vg  -wi-a-  1  linear  52.00M
```

### 9.3. Sorting LVM Reports

Normally the entire output of the `lvs`, `vgs`, or `pvs` command has to be generated and stored internally before it can be sorted and columns aligned correctly. You can specify the `--unbuffered` argument to display unsorted output as soon as it is generated.

To specify an alternative ordered list of columns to sort on, use the `-O` argument of any of the reporting commands. It is not necessary to include these fields within the output itself.

The following example shows the output of the `pvs` command that displays the physical volume name, size, and free space.

```
# pvs -o pv_name,pv_size,pv_free
PV    PSize  PFree
/dev/sdb1   17.14G  17.14G
/dev/sdc1   17.14G  17.09G
/dev/sdd1   17.14G  17.14G
```

The following example shows the same output, sorted by the free space field.

```
# pvs -o pv_name,pv_size,pv_free -O pv_free
PV    PSize  PFree
/dev/sdc1   17.14G  17.09G
/dev/sdd1   17.14G  17.14G
/dev/sdb1   17.14G  17.14G
```

The following example shows that you do not need to display the field on which you are sorting.

```
# pvs -o pv_name,pv_size -O pv_free
PV    PSize
/dev/sdc1   17.14G
/dev/sdd1   17.14G
/dev/sdb1   17.14G
```
To display a reverse sort, precede a field you specify after the \texttt{-o} argument with the \texttt{-} character.

\texttt{# pvs \textasciitilde o pv\_name,pv\_size,pv\_free \textasciitilde O \textasciitilde pv\_free}

\begin{verbatim}
PV PSize PFree
/dev/sdd1 17.14G 17.14G
/dev/sdb1 17.14G 17.14G
/dev/sdc1 17.14G 17.09G
\end{verbatim}

9.4. Specifying Units

To specify the unit for the LVM report display, use the \texttt{--units} argument of the report command. You can specify \texttt{(b)ytes}, \texttt{(k)ilobytes}, \texttt{(m)egabytes}, \texttt{(g)igabytes}, \texttt{(t)erabytes}, \texttt{(e)xabytes}, \texttt{(p)etabytes}, and \texttt{(h)uman-readable}. The default display is human-readable. You can override the default by setting the \texttt{units} parameter in the \texttt{global} section of the \texttt{lvm.conf} file.

The following example specifies the output of the \texttt{pvs} command in megabytes rather than the default gigabytes.

\texttt{# pvs \textasciitilde units m}

\begin{verbatim}
PV VG Fmt Attr PSize PFree
/dev/sda1 lvm2 -- 17555.40M 17555.40M
/dev/sdb1 new\_vg lvm2 a- 17552.00M 17552.00M
/dev/sdc1 new\_vg lvm2 a- 17552.00M 17500.00M
/dev/sdd1 new\_vg lvm2 a- 17552.00M 17552.00M
\end{verbatim}

By default, units are displayed in powers of 2 (multiples of 1024). You can specify that units be displayed in multiples of 1000 by capitalizing the unit specification (B, K, M, G, T, H).

The following command displays the output as a multiple of 1024, the default behavior.

\texttt{# pvs}

\begin{verbatim}
PV VG Fmt Attr PSize PFree
/dev/sdb1 new\_vg lvm2 a- 17.14G 17.14G
/dev/sdc1 new\_vg lvm2 a- 17.14G 17.09G
/dev/sdd1 new\_vg lvm2 a- 17.14G 17.14G
\end{verbatim}

The following command displays the output as a multiple of 1000.

\texttt{# pvs \textasciitilde units G}

\begin{verbatim}
PV VG Fmt Attr PSize PFree
/dev/sdb1 new\_vg lvm2 a- 18.40G 18.40G
/dev/sdc1 new\_vg lvm2 a- 18.40G 18.35G
/dev/sdd1 new\_vg lvm2 a- 18.40G 18.40G
\end{verbatim}

You can also specify \texttt{(s)ectors} (defined as 512 bytes) or custom units.

The following example displays the output of the \texttt{pvs} command as a number of sectors.

\texttt{# pvs \textasciitilde units s}

\begin{verbatim}
PV VG Fmt Attr PSize PFree
/dev/sdb1 new\_vg lvm2 a- 35946496S 35946496S
/dev/sdc1 new\_vg lvm2 a- 35946496S 35840000S
/dev/sdd1 new\_vg lvm2 a- 35946496S 35946496S
\end{verbatim}
The following example displays the output of the `pvs` command in units of 4 megabytes.

```
# pvs --units 4m
PV   VG    Fmt  Attr  PSize   PFree
/dev/sdb1  new_vg  lvm2  a-  4388.00U  4388.00U
/dev/sdc1  new_vg  lvm2  a-  4388.00U  4375.00U
/dev/sdd1  new_vg  lvm2  a-  4388.00U  4388.00U
```
Chapter 5. LVM Configuration
Examples

This chapter provides some basic LVM configuration examples.

1. Creating an LVM Logical Volume on Three Disks

This example creates an LVM logical volume called `new_logical_volume` that consists of the disks at `/dev/sda1`, `/dev/sdb1`, and `/dev/sdc1`.

1.1. Creating the Physical Volumes

To use disks in a volume group, you label them as LVM physical volumes.

```
Caution
This command destroys any data on /dev/sda1, /dev/sdb1, and /dev/sdc1.
```

```
[root@tng3-1 ~]# pvcreate /dev/sda1 /dev/sdb1 /dev/sdc1
Physical volume "/dev/sda1" successfully created
Physical volume "/dev/sdb1" successfully created
Physical volume "/dev/sdc1" successfully created
```

1.2. Creating the Volume Group

The following command creates the volume group `new_vol_group`.

```
[root@tng3-1 ~]# vgcreate new_vol_group /dev/sda1 /dev/sdb1 /dev/sdc1
Volume group "new_vol_group" successfully created
```

You can use the `vgs` command to display the attributes of the new volume group.

```
[root@tng3-1 ~]# vgs
VG #PV #LV #SN Attr  VSize  VFree
new_vol_group 3 0 0 wz--n- 51.45G 51.45G
```

1.3. Creating the Logical Volume

The following command creates the logical volume `new_logical_volume` from the volume group `new_vol_group`. This example creates a logical volume that uses 2GB of the volume group.

```
[root@tng3-1 ~]# lvcreate -L2G -n new_logical_volume new_vol_group
```
1.4. Creating the File System

The following command creates a GFS file system on the logical volume.

```
[root@tng3-1 ~]# gfs_mkfs -plock_nolock -j 1 /dev/new_vol_group/new_logical_volume
```

This will destroy any data on /dev/new_vol_group/new_logical_volume.

Are you sure you want to proceed? [y/n]  y

Device: /dev/new_vol_group/new_logical_volume
Blocksize: 4096
Filesystem Size: 491460
Journals: 1
Resource Groups: 8
Locking Protocol: lock_nolock
Lock Table:

Syncing...
All Done

The following commands mount the logical volume and report the file system disk space usage.

```
[root@tng3-1 ~]# mount /dev/new_vol_group/new_logical_volume /mnt
[root@tng3-1 ~]# df
```

Filesystem 1K-blocks Used Available Use% Mounted on
/dev/new_vol_group/new_logical_volume 1965840 20 1965820 1% /mnt

2. Creating a Striped Logical Volume

This example creates an LVM striped logical volume called striped_logical_volume that stripes data across the disks at /dev/sda1, /dev/sdb1, and /dev/sdc1.

2.1. Creating the Physical Volumes

Label the disks you will use in the volume groups as LVM physical volumes.

```
[root@tng3-1 ~]# pvcreate /dev/sda1 /dev/sdb1 /dev/sdc1
```

Physical volume "/dev/sda1" successfully created
Physical volume "/dev/sdb1" successfully created
Physical volume "/dev/sdc1" successfully created

2.2. Creating the Volume Group

The following command creates the volume group striped_vol_group.

