Red Hat Enterprise Linux 5

LVM

Руководство администратора LVM

Редакция 3

Last Updated: 2017-10-16

Red Hat Enterprise Linux 5 LVM

Руководство администратора LVM
Редакция 3

Landmann
rlandmann@redhat.com

Юридическое уведомление

Аннотация

В данном документе будет рассмотрен менеджер логических томов LVM, а также сведения о выполнении LVM в кластерном окружении. Приводимая информация специфична для LVM2.

Содержание

ВВЕДЕНИЕ

1. ОБ ЭТОМ РУКОВОДСТВЕ

В данном документе рассматривается менеджер логических томов (LVM, Logical Volume Manager), включая сведения о выполнении LVM в кластерном окружении. Приводимая информация специфична для LVM2.

2. ЦЕЛЕВАЯ АУДИТОРИЯ

Содержимое данного документа предназначено для системных администраторов, управляющих системами с операционной системой (ОС) Linux. Обязательны навыки администрирования Red Hat Enterprise Linux 5 и GFS.

3. ПРОГРАММНЫЕ ВЕРСИИ

Таблица 1. Программные версии

ПО	Описание
RHEL5	RHEL5 или более поздние версии
GFS	GFS для RHEL5 или более поздних версий

4. ДОПОЛНИТЕЛЬНАЯ ДОКУМЕНТАЦИЯ

Подробная информация об использовании Red Hat Enterprise Linux может быть найдена в следующих руководствах:

- *Руководство по установке Red Hat Enterprise Linux*: Рассматривает различные аспекты установки Red Hat Enterprise Linux 5.
- *Руководство по развертыванию Red Hat Enterprise Linux*: Содержит информацию о развертывании, конфигурации и администрированию Red Hat Enterprise Linux 5.

Подробная информация об Red Hat Cluster Suite для Red Hat Enterprise Linux 5 может быть найдена в следующих руководствах:

- *Обзор Red Hat Cluster Suite*: Содержит общую информацию о Red Hat Cluster Suite.
- *Конфигурация и управление кластером Red Hat*: Содержит информацию об установке, настройке и управлении кластерными компонентами Red Hat.
- *GFS: Конфигурация и администрирование*: Содержит информацию об установке, настройке и поддержке работы глобальной файловой системы (GFS, Global File System) Red Hat.

- *GFS2: Конфигурация и администрирование*: Содержит информацию об установке, настройке и поддержке работы глобальной файловой системы 2 (GFS2, Global File System 2) Red Hat.
- *Использование Device-Mapper Multipath*: Содержит информацию о возможностях Device-Mapper Multipath для Red Hat Enterprise Linux 5.
- *Использование GNBD с GFS*: Содержит обзор использования устройств GNBD с Red Hat GFS.
- *Администрирование виртуального сервера Linux*: Предоставляет информацию о настройке высокопроизводительных систем и служб для работы с виртуальным сервером Linux (LVS, Linux Virtual Server).
- *Замечания к выпуску Red Hat Cluster Suite*: Предоставляют информацию о текущем выпуске Red Hat Cluster Suite.

Документация Red Hat Cluster Suite и другие руководства Red Hat доступны в форматах HTML и PDF на диске документации Red Hat Enterprise Linux и в Интернете по адресу http://www.redhat.com/docs/.

5. ЖДЕМ ВАШИХ ОТЗЫВОВ

Если вы обнаружили ошибку или опечатку, а, может, у вас есть предложения по усовершенствованию данного документа, мы бы хотели услышать об этом. Отправьте сообщение в систему регистрации ошибок Bugzilla (http://bugzilla.redhat.com/bugzilla/) относительно компонента **rh-cs**.

Be sure to mention the manual's identifier:

```
Bugzilla component: Documentation-cluster
Book identifier: Cluster_Logical_Volume_Manager(EN)-5 (2009-01-05T15:20)
```

By mentioning this manual's identifier, we know exactly which version of the guide you have.

Если у вас есть предложения по улучшению документации, попытайтесь описать их как можно более детально. Если вы нашли ошибку, пожалуйста, укажите номер раздела и часть окружающего текста, чтобы мы смогли быстрее ее найти.

ГЛАВА 1. МЕНЕДЖЕР ЛОГИЧЕСКИХ ТОМОВ (LVM)

Данная глава содержит общий обзор компонентов менеджера логических томов (LVM, Logical Volume Manager).

1.1. ЛОГИЧЕСКИЕ ТОМА

При управлении томами над физическим хранилищем создается слой абстракции, что позволяет создавать логические тома. При этом достигается бóльшая гибкость по сравнению с использованием физического хранилища напрямую.

Логический том позволяет обеспечить виртуализацию хранилища; снимаются ограничения, накладываемые размерами физического диска. Дополнительно, конфигурация аппаратного накопителя спрятана от ПО, поэтому возможно изменить его размер или переместить без необходимости остановки приложений и отключения файловых систем, что поможет сократить издержки.

Преимущества логических томов по сравнению с использованием физических накопителей:

- Возможность изменения емкости

  При использовании логических томов размер файловых систем не ограничивается одним диском, так как вы объединяете диски и разделы в один логический том.

- Возможность изменения размера пула хранилища

  С помощью простых команд можно увеличивать или уменьшать размер логических томов без необходимости переформатирования или переразбиения дисков.

- Живое перемещение данных

  Для создания новых, более быстрых и более устойчивых подсистем хранения вы теперь можете перемещать данные, в то время как система активна. Данные можно перемещать, даже если к дискам выполняется обращение. Например, можно на ходу освободить заменяемый диск перед его удалением.

- Удобство присвоения имен устройствам

  Логические тома могут объединяться в группы для облегчения их управления. Группам, в свою очередь, могут присваиваться любые имена.

- Чередование дисков

  Возможно создать логический том с чередованием данных на двух или нескольких дисках. Это может значительно улучшить производительность.

- Зеркалирование томов

  Используя логические тома, можно легко настроить зеркало для ваших данных.

- Снимки томов

  Используя логические тома, можно создавать снимки устройств с целью создания резервных копий или тестирования результата внесенных изменений, не рискуя действительными данными.

Все перечисленные возможности будут рассмотрены в этом документе.

1.2. ОБЗОР АРХИТЕКТУРЫ LVM

Исходный менеджер логических томов LVM1, появившийся впервые в RHEL4, был заменен на LVM2, архитектура которого более универсальна. LVM2 обладает следующими преимуществами:

- гибкая емкость
- эффективность хранения метаданных
- улучшенный формат восстановления
- новый формат метаданных ASCII
- атомарные изменения метаданных
- избыточные копии метаданных

LVM2 обратно совместим с LVM1 (за исключением снимков и кластерной поддержки). Вы можете преобразовать формат LVM1 группы томов в LVM2 с помощью команды **vgconvert**. Информация о преобразовании формата метаданных LVM может быть найдена на странице помощи **vgconvert**(8).

В основе логического тома LVM лежит блочное устройство (раздел или даже целый диск). Устройство инициализируется как *физический том* LVM.

При создании логического тома LVM физические тома объединяются в *группы томов*, что создает пул дискового пространства для организации логических томов. Этот процесс аналогичен разбиению дисков на разделы. Логический том используется файловыми системами и приложениями подобно тому, как используются базы данных.

Рисунок 1.1, «LVM Logical Volume Components» shows the components of a simple LVM logical volume:

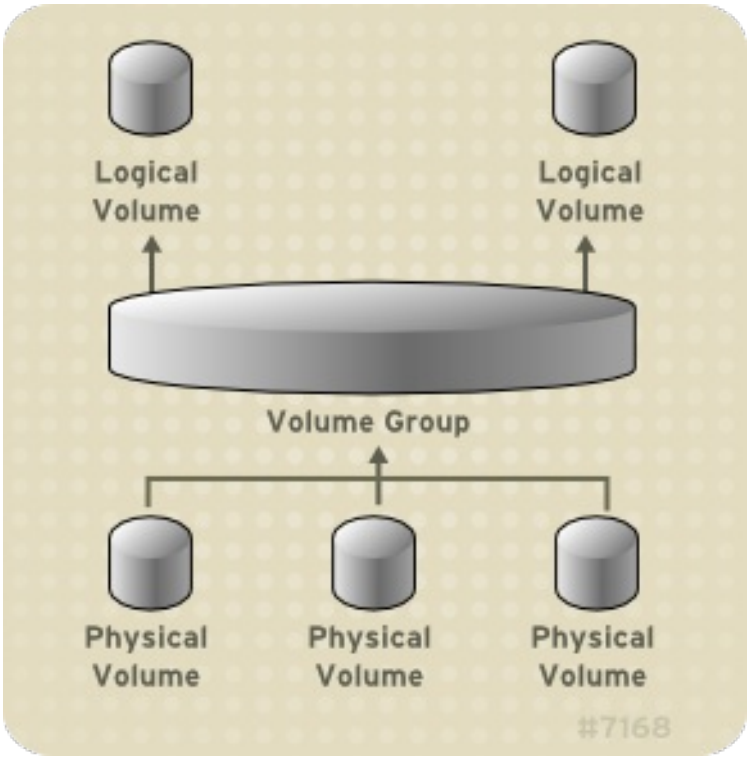


Рисунок 1.1. LVM Logical Volume Components

For detailed information on the components of an LVM logical volume, see Глава 2, *Компоненты LVM*.