```
[root@tng3-1 ~]# vgcreate striped_vol_group /dev/sda1 /dev/sdb1 /dev/sdc1
```

Volume group "striped_vol_group" successfully created
2.3. Creating the Logical Volume

The following command creates the striped logical volume `striped_logical_volume` from the volume group `striped_vol_group`. This example creates a logical volume that is 2 gigabytes in size, with three stripes and a stripe size of 4 kilobytes.

```
[root@tng3-1 ~]# lvcreate -l3 -i4 -L2G -nstriped_logical_volume striped_vol_group
Rounding size (512 extents) up to stripe boundary size (513 extents)
Logical volume "striped_logical_volume" created
```

2.4. Creating the File System

The following command creates a GFS file system on the logical volume.

```
[root@tng3-1 ~]# gfs_mkfs -plock_nolock -j1 /dev/striped_vol_group/striped_logical_volume
This will destroy any data on /dev/striped_vol_group/striped_logical_volume.
Are you sure you want to proceed? [y/n] y
Device: /dev/striped_vol_group/striped_logical_volume
Blocksize: 4096
Filesystem Size: 492484
Journals: 1
Resource Groups: 8
Locking Protocol: lock_nolock
Lock Table:
Syncing...
All Done
```

The following commands mount the logical volume and report the file system disk space usage.

```
[root@tng3-1 ~]# mount /dev/striped_vol_group/striped_logical_volume /mnt
[root@tng3-1 ~]# df
Filesystem 1K-blocks Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol100 13902624 1656776 11528232 13% /
/dev/hda1 101086 10787 85080 12% /boot
tmpfs 127880 0 127880 0% /dev/shm
/dev/striped_vol_group/striped_logical_volume 1969936 20 1969916 1% /mnt
```

3. Splitting a Volume Group

In this example, an existing volume group consists of three physical volumes. If there is enough
unused space on the physical volumes, a new volume group can be created without adding new disks.

In the initial set up, the logical volume mylv is carved from the volume group myvol, which in turn consists of the three physical volumes, /dev/sda1, /dev/sdb1, and /dev/sdc1.

After completing this procedure, the volume group myvg will consist of /dev/sdal and /dev/sdb1. A second volume group, yourvg, will consist of /dev/sdc1.

### 3.1. Determining Free Space

You can use the `pvscan` command to determine how much free space is currently available in the volume group.

```
[root@tng3-1 ~]# pvscan
  PV /dev/sda1 VG myvg lvm2 [17.15 GB / 0 free]
  PV /dev/sdb1 VG myvg lvm2 [17.15 GB / 12.15 GB free]
  PV /dev/sdc1 VG myvg lvm2 [17.15 GB / 15.80 GB free]
  Total: 3 [51.45 GB] / in use: 3 [51.45 GB] / in no VG: 0 [0]
```

### 3.2. Moving the Data

You can move all the used physical extents in /dev/sdc1 to /dev/sdb1 with the `pvmove` command. The `pvmove` command can take a long time to execute.

```
[root@tng3-1 ~]# pvmove /dev/sdc1 /dev/sdb1
/dev/sdc1: Moved: 14.7%
/dev/sdc1: Moved: 30.3%
/dev/sdc1: Moved: 45.7%
/dev/sdc1: Moved: 61.0%
/dev/sdc1: Moved: 76.6%
/dev/sdc1: Moved: 92.2%
/dev/sdc1: Moved: 100.0%
```

After moving the data, you can see that all of the space on /dev/sdc1 is free.

```
[root@tng3-1 ~]# pvscan
  PV /dev/sda1 VG myvg lvm2 [17.15 GB / 0 free]
  PV /dev/sdb1 VG myvg lvm2 [17.15 GB / 10.80 GB free]
  PV /dev/sdc1 VG myvg lvm2 [17.15 GB / 17.15 GB free]
  Total: 3 [51.45 GB] / in use: 3 [51.45 GB] / in no VG: 0 [0]
```

### 3.3. Splitting the Volume Group

To create the new volume group yourvg, use the `vgsplit` command to split the volume group myvg.

Before you can split the volume group, the logical volume must be inactive. If the file system is mounted, you must unmount the file system before deactivating the logical volume.

You can deactivate the logical volumes with the `lvchange` command or the `vgchange` command. The following command deactivates the logical volume mylv and then splits the volume group yourvg from the volume group myvg, moving the physical volume /dev/sdc1 into the new volume group.
group yourvg.

[root@tng3-1 ~]# lvchange -a n /dev/myvg/mylv
[root@tng3-1 ~]# vgsplit myvg yourvg /dev/sdc1
  Volume group "yourvg" successfully split from "myvg"

You can use the vgs command to see the attributes of the two volume groups.

[root@tng3-1 ~]# vgs
  VG   #PV #LV #SN Attr  VSize   VFree
  myvg 2 1 0 wz--n- 34.30G 10.80G
  yourvg 1 0 0 wz--n- 17.15G 17.15G

3.4. Creating the New Logical Volume

After creating the new volume group, you can create the new logical volume yourlv.

[root@tng3-1 ~]# lvcreate -L5G -n yourlv yourvg
  Logical volume "yourlv" created

3.5. Making a File System and Mounting the New Logical Volume

You can make a file system on the new logical volume and mount it.

[root@tng3-1 ~]# gfs_mkfs -plock_nolock -j 1 /dev/yourvg/yourlv
  This will destroy any data on /dev/yourvg/yourlv.
  Are you sure you want to proceed? [y/n] y

  Device: /dev/yourvg/yourlv
  Blocksize: 4096
  Filesystem Size: 1277816
  Journals: 1
  Resource Groups: 20
  Locking Protocol: lock_nolock
  Lock Table: Syncing...
  All Done

[root@tng3-1 ~]# mount /dev/yourvg/yourlv /mnt

3.6. Activating and Mounting the Original Logical Volume

Since you had to deactivate the logical volume mylv, you need to activate it again before you can mount it.

[root@tng3-1 ~]# lvchange -a y mylv

[root@tng3-1 ~]# mount /dev/myvg/mylv /mnt

[root@tng3-1 ~]# df
  Filesystem  1K-blocks  Used  Available  Use% Mounted on
  /dev/yourvg/yourlv  24507776   32  24507744   1% /mnt
  /dev/myvg/mylv  24507776   32  24507744   1% /mnt
4. Removing a Disk from a Logical Volume

This example shows how you can remove a disk from an existing logical volume, either to replace the disk or to use the disk as part of a different volume. In order to remove a disk, you must first move the extents on the LVM physical volume to a different disk or set of disks.

4.1. Moving Extents to Existing Physical Volumes

In this example, the logical volume is distributed across four physical volumes in the volume group `myvg`.

```
[root@tng3-1]# pvs -o+pv_used
PV      VG    Fmt  Attr  PSize  PFree  Used
/dev/sda1 myvg lvm2 a-  17.15G  12.15G  5.00G
/dev/sdb1 myvg lvm2 a-  17.15G  12.15G  5.00G
/dev/sdc1 myvg lvm2 a-  17.15G  12.15G  5.00G
/dev/sdd1 myvg lvm2 a-  17.15G  2.15G  15.00G
```

We want to move the extents off of `/dev/sdb1` so that we can remove it from the volume group.

If there are enough free extents on the other physical volumes in the volume group, you can execute the `pvmove` command on the device you want to remove with no other options and the extents will be distributed to the other devices.

```
[root@tng3-1 ~]# pvmove /dev/sdb1
/dev/sdb1: Moved: 2.0%
...
/dev/sdb1: Moved: 79.2%
...
/dev/sdb1: Moved: 100.0%
```

After the `pvmove` command has finished executing, the distribution of extents is as follows:

```
[root@tng3-1]# pvs -o+pv_used
PV      VG    Fmt  Attr  PSize  PFree  Used
/dev/sda1 myvg lvm2 a-  17.15G  7.15G  10.00G
/dev/sdb1 myvg lvm2 a-  17.15G  17.15G   0
/dev/sdc1 myvg lvm2 a-  17.15G  12.15G  5.00G
/dev/sdd1 myvg lvm2 a-  17.15G  2.15G  15.00G
```

Use the `vgreduce` command to remove the physical volume `/dev/sdb1` from the volume group.

```
[root@tng3-1 ~]# vgreduce myvg /dev/sdb1
Removed "/dev/sdb1" from volume group "myvg"
[root@tng3-1 ~]# pvs
PV      VG    Fmt  Attr  PSize  PFree
/dev/sda1 myvg lvm2 a-  17.15G  7.15G
/dev/sdb1  lvm2   --  17.15G  17.15G
/dev/sdc1 myvg lvm2 a-  17.15G  12.15G
/dev/sdd1 myvg lvm2 a-  17.15G  2.15G
```

The disk can now be physically removed or allocated to other users.
4.2. Moving Extents to a New Disk

In this example, the logical volume is distributed across three physical volumes in the volume group `myvg` as follows:

```
[root@tng3-1]# pvs -o+pv_used
PV          VG  Fmt  Attr  PSize  PFree  Used
/dev/sda1   myvg  lvm2  a-     17.15G  7.15G  10.00G
/dev/sdb1   myvg  lvm2  a-     17.15G 15.15G  2.00G
/dev/sdc1   myvg  lvm2  a-     17.15G 15.15G  2.00G
```

We want to move the extents of `/dev/sdb1` to a new device, `/dev/sdd1`.

4.2.1. Creating the New Physical Volume

Create a new physical volume from `/dev/sdd1`.

```
[root@tng3-1 ~]# pvcreate /dev/sdd1
Physical volume "/dev/sdd1" successfully created
```

4.2.2. Adding the New Physical Volume to the Volume Group

Add `/dev/sdd1` to the existing volume group `myvg`.

```
[root@tng3-1 ~]# vgextend myvg /dev/sdd1
Volume group "myvg" successfully extended
```

4.2.3. Moving the Data

Use the `pvmove` command to move the data from `/dev/sdb1` to `/dev/sdd1`.

```
[root@tng3-1 ~]# pvmove /dev/sdb1 /dev/sdd1
/dev/sdb1: Moved: 10.0%
...  
/dev/sdb1: Moved: 79.7%
...  
/dev/sdb1: Moved: 100.0%
```

```
[root@tng3-1]# pvs -o+pv_used
PV          VG  Fmt  Attr  PSize  PFree  Used
/dev/sda1   myvg  lvm2  a-     17.15G  7.15G  10.00G
/dev/sdb1   myvg  lvm2  a-     17.15G 17.15G  0
/dev/sdc1   myvg  lvm2  a-     17.15G 15.15G  2.00G
```

4.2.4. Removing the Old Physical Volume from the Volume Group

After you have moved the data off `/dev/sdb1`, you can remove it from the volume group.
4.2. Moving Extents to a New Disk

[root@tng3-1 ~]# vgreduce myvg /dev/sdb1
  Removed "/dev/sdb1" from volume group "myvg"

You can now reallocate the disk to another volume group or remove the disk from the system.
Chapter 6. LVM Troubleshooting

This chapter provides instructions for troubleshooting a variety of LVM issues.

1. Troubleshooting Diagnostics

If a command is not working as expected, you can gather diagnostics in the following ways:

- Use the `-v`, `-vv`, `-vvv`, or `-vvvv` argument of any command for increasingly verbose levels of output.

- If the problem is related to the logical volume activation, set `activation = 1` in the `log` section of the configuration file and run the command with the `-vvvv` argument. After you have finished examining this output be sure to reset this parameter to 0, to avoid possible problems with the machine locking during low memory situations.

- Run the `lvmdump` command, which provides information dump for diagnostic purposes. For information, see the `lvmdump(8)` man page.

- Execute the `lvs -v`, `pvs -a`, or `dmsetup info -c` command for additional system information.

- Examine the last backup of the metadata in the `/etc/lvm/backup` file and archived versions in the `/etc/lvm/archive` file.

- Check the current configuration information by running the `lvm dumpconfig` command.

- Check the `.cache` file in the `/etc/lvm` directory for a record of which devices have physical volumes on them.

2. Displaying Information on Failed Devices

You can use the `-P` argument of the `lvs` or `vgs` command to display information about a failed volume that would otherwise not appear in the output. This argument permits some operations even though the metadata is not completely consistent internally. For example, if one of the devices that made up the volume group `vg` failed, the `vgs` command might show the following output.

```
[root@link-07 tmp]# vgs -o +devices
Volume group "vg" not found
```

If you specify the `-P` argument of the `vgs` command, the volume group is still unusable but you can see more information about the failed device.

```
[root@link-07 tmp]# vgs -P -o +devices
Partial mode. Incomplete volume groups will be activated read-only.
VG  #PV  #LV  #SN  Attr  VSize  VFree  Devices
vg  9  2  0  rz-pn-  2.11T  2.07T  unknown device(0)
vg  9  2  0  rz-pn-  2.11T  2.07T  unknown device(5120),/dev/sda1(0)
```
In this example, the failed device caused both a linear and a striped logical volume in the volume group to fail. The `lvs` command without the `-P` argument shows the following output.

```
[root@link-07 tmp]# lvs -a -o +devices
  Volume group "vg" not found
```

Using the `-P` argument shows the logical volumes that have failed.

```
[root@link-07 tmp]# lvs -P -a -o +devices
  Partial mode. Incomplete volume groups will be activated read-only.
  LV    VG    Attr LSize Origin Snap% Move Log Copy% Devices
  linear vg    -wi-a-  20.00G unknown device(0)
  stripe vg    -wi-a-  20.00G unknown device(5120),/dev/sda1(0)
```

The following examples show the output of the `pvs` and `lvs` commands with the `-P` argument specified when a leg of a mirrored logical volume has failed.

```
root@link-08 ~]# vgs -a -o +devices -P
  Partial mode. Incomplete volume groups will be activated read-only.
  VG    #PV #LV #SN Attr  VSize  VFree Devices
  corey 4 4 0 rz-pnc 1.58T 1.34T my_mirror_mimage_0(0),my_mirror_mimage_1(0)
  corey 4 4 0 rz-pnc 1.58T 1.34T /dev/sdb1(0)
  corey 4 4 0 rz-pnc 1.58T 1.34T unknown device(0)
  corey 4 4 0 rz-pnc 1.58T 1.34T /dev/sdd1(0)
```

```
[root@link-08 ~]# lvs -a -o +devices -P
  Partial mode. Incomplete volume groups will be activated read-only.
  LV    VG    Attr LSize Origin Snap% Move Log Copy% Devices
  my_mirror corey mwi-a- 120.00G my_mirror_mlog 1.95 my_mirror_mimage_0(0),unknown device(0)
  [my_mirror_mimage_0] corey iwi-ao 120.00G unknown device(0)
  [my_mirror_mimage_1] corey iwi-ao 120.00G /dev/sdb1(0)
  [my_mirror_mlog]    corey lwi-ao  4.00M /dev/sdd1(0)
```

### 3. Recovering from LVM Mirror Failure

This section provides an example of recovering from a situation where one leg of an LVM mirrored volume fails because the underlying device for a physical volume goes down. When a mirror leg fails, LVM converts the mirrored volume into a linear volume, which continues to operate as before but without the mirrored redundancy. At that point, you can add a new disk device to the system to use as a replacement physical device and rebuild the mirror.

The following command creates the physical volumes which will be used for the mirror.

```
[root@link-08 ~]# pvcreate /dev/sd[abcdefgh][12]
  Physical volume "/dev/sda1" successfully created
  Physical volume "/dev/sda2" successfully created
  Physical volume "/dev/sdb1" successfully created
  Physical volume "/dev/sdb2" successfully created
  Physical volume "/dev/sdc1" successfully created
  Physical volume "/dev/sdc2" successfully created
  Physical volume "/dev/sdd1" successfully created
  Physical volume "/dev/sdd2" successfully created
  Physical volume "/dev/sde1" successfully created
  Physical volume "/dev/sde2" successfully created
  Physical volume "/dev/sdf1" successfully created
```
Physical volume "/dev/sdf2" successfully created
Physical volume "/dev/sdg1" successfully created
Physical volume "/dev/sdg2" successfully created
Physical volume "/dev/sdh1" successfully created
Physical volume "/dev/sdh2" successfully created

The following commands creates the volume group `vg` and the mirrored volume `groupfs`.