1.3. КЛАСТЕРНЫЙ МЕНЕДЖЕР ЛОГИЧЕСКИХ ТОМОВ

Кластерный менеджер логических томов (CLVM, Clustered Logical Volume Manager) представляет собой набор кластерных расширений для LVM, позволяющих кластеру управлять совместным хранилищем (например, на SAN) с помощью LVM.

Возможность использования CLVM также определяется системными требованиями:

- Если доступ к хранилищу необходим лишь одному узлу, то можно использовать LVM без расширений CLVM. При этом создаваемые логические тома будут локальными.
- Если для обеспечения восстановления после отказа вы используете кластерную систему, в которой только один узел может обращаться к хранилищу в заданный момент времени, потребуются агенты HA-LVM (High Availability Logical Volume Management). *Конфигурация и управление кластером Red Hat* предоставляет информацию об агентах HA-LVM.
- Если нескольким узлам необходимо обращаться к хранилищу, доступ к которому разделяется между активными узлами, то потребуется использовать CLVM. CLVM позволит пользователю настроить логические тома на разделяемом хранилище за счет блокирования доступа к физическому хранилищу на время настройки логического тома, используя при этом кластерные службы блокирования для управления разделяемым хранилищем.

Для работы CLVM необходимо, чтобы выполнялось программное обеспечение Red Hat Cluster Suite, включая **`clmvd`**. Демон **`clmvd`** является основным кластерным расширением LVM, выполняется на каждом компьютере в кластере и передает им обновления метаданных LVM, тем самым обеспечивая постоянство представления логических томов. Более подробную информацию по установке и администрированию Red Hat Cluster Suite можно найти в руководстве *Конфигурация и управление кластером Red Hat*.

Чтобы проверить, был ли запущен **clmvd** во время загрузки, выполните команду

```
# chkconfig clvmd on
```

Если демон **clvmd** не запущен, выполните команду

```
# service clvmd start
```

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 graphical user interface, as described in Глава 4, *Администрирование LVM с помощью команд* and Глава 7, *Администрирование LVM при помощи графического интерфейса*. 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.

By default, logical volumes created with CLVM on shared storage are visible to all computers that have access to the shared storage. It is possible, however, to create logical volumes when the storage devices are visible to only one node in the cluster. It is also possible to change the status of a logical volume from a local volume to a clustered volume. For information, see Раздел 4.3.2, «Создание групп томов в кластере» and Раздел 4.3.7, «Изменение параметров группы томов».

Рисунок 1.2, «Обзор CLVM» shows a CLVM overview in a Red Hat cluster.

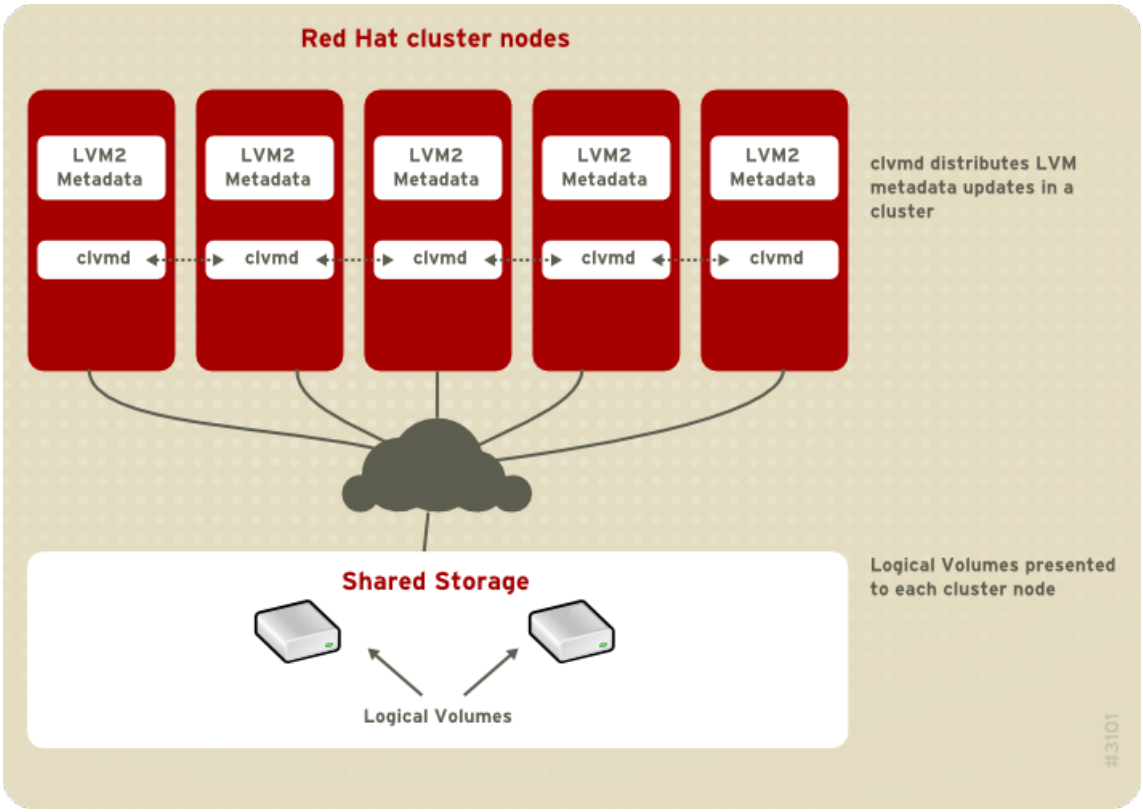
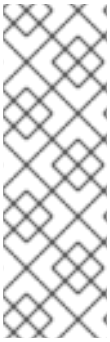


Рисунок 1.2. Обзор CLVM



ПРИМЕЧАНИЕ

Для работы разделяемого хранилища Red Hat Cluster Suite необходимо, чтобы выполнялся демон **clvmd** или агенты HA-LVM (High Availability Logical Volume Management). Если нет возможности использовать **clvmd** или агенты HA-LVM в силу отсутствия полномочий или каких-либо других причин, то не используйте единственный экземпляр LVM на разделяемом диске, так как это может привести к повреждению данных. Если у вас есть вопросы, за помощью можно обратиться к представителю Red Hat.

ПРИМЕЧАНИЕ

CLVM requires changes to the **lvm.conf** file for cluster-wide locking. Information on configuring the **lvm.conf** file to support clustered locking is provided within the **lvm.conf** file itself. For information about the **lvm.conf** file, see Приложение B, *Файлы конфигурации LVM*.

1.4. СОДЕРЖАНИЕ ДОКУМЕНТА

Включает:

- Глава 2, *Компоненты LVM* describes the components that make up an LVM logical volume.
- Глава 3, *Обзор администрирования LVM* 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).
- Глава 4, *Администрирование LVM с помощью команд* summarizes the individual administrative tasks you can perform with the LVM CLI commands to create and maintain logical volumes.
- Глава 5, *Примеры конфигурации LVM* provides a variety of LVM configuration examples.
- Глава 6, *Решение проблем LVM* provides instructions for troubleshooting a variety of LVM issues.
- Глава 7, *Администрирование LVM при помощи графического интерфейса* summarizes the operating of the LVM GUI.
- Приложение A, *Device Mapper* describes the Device Mapper that LVM uses to map logical and physical volumes.
- Приложение B, *Файлы конфигурации LVM* describes the LVM configuration files.
- Приложение C, *Теги объектов LVM* describes LVM object tags and host tags.
- Приложение D, *Метаданные группы томов LVM* describes LVM volume group metadata, and includes a sample copy of metadata for an LVM volume group.