```
[root@link-08 ~]# vgcreate vg /dev/sd[abcdefgh]12
Volume group "vg" successfully created
[root@link-08 ~]# lvcreate -L 750M -n groupfs -m 1 vg /dev/sda1 /dev/sdb1 /dev/sdc1
Rounding up size to full physical extent 752.00 MB
Logical volume "groupfs" created
```

You can use the `lvs` command to verify the layout of the mirrored volume and the underlying devices for the mirror leg and the mirror log. Note that in the first example the mirror is not yet completely synced; you should wait until the Copy% field displays 100.00 before continuing.

```
[root@link-08 ~]# lvs -a -o +devices
LV VG Attr LSize Origin Snap% Move Log Copy% Devices
  groupfs vg mwi-a- 752.00M groupfs_mlog 21.28 groupfs_mimage_0(0),groupfs_mimage_1(0)
[groupfs_mimage_0] vg iwi-ao 752.00M /dev/sda1(0)
[groupfs_mimage_1] vg iwi-ao 752.00M /dev/sdb1(0)
[groupfs_mlog]  vg lwi-ao 4.00M /dev/sdc1(0)
[root@link-08 ~]# lvs -a -o +devices
LV VG Attr LSize Origin Snap% Move Log Copy% Devices
  groupfs vg mwi-a- 752.00M groupfs_mlog 100.00 groupfs_mimage_0(0),groupfs_mimage_1(0)
[groupfs_mimage_0] vg iwi-ao 752.00M /dev/sda1(0)
[groupfs_mimage_1] vg iwi-ao 752.00M /dev/sdb1(0)
[groupfs_mlog]  vg lwi-ao 4.00M i /dev/sdc1(0)
```

In this example, the primary leg of the mirror /dev/sda1 fails. Any write activity to the mirrored volume causes LVM to detect the failed mirror. When this occurs, LVM converts the mirror into a single linear volume. In this case, to trigger the conversion, we execute a `dd` command.

```
[root@link-08 ~]# dd if=/dev/zero of=/dev/vg/groupfs count=10
10+0 records in
10+0 records out
```

You can use the `lvs` command to verify that the device is now a linear device. Because of the failed disk, I/O errors occur.

```
[root@link-08 ~]# lvs -a -o +devices
/dev/sda1: read failed after 0 of 2048 at 0: Input/output error
/dev/sda2: read failed after 0 of 2048 at 0: Input/output error
LV VG Attr LSize Origin Snap% Move Log Copy% Devices
  groupfs vg -wi-a- 752.00M /dev/sdb1(0)
```

At this point you should still be able to use the logical volume, but there will be no mirror redundancy.

To rebuild the mirrored volume, you replace the broken drive and recreate the physical volume. If you use the same disk rather than replacing it with a new one, you will see "inconsistent" warnings when you run the `pvcreate` command.
4. Recovering Physical Volume Metadata

```
[root@link-08 ~]# pvcreate /dev/sda[12]
Physical volume "/dev/sda1" successfully created
Physical volume "/dev/sda2" successfully created

[root@link-08 ~]# pvscan
PV /dev/sdb1 VG vg lvm2 [67.83 GB / 67.10 GB free]
PV /dev/sdb2 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sdc1 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sdc2 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sdd1 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sdd2 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sde1 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sde2 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sdf1 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sdf2 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sdg1 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sdg2 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sdh1 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sdh2 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sda1 VG vg lvm2 [603.94 GB]
PV /dev/sda2 VG vg lvm2 [603.94 GB]

Next you extend the original volume group with the new physical volume.

[root@link-08 ~]# vgextend vg /dev/sda[12]
Volume group "vg" successfully extended

[root@link-08 ~]# pvscan
PV /dev/sdb1 VG vg lvm2 [67.83 GB / 67.10 GB free]
PV /dev/sdb2 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sdc1 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sdc2 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sdd1 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sdd2 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sde1 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sde2 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sdf1 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sdf2 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sdg1 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sdg2 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sdh1 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sdh2 VG vg lvm2 [67.83 GB / 67.83 GB free]
PV /dev/sda1 VG vg lvm2 [603.93 GB / 603.93 GB free]
PV /dev/sda2 VG vg lvm2 [603.93 GB / 603.93 GB free]
Total: 16 [2.11 TB] / in use: 16 [2.11 TB] / in no VG: 0 [0 ]

Convert the linear volume back to its original mirrored state.

[root@link-08 ~]# lvconvert -m 1 /dev/vg/groupfs /dev/sda1 /dev/sdb1 /dev/sdc1
Logical volume mirror converted.

You can use the lvs command to verify that the mirror is restored.

[root@link-08 ~]# lvs -a -o +devices
LV VG Attr LSize Origin Snap% Move Log Copy% Devices
--- --- ------ ------ ------ ------ ------ ------ ------
groupfs vg mwi-a- 752.00M
[groupfs_mimage_0] vg iwi-ao 752.00M
[groupfs_mimage_1] vg iwi-ao 752.00M
[groupfs_mlog] vg lwi-ao 4.00M

62
4. Recovering Physical Volume Metadata

If the volume group metadata area of a physical volume is accidentally overwritten or otherwise destroyed, you will get an error message indicating that the metadata area is incorrect, or that the system was unable to find a physical volume with a particular UUID. You may be able to recover the data the physical volume by writing a new metadata area on the physical volume specifying the same UUID as the lost metadata.

**Caution**

You should not attempt this procedure with a working LVM logical volume. You will lose your data if you specify the incorrect UUID.

The following example shows the sort of output you may see if the metadata area is missing or corrupted.

```
[root@link-07 backup]# lvs -a -o +devices
  Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
  Couldn't find all physical volumes for volume group VG.
  Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
  Couldn't find all physical volumes for volume group VG.
  ...
```

You may be able to find the UUID for the physical volume that was overwritten by looking in the `/etc/lvm/archive` directory. Look in the file `VolumeGroupName_xxxx.vg` for the last known valid archived LVM metadata for that volume group.

Alternately, you may find that deactivating the volume and setting the `partial (-P)` argument will enable you to find the UUID of the missing corrupted physical volume.

```
[root@link-07 backup]# vgchange -an --partial
  Partial mode. Incomplete volume groups will be activated read-only.
  Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
  Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
  ...
```

Use the `--uuid` and `--restorefile` arguments of the `pvcreate` command to restore the physical volume. The following example labels the `/dev/sdh1` device as a physical volume with the UUID indicated above, `FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk`. This command restores the physical volume label with the metadata information contained in `VG_00050.vg`, the most recent good archived metadata for volume group. The `restorefile` argument instructs the `pvcreate` command to make the new physical volume compatible with the old one on the volume group, ensuring that the the new metadata will not be placed where the old physical volume contained data (which could happen, for example, if the original `pvcreate` command had used the command line arguments that control metadata placement, or if the physical volume was originally created using a different version of the software that used different defaults). The `pvcreate` command overwrites only the LVM metadata areas and does not affect the existing data areas.

```
[root@link-07 backup]# pvcreate --uuid "FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk" --restorefile /etc/lvm/archive
```
5. Replacing a Missing Physical Volume

You can then use the `vgcfgrestore` command to restore the volume group's metadata.

```
[root@link-07 backup]# vgcfgrestore VG
Restored volume group VG
```

You can now display the logical volumes.

```
[root@link-07 backup]# lvs -a -o +devices
LV  VG  Attr  LSize  Origin  Snap%  Move  Log  Copy%  Devices
strip VG  -wi---  300.00G  /dev/sdh1 (0),/dev/sda1(0)
strip VG  -wi---  300.00G  /dev/sdh1 (34728),/dev/sdb1(0)
```

The following commands activate the volumes and display the active volumes.

```
[root@link-07 backup]# lvchange -ay /dev/VG/stripe
[root@link-07 backup]# lvs -a -o +devices
LV  VG  Attr  LSize  Origin  Snap%  Move  Log  Copy%  Devices
strip VG  -wi-a-  300.00G  /dev/sdh1 (0),/dev/sda1(0)
strip VG  -wi-a-  300.00G  /dev/sdh1 (34728),/dev/sdb1(0)
```

If the on-disk LVM metadata takes at least as much space as what overrode it, this command can recover the physical volume. If what overrode the metadata went past the metadata area, the data on the volume may have been affected. You might be able to use the `fsck` command to recover that data.

5. Replacing a Missing Physical Volume

If a physical volume fails or otherwise needs to be replaced, you can label a new physical volume to replace the one that has been lost in the existing volume group by following the same procedure as you would for recovering physical volume metadata, described in Section 4, “Recovering Physical Volume Metadata”. You can use the `--partial` and `--verbose` arguments of the `vgdisplay` command to display the UUIDs and sizes of any physical volumes that are no longer present. If you wish to substitute another physical volume of the same size, you can use the `pvcreate` command with the `--restorefile` and `--uuid` arguments to initialize a new device with the same UUID as the missing physical volume. You can then use the `vgcfgrestore` command to restore the volume group's metadata.

6. Removing Lost Physical Volumes from a Volume Group

If you lose a physical volume, you can activate the remaining physical volumes in the volume group with the `--partial` argument of the `vgchange` command. You can remove all the logical volumes that used that physical volume from the volume group with the `--removemissing` argument of the `vgreduce` command.

It is recommended that you run the `vgreduce` command with the `--test` argument to verify what you will be destroying.
Like most LVM operations, the `vgreduce` command is reversible in a sense if you immediately use the `vgcfgrestore` command to restore the volume group metadata to its previous state. For example, if you used the `--removemissing` argument of the `vgreduce` command without the `--test` argument and find you have removed logical volumes you wanted to keep, you can still replace the physical volume and use another `vgcfgrestore` command to return the volume group to its previous state.

### 7. Insufficient Free Extents for a Logical Volume

You may get the error message "Insufficient free extents" when creating a logical volume when you think you have enough extents based on the output of the `vgdisplay` or `vgs` commands. This is because these commands round figures to 2 decimal places to provide human-readable output. To specify exact size, use free physical extent count instead of some multiple of bytes to determine the size of the logical volume.

The `vgdisplay` command, by default, includes this line of output that indicates the free physical extents.

```bash
# vgdisplay
--- Volume group ---
...
Free PE / Size 8780 / 34.30 GB
```

Alternately, you can use the `vg_free_count` and `vg_extent_count` arguments of the `vgs` command to display the free extents and the total number of extents.

```bash
[root@tng3-1 ~]# vgs -o +vg_free_count,vg_extent_count
VG #PV #LV #SN Attr VSize VFree Free Ext
  testvg 2 0 0 wz--n- 34.30G 0 0 8780
```

With 8780 free physical extents, you can run the following command, using the lower-case `l` argument to use extents instead of bytes:

```bash
# lvcreate -l8780 -n testlv testvg
```

This uses all the free extents in the volume group.

```bash
[root@tng3-1 ~]# vgs -o +vg_free_count,vg_extent_count
VG #PV #LV #SN Attr  VSize  VFree  Free  Ext
  testvg 2 1 0 wz--n- 34.30G 0 0 8780
```

Alternately, you can extend the logical volume to use a percentage of the remaining free space in the volume group by using the `-l` argument of the `lvcreate` command. For information, see Section 4.1.1, “Creating Linear Volumes”.

---

65
Chapter 7. LVM Administration with the LVM GUI

In addition to the Command Line Interface (CLI), LVM provides a Graphical User Interface (GUI) which you can use to configure LVM logical volumes. You can bring up this utility by typing `system-config-lvm`. The LVM chapter of the *Red Hat Enterprise Linux Deployment Guide* provides step-by-step instructions for configuring an LVM logical volume using this utility.

In addition, the LVM GUI is available as part of the Conga management interface. For information on using the LVM GUI with Conga, see the online help for Conga.
Appendix A. The Device Mapper

The Device Mapper is a kernel driver that provides a generic framework for volume management. It provides a generic way of creating mapped devices, which may be used as logical volumes. It does not specifically know about volume groups or metadata formats.

The Device Mapper provides the foundation for a number of higher-level technologies. In addition to LVM, device-mapper multipath and the dmsraid command use the Device Mapper.

The user interface to the Device Mapper is the ioctl system call.

LVM logical volumes are activated using the Device Mapper. Each logical volume is translated into a mapped device, each segment translates into a line in the mapping table that describes the device. The Device Mapper provides linear mapping, striped mapping, and error mapping, amongst others. Two disks can be concatenated into one logical volume with a pair of linear mappings, one for each disk.

The dmsetup command is a command line wrapper for communication with the Device Mapper. It provides complete access to the ioctl commands through the libdevmapper command. For general system information about LVM devices, you may find the dmsetup info command to be useful.

For information about the options and capabilities of the dmsetup command, see the dmsetup(8) man page.
Appendix B. The LVM Configuration Files

LVM supports multiple configuration files. At system startup, the \texttt{lvm.conf} configuration file is loaded from the directory specified by the environment variable \texttt{LVM\_SYSTEM\_DIR}, which is set to /etc/lvm by default.

The \texttt{lvm.conf} file can specify additional configuration files to load. Settings in later files override settings from earlier ones. To display the settings in use after loading all the configuration files, execute the \texttt{lvm dumpconfig} command.

For information on loading additional configuration files, see Section 2, “Host Tags”.

1. The LVM Configuration Files

The following files are used for LVM configuration:

\texttt{/etc/lvm/lvm.conf}  
Central configuration file read by the tools.

\texttt{etc/lvm/lvm\_hosttag.conf}  
For each host tag, an extra configuration file is read if it exists: \texttt{lvm\_hosttag.conf}. If that file defines new tags, then further configuration files will be appended to the list of tiles to read in. For information on host tags, see Section 2, “Host Tags”.

In addition to the LVM configuration files, a system running LVM includes the following files that affect LVM system setup:

\texttt{/etc/lvm/.cache}  
Device name filter cache file (configurable).

\texttt{/etc/lvm/backup/}  
Directory for automatic volume group metadata backups (configurable).

\texttt{/etc/lvm/archive/}  
Directory for automatic volume group metadata archives (configurable with regard to directory path and archive history depth).

\texttt{/var/lock/lvm/}  
In single-host configuration, lock files to prevent parallel tool runs from corrupting the metadata; in a cluster, cluster-wide DLM is used.

2. Sample \texttt{lvm.conf} File

The following is a sample \texttt{lvm.conf} configuration file.

\begin{verbatim}
[root@tng3-1 lvm]# cat lvm.conf
\end{verbatim}
# This is an example configuration file for the LVM2 system.
# It contains the default settings that would be used if there was no
# /etc/lvm/lvm.conf file.
# Refer to 'man lvm.conf' for further information including the file layout.
# To put this file in a different directory and override /etc/lvm set
# the environment variable LVM_SYSTEM_DIR before running the tools.

This section allows you to configure which block devices should
be used by the LVM system.

devices {

    # Where do you want your volume groups to appear ?
    dir = "/dev"

    # An array of directories that contain the device nodes you wish
    # to use with LVM2.
    scan = [ "/dev" ]

    # A filter that tells LVM2 to only use a restricted set of devices.
    # The filter consists of an array of regular expressions. These
    # expressions can be delimited by a character of your choice, and
    # prefixed with either an 'a' (for accept) or 'r' (for reject).
    # The first expression found to match a device name determines if
    # the device will be accepted or rejected (ignored). Devices that
    # don't match any patterns are accepted.