ГЛАВА 2. КОМПОНЕНТЫ LVM

В данном разделе рассматриваются компоненты логических томов LVM.

2.1. ФИЗИЧЕСКИЕ ТОМА

В основе логического тома LVM лежит блочное устройство (раздел или даже целый диск). Устройство инициализируется как *физический том* LVM. При этом в начале размещается специальная метка.

Метка LVM по умолчанию размещается во втором 512-байтном секторе. Это может быть изменено -- метка может быть расположена в любом из первых четырех секторов, что позволяет логическим томам использовать эти сектора параллельно с другими пользователями.

Метка LVM идентифицирует и организует устройства для физического устройства (т.к. они могут быть определены в любом порядке при загрузке системы). Она сохраняется между перезагрузками в пределах кластера.

Метка LVM идентифицирует устройство как физический том LVM. Она содержит случайный уникальный идентификатор (UUID), размер блочного устройства (в байтах) и записывает расположение метаданных LVM на этом устройстве.

Метаданные LVM включают детали конфигурации групп томов LVM в вашей системе. По умолчанию идентичные копии метаданных поддерживаются в каждой секции метаданных всех логических томов в составе группы. Метаданные не занимают много места и хранятся в формате ASCII.

В настоящее время LVM позволяет сохранять 0, 1 или 2 идентичных копии метаданных для каждого физического тома. По умолчанию сохраняется одна копия. Задав число копий один раз, вы не сможете его изменить позднее. Первая копия хранится в начале устройства, вскоре после метки. Вторая копия (если она существует) располагается в конце устройства. Если вы случайно перезаписали область в начале диска, вторая копия, расположенная в конце диска, позволит восстановить метаданные.

For detailed information about the LVM metadata and changing the metadata parameters, see Приложение D, *Метаданные группы томов LVM*.

2.1.1. LVM Physical Volume Layout

Рисунок 2.1, «Организация физических томов» 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.

ПРИМЕЧАНИЕ

В ядре Linux (а также в данном документе) подразумевается, что размер секторов составляет 512 байт.

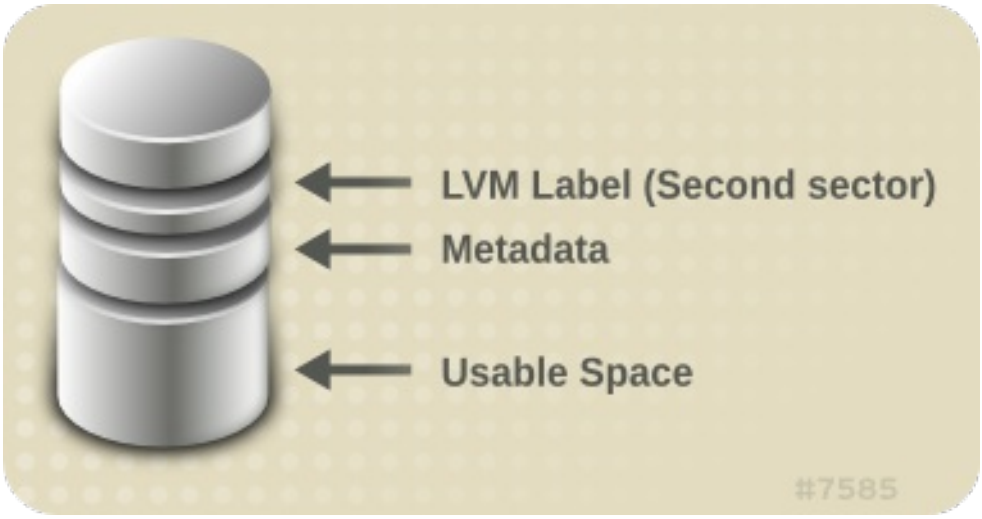


Рисунок 2.1. Организация физических томов

2.1.2. Несколько разделов на диске

LVM позволяет создавать физические тома на основе дисковых разделов. Обычно рекомендуется создать единственный раздел, охватывающий весь диск, и присвоить ему метку физического тома LVM по следующим причинам:

- Облегчение администрирования

  Намного легче следить за системным оборудованием, если каждый настоящий диск появляется лишь раз, в особенности, если произошел его сбой. Кроме того, несколько физических томов на одном диске при загрузке приведут к отображению предупреждения ядра о неизвестных типах разделов.

- Чередование производительности

  LVM не может определить, расположены ли два физических тома на одном и том же физическом диске. Если вы создадите логический том с чередованием, и при этом два физических тома расположены на одном физическом диске, может оказаться так, что области чередования расположены на различных разделах одного диска. Это приведет к снижению производительности.

Хоть это и не рекомендуется, но может случиться так, что необходимо разделить диск на различные физические тома LVM. Например, в системе с несколькими дисками может понадобиться перенести данные между разделами в случае миграции существующей системы на тома LVM. Или если в наличии имеется большой диск, а вы хотите создать несколько групп томов, тогда придется разбить диск на разделы. Если у вас есть такой диск с несколькими разделами, и эти разделы принадлежат одной группе томов, при создании томов с чередованием уделите особое внимание тому, какие разделы будут включены в логический том.

2.2. ГРУППЫ ТОМОВ

Физические тома объединяются в группы томов, что позволяет создать пул дискового пространства, из которого будет выделяться место для логических томов.

В пределах группы доступное дисковое пространство разделяется на блоки фиксированного размера, называемые «экстентами». Экстент является наименьшим блоком, который может быть выделен. На уровне физических томов используется понятие физических экстентов.

Логическому тому будут выделяться логические экстенты, размер которых равен размеру физических экстентов. Поэтому размер экстентов всегда один и тот же для всех логических томов в группе томов. Группа томов задает соответствие логических экстентов физическим.

2.3. ЛОГИЧЕСКИЕ ТОМА LVM

В LVM группа томов разделяется на логические тома. Существует три типа логических томов LVM — *линейные* тома, тома *с чередованием* и *зеркальные*. Они будут рассмотрены далее.

2.3.1. Линейный том

Линейный том объединяет физические тома в один логический том. Например, если в наличии имеется два диска по 60 Гб, то вы можете создать логический том размером 120 Гб.

Creating a linear volume assigns a range of physical extents to an area of a logical volume in order. For example, as shown in Рисунок 2.2, «Сопоставления экстентов» 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.