    # Be careful if there there are symbolic links or multiple filesystem
    # entries for the same device as each name is checked separately against
    # the list of patterns. The effect is that if any name matches any 'a'
    # pattern, the device is accepted; otherwise if any name matches any 'r'
    # pattern it is rejected; otherwise it is accepted.

    # Don't have more than one filter line active at once: only one gets used.

    # Run vgscan after you change this parameter to ensure that
    # the cache file gets regenerated (see below).
    # If it doesn't do what you expect, check the output of 'vgscan -vvv'.

    # By default we accept every block device:
    # Steel Toe installed filter -- use this line while running tests
    # filter = [ "r/hda/", "r/disk/", "a/.*" ]
    # Steel Toe installed filter -- use this line when installing new kernels
    # filter = [ "r/disk/", "a/.*" ]

    # Exclude the cdrom drive
    # filter = [ "r[/dev/cdrom|" ]

    # When testing I like to work with just loopback devices:
    # filter = [ "a/loop/", "r/.*" ]

    # Or maybe all loops and ide drives except hdc:
    # filter = [ "a|loop|", "r[/dev/hdc|", "a[/dev/ide|", "r|.*|" ]

    # Use anchors if you want to be really specific
    # filter = [ "a/[^/dev/hda8$|", "r/.*" ]

    # The results of the filtering are cached on disk to avoid
    # rescanning dud devices (which can take a very long time). By
    # default this cache file is hidden in the /etc/lvm directory.
    # It is safe to delete this file: the tools regenerate it.
    cache = "/etc/lvm/.cache"

    # You can turn off writing this cache file by setting this to 0.
    write_cache_state = 1
# Advanced settings.

# List of pairs of additional acceptable block device types found in /proc/devices with maximum (non-zero) number of partitions.
# types = [ "fd", 16 ]

# If sysfs is mounted (2.6 kernels) restrict device scanning to the block devices it believes are valid.
# 1 enables; 0 disables.
sysfs_scan = 1

# By default, LVM2 will ignore devices used as components of software RAID (md) devices by looking for md superblocks.
# 1 enables; 0 disables.
md_component_detection = 1

# This section that allows you to configure the nature of the information that LVM2 reports.
log {
  
  # Controls the messages sent to stdout or stderr.
  # There are three levels of verbosity, 3 being the most verbose.
  # 1 is yes; 0 is no.
  verbose = 0

  # Should we send log messages through syslog?
  syslog = 1

  # Should we log error and debug messages to a file?
  # By default there is no log file.
  #file = "/var/log/lvm2.log"
  # Overwrite the log file each time the program is run?
  # By default we append.
  overwrite = 0

  # What level of log messages should we send to the log file and/or syslog?
  # There are 6 syslog-like log levels currently in use - 2 to 7 inclusive.
  # 7 is the most verbose (LOG_DEBUG).
  level = 0

  # Format of output messages
  # Whether or not (1 or 0) to indent messages according to their severity
  indent = 1

  # Whether or not (1 or 0) to display the command name on each line output
  command_names = 0

  # A prefix to use before the message text (but after the command name, if selected).
  # Default is two spaces, so you can see/grep the severity of each message.
  prefix = " 

  # To make the messages look similar to the original LVM tools use:
  indent = 0
  command_names = 1
  prefix = " -- "

  # Set this if you want log messages during activation.
  # Don't use this in low memory situations (can deadlock).
  #activation = 0
}

# Configuration of metadata backups and archiving. In LVM2 when we talk about a 'backup' we mean making a copy of the metadata for the "current" system. The 'archive' contains old metadata configurations.
# Backups are stored in a human readable text format.
backup {
  # Should we maintain a backup of the current metadata configuration?
  # Use 1 for Yes; 0 for No.
  # Think very hard before turning this off!
  backup = 1

  # Where shall we keep it?
  # Remember to back up this directory regularly!
  backup_dir = "/etc/lvm/backup"

  # Should we maintain an archive of old metadata configurations.
  # Use 1 for Yes; 0 for No.
  # On by default. Think very hard before turning this off.
  archive = 1

  # Where should archived files go?
  # Remember to back up this directory regularly!
  archive_dir = "/etc/lvm/archive"

  # What is the minimum number of archive files you wish to keep?
  retain_min = 10

  # What is the minimum time you wish to keep an archive file for?
  retain_days = 30
}

# Settings for the running LVM2 in shell (readline) mode.
shell {
  # Number of lines of history to store in ~/.lvm_history
  history_size = 100
}

# Miscellaneous global LVM2 settings
global {
  library_dir = "/usr/lib"

  # The file creation mask for any files and directories created.
  # Interpreted as octal if the first digit is zero.
  umask = 077

  # Allow other users to read the files
  #umask = 022

  # Enabling test mode means that no changes to the on disk metadata
  # will be made. Equivalent to having the -t option on every
  # command. Defaults to off.
  test = 0

  # Whether or not to communicate with the kernel device-mapper.
  # Set to 0 if you want to use the tools to manipulate LVM metadata
  # without activating any logical volumes.
  # If the device-mapper kernel driver is not present in your kernel
  # setting this to 0 should suppress the error messages.
  activation = 1

  # If we can't communicate with device-mapper, should we try running
  # the LVM1 tools?
  # This option only applies to 2.4 kernels and is provided to help you
  # switch between device-mapper kernels and LVM1 kernels.
  # The LVM1 tools need to be installed with .lvm1 suffixes
  # e.g. vgscan.lvm1 and they will stop working after you start using
  # the new lvm2 on-disk metadata format.
  # The default value is set when the tools are built.
  fallback_to_lvm1 = 0
# The default metadata format that commands should use - "lvm1" or "lvm2".
# The command line override is -M1 or -M2.
# Defaults to "lvm1" if compiled in, else "lvm2".
# format = "lvm1"

# Location of proc filesystem
proc = "/proc"

# Type of locking to use. Defaults to local file-based locking (1).
# Turn locking off by setting to 0 (dangerous: risks metadata corruption
# if LVM2 commands get run concurrently).
# Type 2 uses the external shared library locking_library.
# Type 3 uses built-in clustered locking.
locking_type = 1

# If using external locking (type 2) and initialisation fails,
# with this set to 1 an attempt will be made to use the built-in
# clustered locking.
# If you are using a customised locking_library you should set this to 0.
fallback_to_clustered_locking = 1

# If an attempt to initialise type 2 or type 3 locking failed, perhaps
# because cluster components such as clvmd are not running, with this set
# to 1 an attempt will be made to use local file-based locking (type 1).
# If this succeeds, only commands against local volume groups will proceed.
# Volume Groups marked as clustered will be ignored.
fallback_to_local_locking = 1

# Local non-LV directory that holds file-based locks while commands are
# in progress. A directory like /tmp that may get wiped on reboot is OK.
locking_dir = "/var/lock/lvm"

# Other entries can go here to allow you to load shared libraries
# e.g. if support for LVM1 metadata was compiled as a shared library use
# format_libraries = "liblvm2format1.so"
# Full pathnames can be given.

# Search this directory first for shared libraries.
# library_dir = "/lib"

# The external locking library to load if locking_type is set to 2.
# locking_library = "liblvm2clusterlock.so"

activation {
    # Device used in place of missing stripes if activating incomplete volume.
    # For now, you need to set this up yourself first (e.g. with 'dmsetup')
    # For example, you could make it return I/O errors using the 'error'
    # target or make it return zeros.
    missing_stripe_filler = "/dev/ioerror"

    # How much stack (in KB) to reserve for use while devices suspended
    reserved_stack = 256

    # How much memory (in KB) to reserve for use while devices suspended
    reserved_memory = 8192

    # Nice value used while devices suspended
    process_priority = -18

    # If volume_list is defined, each LV is only activated if there is a
    # match against the list.
    # "vgname" and "vgname/lvname" are matched exactly.
    # "@tag" matches any tag set in the LV or VG.
    # "@*" matches if any tag defined on the host is also set in the LV or VG
    # volume_list = [ "vg1", "vg2/lvoll", "@tag1", "@*" ]
# Size (in KB) of each copy operation when mirroring
mirror_region_size = 512

# 'mirror_image_fault_policy' and 'mirror_log_fault_policy' define
# how a device failure affecting a mirror is handled.
# A mirror is composed of mirror images (copies) and a log.
# A disk log ensures that a mirror does not need to be re-synced
# (all copies made the same) every time a machine reboots or crashes.
# In the event of a failure, the specified policy will be used to
# determine what happens:
#
# "remove" - Simply remove the faulty device and run without it. If
# the log device fails, the mirror would convert to using
# an in-memory log. This means the mirror will not
# remember its sync status across crashes/reboots and
# the entire mirror will be re-synced. If a
# mirror image fails, the mirror will convert to a
# non-mirrored device if there is only one remaining good
# copy.
#
# "allocate" - Remove the faulty device and try to allocate space on
# a new device to be a replacement for the failed device.
# Using this policy for the log is fast and maintains the
# ability to remember sync state through crashes/reboots.
# Using this policy for a mirror device is slow, as it
# requires the mirror to resynchronize the devices, but it
# will preserve the mirror characteristic of the device.
# This policy acts like "remove" if no suitable device and
# space can be allocated for the replacement.
# Currently this is not implemented properly and behaves
# similarly to:
#
# "allocate_anywhere" - Operates like "allocate", but it does not
# require that the new space being allocated be on a
# device that is not part of the mirror. For a log device
# failure, this could mean that the log is allocated on
# the same device as a mirror device. For a mirror
# device, this could mean that the mirror device is
# allocated on the same device as another mirror device.
# This policy acts like "remove" if no suitable
# device and space can be allocated for the replacement.

mirror_log_fault_policy = "allocate"
mirror_device_fault_policy = "remove"

# Metadata settings
# metadata {
# Default number of copies of metadata to hold on each PV. 0, 1 or 2.
# You might want to override it from the command line with 0
# when running pvcreate on new PVs which are to be added to large VGs.

# pvmetadaticopies = 1

# Approximate default size of on-disk metadata areas in sectors.
# You should increase this if you have large volume groups or
# you want to retain a large on-disk history of your metadata changes.
# pvmetadatasize = 255

# List of directories holding live copies of text format metadata.
# These directories must not be on logical volumes!
# It's possible to use LVM2 with a couple of directories here,
# preferably on different (non-LV) filesystems, and with no other
# on-disk metadata (pvmetadatacopies = 0). Or this can be in
# addition to on-disk metadata area.
# The feature was originally added to simplify testing and is not
# supported under low memory situations - the machine could lock up.
# Never edit any files in these directories by hand unless you
# are absolutely sure you know what you are doing! Use
# the supplied toolset to make changes (e.g. vgcfgrestore).

# dirs = [ "/etc/lvm/metadata", "/mnt/disk2/lvm/metadata2" ]
#
# Event daemon
# dmeventd {
  # mirror_library is the library used when monitoring a mirror device.
  # "libdevmapper-event-lvm2mirror.so" attempts to recover from failures.
  # It removes failed devices from a volume group and reconfigures a
  # mirror as necessary.
  #
  # mirror_library = "libdevmapper-event-lvm2mirror.so"
#
#}
Appendix C. LVM Object Tags

An LVM tag is a word that can be used to group LVM2 objects of the same type together. Tags can be attached to objects such as physical volumes, volume groups, and logical volumes. Tags can be attached to hosts in a cluster configuration. Snapshots cannot be tagged.

Tags can be given on the command line in place of PV, VG or LV arguments. Tags should be prefixed with @ to avoid ambiguity. Each tag is expanded by replacing it with all objects possessing that tag which are of the type expected by its position on the command line.

LVM tags are strings using [A-Za-z0-9_+.-] of up to 128 characters. They cannot start with a hyphen.

Only objects in a volume group can be tagged. Physical volumes lose their tags if they are removed from a volume group; this is because tags are stored as part of the volume group metadata and that is deleted when a physical volume is removed. Snapshots cannot be tagged.

The following command lists all the logical volumes with the database tag.

lvs @database

1. Adding and Removing Object Tags

To add or delete tags from physical volumes, use the --addtag or --deltag option of the pvchange command.

To add or delete tags from volume groups, use the --addtag or --deltag option of the vgchange or vgcreate commands.

To add or delete tags from logical volumes, use the --addtag or --deltag option of the lvchange or lvcreate commands.

2. Host Tags

In a cluster configuration, you can define host tags in the configuration files. If you set hosttags = 1 in the tags section, a host tag is automatically defined using the machine's hostname. This allow you to use a common configuration file which can be replicated on all your machines so they hold identical copies of the file, but the behavior can differ between machines according to the hostname.

For information on the configuration files, see Appendix B, The LVM Configuration Files.

For each host tag, an extra configuration file is read if it exists: lvm_hosttag.conf. If that file defines new tags, then further configuration files will be appended to the list of files to read in.

For example, the following entry in the configuration file always defines tag1, and defines tag2 if the hostname is host1.

tags { tag1 { } tag2 { host_list = ["host1"] } }

75
3. Controlling Activation with Tags

You can specify in the configuration file that only certain logical volumes should be activated on that host. For example, the following entry acts as a filter for activation requests (such as `vgchange -a-y`) and only activates `vg1/lvol0` and any logical volumes or volume groups with the `database` tag in the metadata on that host.

```
activation { volume_list = ["vg1/lvol0", ":database"] }
```

There is a special match `@*` that causes a match only if any metadata tag matches any host tag on that machine.

As another example, consider a situation where every machine in the cluster has the following entry in the configuration file:

```
tags { hosttags = 1 }
```

If you want to activate `vg1/lvol2` only on host `db2`, do the following:

1. Run `lvchange --addtag @db2 vg1/lvol2` from any host in the cluster.

This solution involves storing hostnames inside the volume group metadata.
Appendix D. LVM Volume Group Metadata

The configuration details of a volume group are referred to as the metadata. By default, an identical copy of the metadata is maintained in every metadata area in every physical volume within the volume group. LVM volume group metadata is small and stored as ASCII.

If a volume group contains many physical volumes, having many redundant copies of the metadata is inefficient. It is possible to create a physical volume without any metadata copies by using the `--metadatacopies 0` option of the `pvcreate` command. Once you have selected the number of metadata copies the physical volume will contain, you cannot change that at a later point. Selecting 0 copies can result in faster updates on configuration changes. Note, however, that at all times every volume group must contain at least one physical volume with a metadata area (unless you are using the advanced configuration settings that allow you to store volume group metadata in a file system). If you intend to split the volume group in the future, every volume group needs at least one metadata copy.

The core metadata is stored in ASCII. A metadata area is a circular buffer. New metadata is appended to the old metadata and then the pointer to the start of it is updated.

You can specify the size of metadata area with the `--metadatasize` option of the `pvcreate` command. The default size is too small for volume groups with many logical volumes or physical volumes.

1. The Physical Volume Label

By default, the `pvcreate` command places the physical volume label in the 2nd 512-byte sector. This label can optionally be placed in any of the first four sectors, since the LVM tools that scan for a physical volume label check the first 4 sectors. The physical volume label begins with the string `LABELONE`.

The physical volume label Contains:

- Physical volume UUID
- Size of block device in bytes
- NULL-terminated list of data area locations
- NULL-terminated lists of metadata area locations

Metadata locations are stored as offset and size (in bytes). There is room in the label for about 15 locations, but the LVM tools currently use 3: a single data area plus up to two metadata areas.