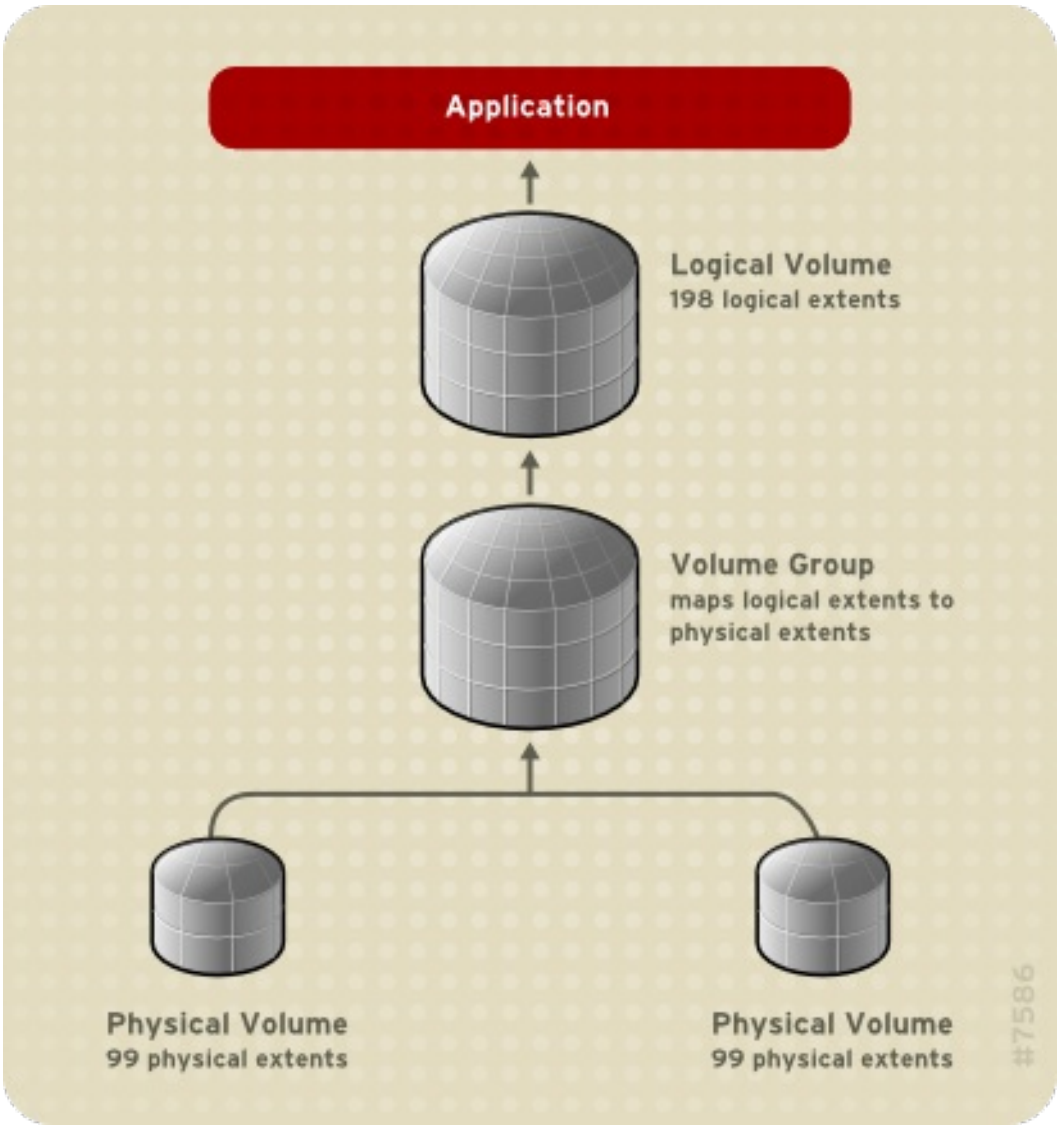


Рисунок 2.2. Сопоставления экстентов

The physical volumes that make up a logical volume do not have to be the same size. Рисунок 2.3, «Линейный том с неравными физическими томами» 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.

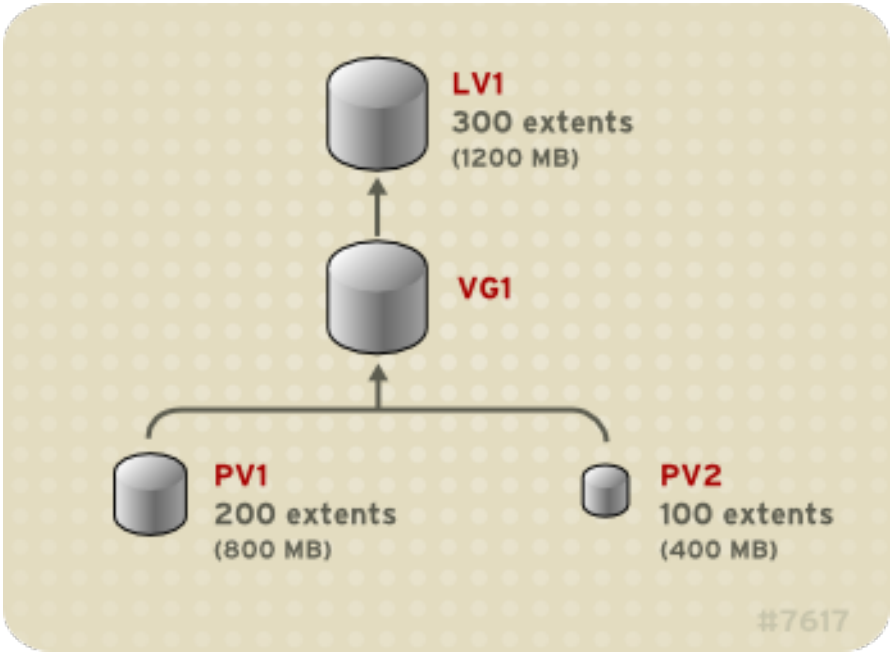


Рисунок 2.3. Линейный том с неравными физическими томами

You can configure more than one linear logical volume of whatever size you desire from the pool of physical extents. Рисунок 2.4, «Различные логические тома» shows the same volume group as in Рисунок 2.3, «Линейный том с неравными физическими томами», 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).

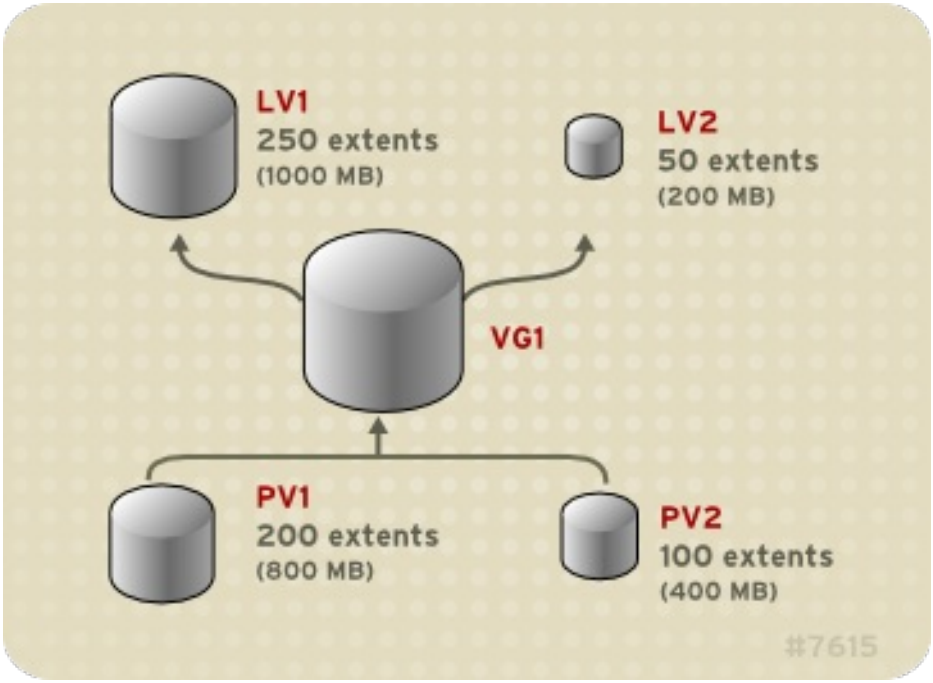


Рисунок 2.4. Различные логические тома

2.3.2. Логические тома с чередованием

При записи данных в логический том LVM файловая система располагает данные на физических томах, на основе которых и создан логический том. Если вы хотите контролировать, как происходит запись данных, то создайте логический том с чередованием. Это поможет улучшить производительность ввода/ вывода в случае интенсивных действий чтения и записи.

Чередование повышает производительность, так как данные поочередно записываются в предопределенные тома. Поэтому операции ввода и вывода могут выполняться параллельно. Иногда производительность для каждого дополнительного физического тома может даже сравниться с линейной организацией.

Следующий пример демонстрирует чередование данных между тремя физическими томами. На этой картинке:

- первая секция данных записывается на PV1
- вторая секция данных записывается на PV2
- третья секция данных записывается на PV3
- четвертая секция данных записывается на PV1

Размер сегментов логического тома с чередованием не может превышать размер экстента.

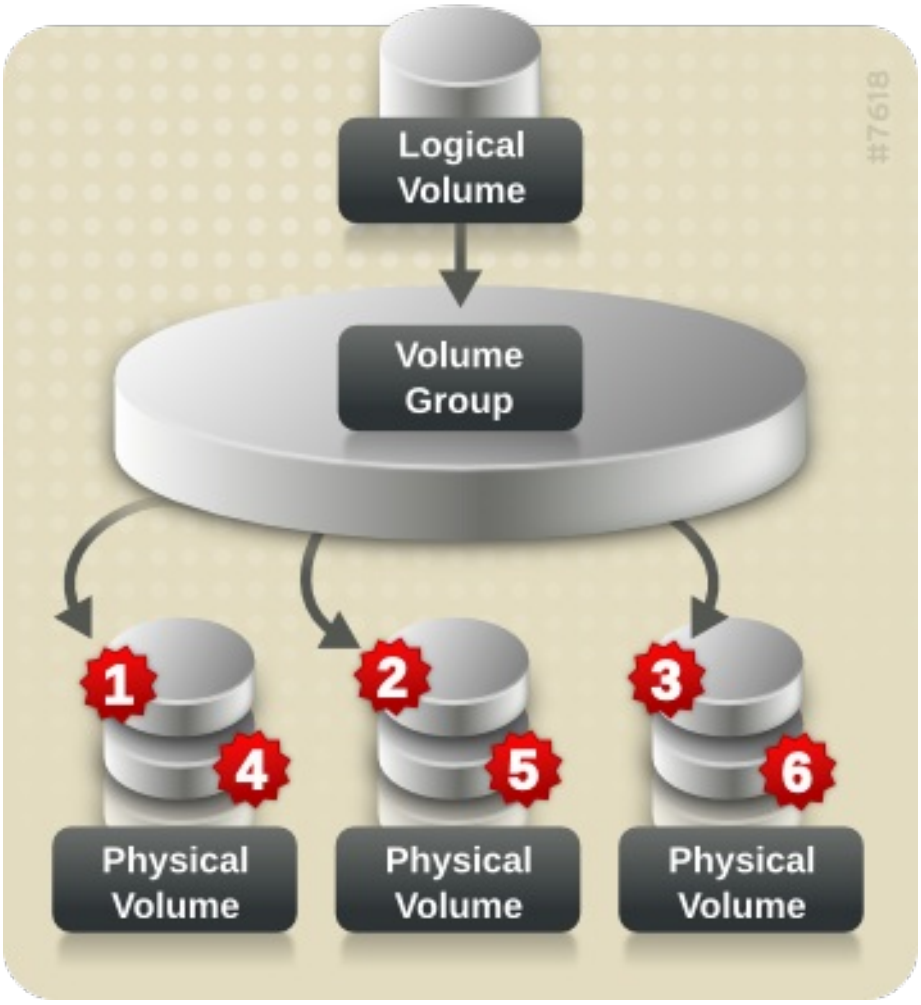


Рисунок 2.5. Чередование данных между тремя физическими томами

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 Раздел 4.4.9, «Увеличение размера тома, использующего чередование».

2.3.3. Зеркальные логические тома

Зеркало обычно содержит идентичные копии данных на разных устройствах. При записи данных на одно устройство их копия также записывается на другое, что обеспечивает безболезненное восстановление после сбоя. В случае сбоя одной стороны зеркала логический том будет преобразован в линейный и продолжит работу.

LVM поддерживает возможность зеркалирования томов. При создании такого логического тома LVM записывает копию данных на отдельный физический том. LVM также позволяет создавать несколько зеркал для логических томов.

Зеркало LVM подразделяет зеркалируемое устройство на секции, размер которых обычно составляет 512 Кбайт. LVM ведет журнал для отслеживания того, какие регионы синхронизированы с зеркалами. Этот журнал может храниться на диске (в таком случае он будет сохраняться между перезагрузками) или просто временно находиться в памяти.

Рисунок 2.6, «Mirrored Logical Volume» shows a mirrored logical volume with one mirror. In this configuration, the log is maintained on disk.

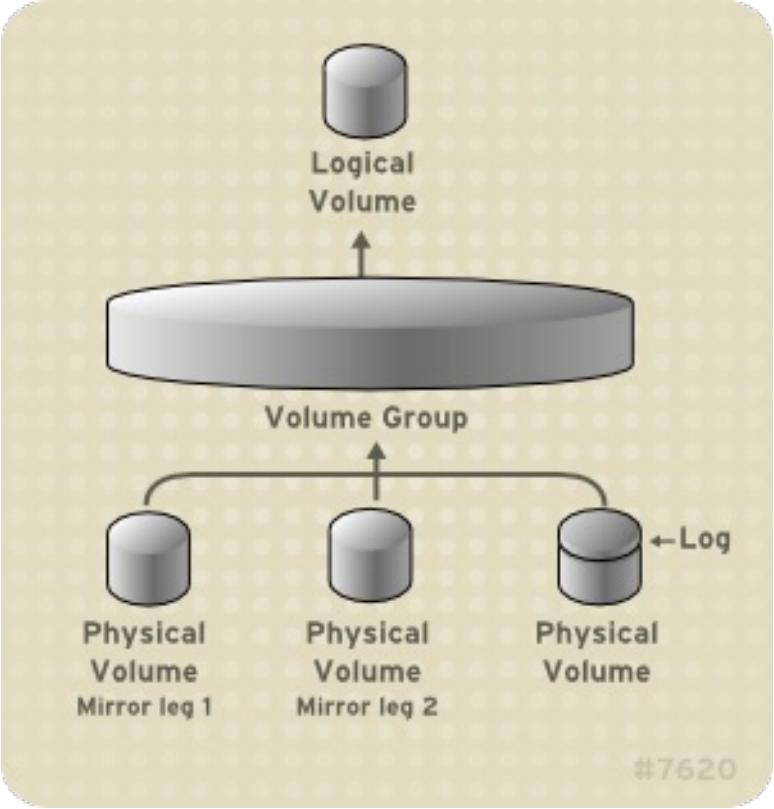


Рисунок 2.6. Mirrored Logical Volume

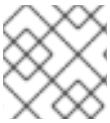
ПРИМЕЧАНИЕ

Начиная с версии RHEL 5.3, осуществляется поддержка зеркальных логических томов.

For information on creating and modifying mirrors, see Раздел 4.4.1.3, «Создание зеркальных томов».

2.3.4. Снимки

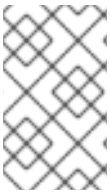
Возможность LVM создавать снимки позволяет создавать виртуальные образы устройств в определенный момент без необходимости остановки служб. Если исходное устройство было изменено уже после создания снимка, будет создана копия измененных данных, чтобы впоследствии можно было бы воссоздать состояние устройства.



ПРИМЕЧАНИЕ

Снимки LVM не работают с разными узлами кластера.

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.



ПРИМЕЧАНИЕ

Снимки файловой системы не являются полнофункциональными резервными копиями, а всего лишь виртуальными. Поэтому заменять резервирование они не могут.

Если снимок заполняется, он будет удален. Это делается для того, чтобы освободить место для исходной файловой системы. Рекомендуется периодически проверять размер снимка. Его размер также можно изменять, поэтому можно увеличить размер снимка при необходимости. И наоборот, если вы обнаружили, что размер снимка слишком большой, можно его уменьшить и освободить дополнительный объем.

При создании снимка файловой системы полный доступ чтения и записи к ней остается неизменным. Если же часть снимка изменена, то она будет отмечена и не будет копироваться.

Когда же применяются снимки?

- Обычно снимок создается, если необходимо создать резервную копию логического тома, не останавливая активную систему, данные в которой постоянно обновляются.
- Для проверки целостности файловой системы снимка можно использовать команду **`fsck`**.
- Так как сам снимок доступен для чтения и записи, можно тестировать приложения, создавая снимок и экспериментируя с ним. При этом действительные данные сохраняются нетронутыми.
- Снимки могут использоваться при работе с виртуальной машиной Xen. При этом можно создать образ диска, сделать его снимок и затем его изменить для конкретного домена domU. Для другого domU можно создать другой снимок. Так сохраняются лишь изменяющиеся секции, а основной объем тома доступен для совместного использования.

ГЛАВА 3. ОБЗОР АДМИНИСТРИРОВАНИЯ LVM

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 Глава 5, *Примеры конфигурации LVM*.

For descriptions of the CLI commands you can use to perform LVM administration, see Глава 4, *Администрирование LVM с помощью команд*. Alternately, you can use the LVM GUI, which is described in Глава 7, *Администрирование LVM при помощи графического интерфейса*.

3.1. СОЗДАНИЕ ТОМОВ LVM В КЛАСТЕРЕ

To create logical volumes in a cluster environment, you use the Clustered Logical Volume Manager (CLVM), which 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. In order to use CLVM, the Red Hat Cluster Suite software, including the **clmvd** daemon, must be started at boot time, as described in Раздел 1.3, «Кластерный менеджер логических томов».

Процесс создания логических томов LVM в кластерном окружении идентичен созданию томов на отдельном узле. Используемые команды не отличаются, а при работе с графическим интерфейсом ничего не меняется. Чтобы иметь возможность активации создаваемых томов LVM, необходимо, чтобы кластерная инфраструктура была активна, а для кластера должен быть определен кворум.

CLVM requires changes to the **lvm.conf** file for cluster-wide locking. Information on configuring the **lvm.conf** file to support clustered locking is provided within the **lvm.conf** file itself. For information about the **lvm.conf** file, see Приложение B, *Файлы конфигурации LVM*.