2. Metadata Contents
The volume group metadata contains:

- Information about how and when it was created
- Information about the volume group:

The volume group information contains:

- Name and unique id
- A version number which is incremented whenever the metadata gets updated
- Any properties: Read/Write? Resizeable?
- Any administrative limit on the number of physical volumes and logical volumes it may contain
- The extent size (in units of sectors which are defined as 512 bytes)
- An unordered list of physical volumes making up the volume group, each with:
  - Its UUID, used to determine the block device containing it
  - Any properties, such as whether the physical volume is allocatable
  - The offset to the start of the first extent within the physical volume (in sectors)
  - The number of extents
- An unordered list of logical volumes, each consisting of
  - An ordered list of logical volume segments. For each segment the metadata includes a mapping applied to an ordered list of physical volume segments or logical volume segments

3. Sample Metadata

The following shows an example of LVM volume group metadata for a volume group called myvg.

```
# Generated by LVM2: Tue Jan 30 16:28:15 2007
contents = "Text Format Volume Group"
version = 1
description = "Created *before* executing 'lvextend -L+5G /dev/myvg/mylv /dev/sdc'"
creation_host = "tng3-1" # Linux tng3-1 2.6.18-8.el5 #1 SMP Fri Jan 26 14:15:21 EST 2007 i686
creation_time = 1170196095 # Tue Jan 30 16:28:15 2007

myvg {
    id = "0zd3UT-wbYT-lDHq-lMPs-EjoE-0o18-wL28X4"
    seqno = 3
    status = ["RESIZEABLE", "READ", "WRITE"]
```
3. Sample Metadata

extent_size = 8192       # 4 Megabytes
max_lv = 0
max_pv = 0

physical_volumes {

  pv0 {
    id = "ZBW5qW-dXF2-0bGw-2CAd-2RlV-phwu-1c1RFf"
    device = "/dev/sda"            # Hint only
    status = ["ALLOCATABLE"]
    dev_size = 35964301           # 17.1491 Gigabytes
    pe_start = 384
    pe_count = 4390               # 17.1484 Gigabytes
  }

  pv1 {
    id = "ZHEZJW-MR64-D3QM-Rv7V-Hxsa-zU24-wztY19"
    device = "/dev/sdb"            # Hint only
    status = ["ALLOCATABLE"]
    dev_size = 35964301           # 17.1491 Gigabytes
    pe_start = 384
    pe_count = 4390               # 17.1484 Gigabytes
  }

  pv2 {
    id = "wCoG4p-55Ui-9tbp-VTEA-jO6s-RAVx-UREW0G"
    device = "/dev/sdc"            # Hint only
    status = ["ALLOCATABLE"]
    dev_size = 35964301           # 17.1491 Gigabytes
    pe_start = 384
    pe_count = 4390               # 17.1484 Gigabytes
  }

  pv3 {
    id = "hGlUwi-zsBg-39FF-do88-pHxY-8XA2-9WKIiA"
    device = "/dev/sdd"            # Hint only
    status = ["ALLOCATABLE"]
    dev_size = 35964301           # 17.1491 Gigabytes
    pe_start = 384
    pe_count = 4390               # 17.1484 Gigabytes
  }
}

logical_volumes {

  mylv {
    id = "GhUYSF-qVM3-rzQo-a6D2-o0aV-LQet-Ur90F9"
    status = ["READ", "WRITE", "VISIBLE"]
    segment_count = 2

    segment1 {
      start_extent = 0
      extent_count = 1280               # 5 Gigabytes

      type = "striped"                  # linear
      stripe_count = 1
      stripes = [
        "pv0", 0
      ]
    }

    segment2 {
      start_extent = 1280
      extent_count = 1280               # 5 Gigabytes
    }
  }
}
3. Sample Metadata

type = "striped"
stripe_count = 1       # linear

stripes = [
    "pvl", 0
]
}
Index

A
activating logical volumes
   individual nodes, 39
activating volume groups, 25
   individual nodes, 26
   local node only, 26
administrative procedures, 15
allocation
   policy, 22
   preventing, 21
archive file, 16, 26

B
backup
   file, 16
   metadata, 16, 26
backup file, 26
block device
   scanning, 20

C
cache file
   building, 24
cluster environment, 3, 15
CLVM
   definition, 3
clvmd daemon, 3
command line units, 18
configuration examples, 51
creating
   logical volume, 28
   logical volume, example, 51
LVM volumes in a cluster, 15
physical volumes, 19
striped logical volume, example, 52
volume groups, 22
creating LVM volumes
   overview, 15

D
data relocation, online, 39
deactivating volume groups, 25
   exclusive on one node, 26
   local node only, 26
device numbers
   major, 32
   minor, 32
   persistent, 32
device path names, 18
device scan filters, 38
device size, maximum, 23
device special file directory, 23
display
   sorting output, 48
displaying
   logical volumes, 33, 45
   physical volumes, 21, 42
   volume groups, 23, 44

E
extent
   allocation, 22
   definition, 8, 22

F
failed devices
   displaying, 59
file system
   growing on a logical volume, 16
   filters, 38

G
growing file system
   logical volume, 16

H
help display, 19

I
initializing
   partitions, 20
   physical volumes, 20
Insufficient Free Extents message, 65

L
linear logical volume
   converting to mirrored, 32
   creation, 28
   definition, 8
logging, 16
logical volume
   administration, general, 28
   changing parameters, 33
   creation, 28
creation example, 51
definition, 1, 8
displaying, 33, 40, 45
exclusive access, 39
extending, 34
growing, 34
linear, 28
local access, 39
lvs display arguments, 45
mirrored, 30
reducing, 36
removing, 33
renaming, 33
resizing, 32
shrinking, 36
snapshot, 37
striped, 30
lvchange command, 33
lvconvert command, 32
lvcreate command, 28
lvdisplay command, 34
lvextend command, 34
LVM
architecture overview, 2
clustered, 3
components, 2, 6
custom report format, 40
directory structure, 23
help, 19
history, 2
label, 6
logging, 16
logical volume administration, 28
physical volume administration, 19
physical volume, definition, 6
volume group, definition, 8
LVM1, 2
LVM2, 2
lvmdiskscan command, 20
lvreduce command, 32, 36
lvremove command, 33
lvrename command, 33
lvs command, 40, 45
display arguments, 45
lvscan command, 34
M
man page display, 19
metadata
backup, 16, 26
recovery, 63
mirrored logical volume
converting to linear, 32
creation, 30
definition, 12
failure recovery, 60
reconfiguration, 32
O
online data relocation, 39
P
partition type, setting, 19
partitions
multiple, 7
path names, 18
persistent device numbers, 32
physical extent
preventing allocation, 21
physical volume
adding to a volume group, 23
administration, general, 19
creating, 19
definition, 6
display, 42
displaying, 21, 40
illustration, 6
initializing, 20
layout, 6
pvs display arguments, 42
recovery, 64
removing, 22
removing from volume group, 24
removing lost volume, 64
resizing, 22
pvdisplay command, 21
pvmove command, 39
pvremove command, 22
pvresize command, 22
pvs command, 40
display arguments, 42
pvscan command, 21
R
removing
disk from a logical volume, 56
logical volume, 33
physical volumes, 22
renaming
  logical volume, 33
  volume group, 27
report format, LVM devices, 40
resizing
  logical volume, 32
  physical volume, 22

S
scanning
  block devices, 20
scanning devices, filters, 38
snapshot logical volume
  creation, 37
snapshot volume
  definition, 13
striped logical volume
  creation, 30
  creation example, 52
  definition, 11
  extending, 35
  growing, 35

T
troubleshooting, 59

U
units, command line, 18

V
verbose output, 18
vgcfbackup command, 27
vgcfrestore command, 27
vgchange command, 25
vgcreate command, 22
vgdisplay command, 23
vgexport command, 27
vgextend command, 23
vgimport command, 27
vgmerge command, 26
vgmknodes command, 28
vgreduce command, 24
vgrename command, 27
vgs command, 40
  display arguments, 44
vgscan command, 24
vgsplit command, 26
volume group
  activating, 25
  administration, general, 22
  changing parameters, 25
  combining, 26
  creating, 22
  deactivating, 25
  definition, 8
  displaying, 23, 40, 44
  extending, 23
  growing, 23
  merging, 26
  moving between systems, 27
  reducing, 24
  removing, 26
  renaming, 27
  shrinking, 24
  splitting, 26
  example procedure, 53
vgs display arguments, 44