By default, logical volumes created with CLVM on shared storage are visible to all computers that have access to the shared storage. It is possible, however, to create logical volumes when the storage devices are visible to only one node in the cluster. It is also possible to change the status of a logical volume from a local volume to a clustered volume. For information, see Раздел 4.3.2, «Создание групп томов в кластере» and Раздел 4.3.7, «Изменение параметров группы томов»

ПРИМЕЧАНИЕ

Для работы разделяемого хранилища Red Hat Cluster Suite необходимо, чтобы выполнялся демон **clvmd** или агенты HA-LVM (High Availability Logical Volume Management). Если нет возможности использовать **clvmd** или агенты HA-LVM в силу отсутствия полномочий или каких-либо других причин, то не используйте единственный экземпляр LVM на разделяемом диске, так как это может привести к повреждению данных. Если у вас есть вопросы, за помощью можно обратиться к представителю Red Hat.

Документ *Конфигурация и администрирование кластера Red Hat* содержит информацию об установке Red Hat Cluster Suite и настройке инфраструктуры кластера.

3.2. ОБЗОР СОЗДАНИЯ ЛОГИЧЕСКОГО ТОМА

Последовательность шагов при создании логического тома LVM:

1. Инициализация разделов как физических томов, которые будут использоваться логическим томом (присвоение им метки).

2. Создание группы томов.

3. Создание логического тома.

После создания логического тома можно создать и смонтировать файловую систему. Приводимые в данном документе примеры подразумевают использование файловых систем GFS.

1. С помощью команды **gfs_mkfs** создайте файловую систему GFS на логическом томе.

2. С помощью команды **mkdir** создайте точку монтирования. В кластерной системе создайте точку монтирования на каждом узле.

3. Смонтируйте файловую систему. Для каждого узла в системе можно добавить отдельную строчку в файле **fstab**.

Или же можно создать и смонтировать файловую систему GFS, воспользовавшись графическим интерфейсом.

Процесс создания тома LVM не зависит от оборудования, так как область хранения информации конфигурации LVM расположена на физических томах, а не на станции, где создается том. Использующие хранилище серверы обычно имеют локальные копии, но имеют возможность их воссоздания из содержимого физических томов. Если версии LVM совместимы, физические тома можно подключить к другому серверу.

3.3. УВЕЛИЧЕНИЕ РАЗМЕРА ФАЙЛОВОЙ СИСТЕМЫ ЛОГИЧЕСКОГО ТОМА

Последовательность действий, необходимых для увеличения размера файловой системы логического тома:

1. Создайте новый физический том.

2. Нарастите группу томов, в которую включен логический том с увеличиваемой файловой системой, так чтобы она содержала новый физический том.

3. Увеличьте размер логического тома, чтобы он включал новый физический том.

4. Увеличьте размер файловой системы.

Если группа томов обладает достаточным объемом нераспределенного пространства, то его можно использовать для расширения логического тома, опустив шаги 1 и 2.

3.4. РЕЗЕРВНОЕ КОПИРОВАНИЕ ЛОГИЧЕСКОГО ТОМА

Резервные копии и архивы метаданных создаются по умолчанию автоматически при каждом изменении настроек логических томов или группы томов (это можно отключить в файле **lvm.conf**). Также по умолчанию резервная копия метаданных сохраняется в **/etc/lvm/backup**, а архивы метаданных — в **/etc/lvm/archive**. Параметры, устанавливаемые в **lvm.conf**, определяют длительность хранения архивов метаданных в **/etc/lvm/archive**, а также число хранимых файлов. Ежедневное резервирование должно включать создание копии каталога **/etc/lvm**.

Обратите внимание, что при создании резервной копии метаданных системные данные и данные пользователя, содержащиеся в логических томах, не резервируются.

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 Раздел 4.3.12, «Создание резервной копии метаданных группы томов».

3.5. ЖУРНАЛИРОВАНИЕ

Вывод всех сообщений обрабатывается модулем журналирования. При этом используются следующие уровни журналирования:

- стандартный вывод/ ошибка
- syslog
- файл журнала
- внешняя функция журналирования

The logging levels are set in the **/etc/lvm/lvm.conf** file, which is described in Приложение B, *Файлы конфигурации LVM*.

ГЛАВА 4. АДМИНИСТРИРОВАНИЕ LVM С ПОМОЩЬЮ КОМАНД

Данный раздел содержит решения некоторых практических задач с помощью текстовых команд.

ПРИМЕЧАНИЕ

If you are creating or modifying an LVM volume for a clustered environment, you must ensure that you are running the **clvmd** daemon. For information, see see Раздел 3.1, «Создание томов LVM в кластере».

4.1. ИСПОЛЬЗОВАНИЕ КОМАНД

Сначала стоит упомянуть о некоторых основных возможностях команд LVM.

Если содержимое аргумента представляет собой величину объема информации, то единицы можно указать вручную. Если же единицы не указаны, по умолчанию будет подразумеваться использование Кбайт или Мбайт. Сами значения должны представлять собой целые числа.

При указании единиц в командной строке регистр не имеет значения. Например, M и m будут эквивалентны, при этом значения будут кратны 1024. Но если задан аргумент **--units**, то указание единиц в нижнем регистре будет означать, что они кратны 1024, а в верхнем регистре — 1000.

Если команды в качестве аргументов принимают имена логических томов или группы томов, то указывать полный путь не обязательно. Например, том **lvol0** в группе **vg0** можно определить как **vg0/lvol0**. Если требуется список групп томов, но при этом он не указан, по умолчанию будет подразумеваться список ВСЕХ групп томов. Если требуется определить список логических томов, но при этом задана группа томов, будет выполнена подстановка всех логических томов в заданной группе. К примеру, команда **lvdisplay vg0** отобразит список всех логических томов в группе **vg0**.

Для отображения подробного вывода укажите опцию **-v**, которую можно использовать каскадно для отображения детальной информации. К примеру, стандартный вывод команды **lvcreate** выглядит так:

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

Использование команды **lvcreate** с опцией **-v** отобразит следующее:

```
# lvcreate -v -L 50MB new_vg
    Finding volume group "new_vg"
  Rounding up size to full physical extent 52.00 MB
    Archiving volume group "new_vg" metadata (seqno 4).
    Creating logical volume lvol0
    Creating volume group backup "/etc/lvm/backup/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/backup/new_vg" (seqno 5).
  Logical volume "lvol0" created
```

Можно было бы использовать **-vv**, **-vvv** или даже **-vvvv**, чтобы отобразить еще более подробную информацию о ходе выполнения команды. Максимально подробный вывод будет достигнут при указании **-vvvv**. Следующий пример демонстрирует лишь первые несколько строк вывода команды **lvcreate -vvvv**:

```
# lvcreate -vvvv -L 50MB new_vg
#lvmcmdline.c:913         Processing: lvcreate -vvvv -L 50MB new_vg
#lvmcmdline.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
...
```

Для отображения вспомогательной информации об использовании команды, используйте опцию **--help**.

```
commandname --help
```

Для отображения страницы помощи используйте команду **man**:

```
man commandname
```

Например, **man lvm** отобразит общую информацию об 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 Раздел 6.4, «Восстановление метаданных физического тома».

4.2. АДМИНИСТРИРОВАНИЕ ФИЗИЧЕСКИХ ТОМОВ

В данной секции рассматриваются команды, используемые для работы с физическими томами.

4.2.1. Создание физических томов

В последующих подсекциях рассмотрены команды, используемые для создания физических томов.

4.2.1.1. Установка типа раздела

Если физический том использует целый диск, необходимо, чтобы на этом диске НЕ было

таблицы разделов. Для разделов DOS идентификатор раздела должен быть установлен в 0x8e с помощью **fdisk**, **cfdisk** или их аналога. При удалении таблицы разделов данные будут удалены автоматически. Существующую таблицу разделов можно удалить путем заполнения первого сектора нулями:

```
dd if=/dev/zero of=PhysicalVolume bs=512 count=1
```

4.2.1.2. Инициализация физических томов

Чтобы инициализировать блочное устройство в качестве физического тома, используйте команду **pvcreate**. Принципиально инициализация подобна форматированию файловой системы.

Команда инициализации **/dev/sdd1**, **/dev/sde1** и **/dev/sdf1** выглядит так:

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

Для инициализации разделов, а не отдельных дисков выполните команду **pvcreate** для раздела. Следующий пример инициализирует **/dev/hdb1** как физический том LVM для дальнейшего включения в логический том LVM.

```
pvcreate /dev/hdb1
```

4.2.1.3. Поиск блочных устройств

С помощью команды **lvmdiskscan** можно выполнить поиск блочных устройств для использования в качестве физических томов. Пример:

```
# lvmdiskscan
/dev/ram0                     [       16.00 MB]
  /dev/sda                    [       17.15 GB]
  /dev/root                   [       13.69 GB]
  /dev/ram                    [       16.00 MB]
  /dev/sda1                   [       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
```

4.2.2. Отображение физических томов

Для отображения информации о физических томах LVM используются команды **pvs**, **pvdisplay** и **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 Раздел 4.9, «Настройка отчетов для LVM».

pvdisplay отображает подробный многострочный вывод для каждого физического тома, включающий информацию о размере, экстентах, группе томов и пр. Формат вывода фиксирован.

Пример вывода команды **pvdisplay** для одного тома:

```
# 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
```

pvscan сканирует все поддерживаемые блочные устройства в системе на предмет наличия физических томов.

Следующая команда отобразит все найденные физические устройства:

```
# 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]
 Total: 3 [2.83 GB] / in use: 2 [1.88 GB] / in no VG: 1 [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 Раздел 4.6, «Управление сканированием устройств LVM с помощью фильтров».

4.2.3. Запрет выделения пространства физического тома

Команда **pvchange** позволяет запретить выделение физических экстентов или дополнительных физических томов, что может потребоваться в случае возникновения ошибок диска или при удалении физического тома.

Команда, запрещающая выделение физических экстентов на **/dev/sdk1**:

```
pvchange -x n /dev/sdk1
```

При указании опций **-xy** выделение экстентов будет разрешено там, где раньше оно было запрещено.

4.2.4. Изменение размера физического тома

Команда **pvresize** позволяет изменить размер блочного устройства. Ее можно выполнить, если LVM использует физический том.

4.2.5. Удаление физических томов

Команда **pvremove** удаляет устройство, в котором больше нет необходимости, заполняя метаданные LVM физического тома нулями.

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 Раздел 4.3.6, «Удаление физических томов из группы».

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

4.3. АДМИНИСТРИРОВАНИЕ ГРУППЫ ТОМОВ

В данной секции описываются команды, используемые при администрировании групп томов.

4.3.1. Создание групп томов

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.

Следующая команда создаст группу томов с именем **vg1**, включающую в себя физические тома **/dev/sdd1** и **/dev/sde1**.

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

Если для создания группы томов используются физические тома, их дисковое пространство по умолчанию подразделяется на экстенты размером 4 Мб. Этот размер и определяет минимальную величину, используемую при увеличении или уменьшении размера логического тома. Число экстентов не оказывает влияния на эффективность операций ввода/ вывода для логического тома.

Размер экстента можно задать с помощью опции **-s** команды **vgcreate**. Также можно ограничить число физических или логических томов в группе — для этого используйте опции **-p** и **-l**.

По умолчанию группа томов осуществляет выделение физических экстентов в согласно стандартным правилам, например, чередующиеся секции не будут располагаться на одном физическом томе. Стандартная политика обозначена как **normal**. Опция **--alloc** команды **vgcreate** позволяет изменить стандартную политику на **contiguous**, **anywhere** или **cling**.

Политика **contiguous** требует, чтобы новые экстенты размещались рядом с уже существующими экстентами. Если в наличии достаточно свободных экстентов, чтобы удовлетворить запрос выделения пространства, а политика **normal** не может их использовать, тогда можно применить политику **anywhere**. Экстенты будут заняты, даже если при этом чередующиеся сегменты будут включены в один том, тем самым оказав негативный эффект на производительность. Команда **vgchange** позволяет изменить политику.

В целом, любой вид политики, отличающийся от обычной (**normal**), используется только в особых, нестандартных случаях.

Группы томов LVM и их логические тома включены в дерево каталогов специальных файлов устройств в **/dev**. Путь к ним:

```
/dev/vg/lv/
```

Например, если вы создали две группы томов **myvg1** и **myvg2** с томами **lvo1**, **lvo2** и **lvo3** в составе каждой группы, то при этом будет создано шесть специальных файлов:

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

Максимальный размер устройства с LVM — 8 Эксабайт для 64-битных процессоров.

4.3.2. Создание групп томов в кластере

Группы томов в кластерном окружении можно создать с помощью команды **vgcreate**.

По умолчанию группы томов, созданные с помощью CLVM в совместном хранилище, видны всем компьютерам, обладающим доступом к этому хранилищу. В принципе, возможно создать локальные группы томов, которые будут видны только одному узлу в кластере, выполнив команду **vgcreate** с опцией **-c n**.

Следующая команда, выполненная в кластерном окружении, создаст локальную группу томов с именем **vg1**, включающую в себя физические тома **/dev/sdd1** и **/dev/sde1**.

```
vgcreate -c n vg1 /dev/sdd1 /dev/sde1
```

You can change whether an existing volume group is local or clustered with the **-c** option of the **vgchange** command, which is described in Раздел 4.3.7, «Изменение параметров группы томов».

Тип группы томов можно проверить с помощью команды **vgs**, которая отобразит атрибут **c**, если том является кластерным. Приведенная далее команда выведет атрибуты групп томов **VolGroup00** и **testvg1**. В этом примере группа **VolGroup00** не является кластерной, а **testvg1** является, о чем свидетельствует атрибут **c** в столбце **Attr**.

```
[root@doc-07]# vgs
  VG            #PV #LV #SN Attr   VSize  VFree
  VolGroup00      1   2   0 wz--n- 19.88G    0
  testvg1         1   1   0 wz--nc 46.00G 8.00M
```

For more information on the **vgs** command, see Раздел 4.3.4, «Отображение групп томов»Раздел 4.9, «Настройка отчетов для LVM», and the **vgs** man page.

4.3.3. Добавление физических томов в группу

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.

Далее приведена команда, добавляющая физический том **/dev/sdf1** в группу **vg1**.

```
vgextend vg1 /dev/sdf1
```

4.3.4. Отображение групп томов

Команды **vgs** и **vgdisplay** позволяют отобразить информацию о группах томов LVM.

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 **vgscan** command, see Раздел 4.3.5, «Поиск групп томов на дисках с целью создания файла кэша».

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 Раздел 4.9, «Настройка отчетов для LVM».

Команда **vgdisplay** отобразит параметры группы томов (размер, экстенты, число физических томов и пр.) в фиксированном формате. Приведенный далее пример демонстрирует вывод команды **vgdisplay** для группы томов **new_vg**. Если группа томов явно не указана, будет отображена информация для всех групп.

```
# vgdisplay new_vg
  --- Volume group ---
  VG Name               new_vg
  System ID
  Format                lvm2
  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
```

4.3.5. Поиск групп томов на дисках с целью создания файла кэша

Команда **vgscan** проверяет все поддерживаемые дисковые устройства в системе на предмет наличия физических томов LVM и групп томов. При этом будет создан файл кэша LVM в **/etc/lvm/.cache**, содержащий перечень текущих устройств LVM.

LVM автоматически выполняет команду **vgscan** при загрузке системы и в процессе работы LVM, например, при вызове команды **vgcreate** или при нахождении несоответствий. Если конфигурация оборудования была изменена, то можно выполнить **vgscan** вручную, чтобы найти новые устройства. Это может понадобиться, например, при добавлении новых дисков в систему по SAN или при горячем подключении нового диска, отмеченного как физический том.

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 Раздел 4.6, «Управление сканированием устройств LVM с помощью фильтров».

Пример вывода команды **vgscan**:

```
# vgscan
  Reading all physical volumes.  This may take a while...
  Found volume group "new_vg" using metadata type lvm2
  Found volume group "officevg" using metadata type lvm2
```

4.3.6. Удаление физических томов из группы

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.

Прежде чем удалить физический том из группы, сначала убедитесь, что он не используется логическими томами. Для этого используйте команду **pvdisplay**.

```
# 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
```

Если физический том все еще используется, необходимо перенести данные на другой том (с помощью команды **pvmove**). И уже затем можно уменьшить размер физического тома, выполнив команду **vgreduce**:

Следующая команда удаляет физический том **/dev/hda1** из группы **my_volume_group**:

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

4.3.7. Изменение параметров группы томов

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 Раздел 4.3.8, «Активация и деактивация групп томов»,

Следующая команда изменяет максимально допустимое число логических томов группы **vg00**, устанавливая его значение в 128:

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

Страница помощи **vgchange**(8) содержит описание настраиваемых параметров групп томов.

4.3.8. Активация и деактивация групп томов

При создании группы томов она активирована по умолчанию. Это означает, что логические тома в составе этой группы доступны и могут изменяться.

Существует множество причин, по которым может понадобиться деактивировать группу томов. Для деактивации и активации группы томов используется опция **-a** (**--available**) команды **vgchange**.

Пример деактивации группы томов с именем **my_volume_group**:

```
vgchange -a n my_volume_group
```

Если кластерная блокировка включена, добавьте "e" для активации/ деактивации группы томов на одном узле или "l" — только на локальном узле. Логические тома с единственным снимком узла всегда активируются отдельно, так как они используются лишь раз на одном узле.

You can deactivate individual logical volumes with the **lvchange** command, as described in Раздел 4.4.4, «Изменение параметров группы логических томов», For information on activating logical volumes on individual nodes in a cluster, see Раздел 4.8, «Активация логических томов на отдельных узлах кластера».

4.3.9. Удаление групп томов

Команда **vgremove** позволяет удалить пустую группу томов.

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

4.3.10. Разделение группы томов

Чтобы разделить физические тома в группе с целью создания новой группы, используется команда **vgsplit**.

Логические тома не могут быть разделены между различными группами. Каждый существующий логический том должен полностью принадлежать физическим томам, входящим в состав либо уже существующей, либо новой группы томов. Для принуждения разбиения можно применить команду **pvmove**.

Следующий пример отделяет новую группу томов **smallvg** от исходной группы **bigvg**.

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

4.3.11. Объединение групп томов

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.

Следующая команда включает неактивную группу томов **my_vg** в активную или неактивную группу **databases** с отображением подробных сведений о процессе выполнения.

```
vgmerge -v databases my_vg
```

4.3.12. Создание резервной копии метаданных группы томов

Резервные копии и архивы метаданных по умолчанию создаются автоматически в случае изменения конфигурации логического тома или группы. Эту возможность можно отключить в файле **lvm.conf**. Резервная копия метаданных обычно сохраняется в **/etc/lvm/backup**, а архивы — в **/etc/lvm/archives**. Команда **vgcfgbackup** позволяет создать резервную копию метаданных в **/etc/lvm/backup** по желанию.

Команда **vgcfrestore** восстанавливает метаданные группы томов из архива и размещает их на всех физических томах в группах.

For an example of using the **vgcfgrestore** command to recover physical volume metadata, see Раздел 6.4, «Восстановление метаданных физического тома».

4.3.13. Переименование группы томов

Для переименования существующей группы томов используется команда **vgrename**.

Обе приведенные далее команды изменяют имя группы **vg02** на **my_volume_group**.

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

```
vgrename vg02 my_volume_group
```

4.3.14. Перенос группы томов в другую систему

С помощью команд **vgexport** и **vgimport** можно осуществлять перемещение целой группы томов LVM между системами.

vgexport запрещает доступ к неактивной группе томов, чтобы можно было отключить ее физические тома. **vgimport** снова разрешает доступ к группе томов.

Последовательность действий при переносе группы томов между системами:

1. Убедитесь, что пользователи не обращаются к файлам в пределах активных томов в группе, затем демонтируйте логические тома.

2. Для того чтобы отметить группу томов как неактивную, используйте команду **vgchange** с аргументом **-a n**.

3. Команда **vgexport** позволяет экспортировать группы томов, что запрещает к доступ к этой группе из системы, откуда она будет удалена.

   После завершения экспортирования группы томов, когда вы запустите команду **pvscan**, будет видно, что физический том принадлежит экспортируемой группе. Пример:

   ```
   [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]
     ...
   ```

   Если система ожидает отключения, то можно отключить диски, заменяющие группу томов, и переподключить их в новую систему.

4. Когда диски уже подключены в новую систему, используйте команду **vgimport** для импорта группы томов, тем самым открывая к ней доступ из системы.

5. Опция **-a y** команды **vgchange** позволяет активировать группу томов.

6. Смонтируйте файловую систему.

4.3.15. Восстановление каталога группы томов

Чтобы воссоздать каталог группы томов и специальные файлы логических томов, используется команда **vgmknodes**. Она проверяет специальные файлы LVM2 в каталоге **/dev**, которые необходимы для работы активных логических томов. При этом будут созданы недостающие специальные файлы и удалены лишние файлы.

Функциональность **vgmknodes** можно включить в **vgscan**, указав опцию **--mknodes**.

4.4. АДМИНИСТРИРОВАНИЕ ЛОГИЧЕСКИХ ТОМОВ

В данной секции будут рассмотрены команды администрирования логических томов.

4.4.1. Создание логических томов

Команда **lvcreate** позволяет создать логический том. Вы можете создать линейные и зеркальные тома, а также тома с чередованием.

Если имя тома не указано, по умолчанию будет использоваться обозначение **lvol#**, где # — внутренний номер логического тома.

Далее приведены примеры создания логических томов трех перечисленных типов.

4.4.1.1. Создание линейных томов

При создании линейного тома из физических томов группы выделяются свободные экстенты. Обычно логические тома используют все доступное пространство. Изменения в логических томах ведут к освобождению или переорганизации пространства физических томов.

Следующая команда создаст логический том размером 10 Гб и в группе **vg1**.

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

Приведенная далее команда создаст линейный логический том **testlv** размером 1500 Мбайт в группе томов **testvg**. При этом будет создано блочное устройство **/dev/testvg/testlv**.

```
lvcreate -L1500 -n testlv testvg
```

Следующая команда создаст логический том **gfslv** размером 50 Гбайт, используя свободные экстенты в логической группе **vg0**.

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

Опция **-l** команды **lvcreate** позволяет задать размер логического тома в экстентах. Также можно указать процент группы томов, используемый для создания логического тома. Далее приведенная команда создаст логический том **mylv**, использующий 60% общего объема группы томов **testvol**.

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

С помощью опции **-l** можно также указать процент свободного пространства группы, которое будет занято логическим томом. Например, команда, создающая логический том **yourlv**, который займет все свободное пространство группы **testvol** будет выглядеть так:

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

You can use **-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 the **lvcreate** command.

Пример команд создания логического тома **mylv**, который займет весь объем группы **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 Раздел 4.3.6, «Удаление

физических томов из группы».

Чтобы создать логический том на основе определенных физических томов, необходимо их указать в командной строке **lvcreate**. Так, следующая команда создаст логический том **testlv** на основе физического тома **/dev/sdg1** в группе **testvg**.

```
lvcreate -L 1500 -ntestlv testvg /dev/sdg1
```

Можно указать, какие экстенты физического тома будут использованы для образования логического тома. В следующем примере будет создан линейный логический том, в состав которого войдут экстенты физического тома **/dev/sda1** с 0 по 25 и **/dev/sdb1** с 50 по 125. Оба физических тома входят в состав группы **testvg**.

```
lvcreate -l 100 -n testlv testvg /dev/sda1:0-25 /dev/sdb1:50-125
```

Следующий пример демонстрирует создание линейного логического тома на основе экстентов с 0 по 25 физического тома **/dev/sda1** и затем продолжит, начиная с экстента 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 Раздел 4.3.1, «Создание групп томов».

4.4.1.2. Создание томов с чередованием

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 Раздел 2.3.2, «Логические тома с чередованием».

При создании логического тома с чередованием число сегментов задается с помощью опции **-i** команды **lvcreate**, что определяет число физических томов, используемых при чередовании. Это значение не может превышать число физических томов в группе (за исключением использования опции **--alloc anywhere**).

Если размеры физических устройств, на основе которых создан логический том, различаются, то максимальный объем тома с чередованием будет определяться размером наименьшего устройства. Например, если для организации чередования используются два физических тома, то максимальный размер логического тома будет равен удвоенному размеру наименьшего устройства. Если же используются три тома, максимальный размер будет равен утроенному размеру наименьшего устройства.

Следующая команда создаст логический том с чередованием на основе двух физических томов в составе группы **vg0**, при этом размер сегмента будет равен 64 Кб. Самому логическому тому будет присвоено имя **gfslv**, его размер будет равен 50 Гб.

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

Так же как и в случае с линейными томами, можно специально указать определенные экстенты физического тома, которые будут заняты сегментами. В приведенном далее примере будет создан том с чередованием (с именем **stripelv**) на основе двух физических томов, размер которого будет составлять 100 экстентов. Новый том будет входить в состав группы **testvg** и занимать секторы 0-50 тома **/dev/sda1** и 50-100 тома **/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
```

4.4.1.3. Создание зеркальных томов

При создании зеркального логического тома необходимо указать число копий, для чего служит опция **-m** команды **lvcreate**. Так, если указать **-m1**, будет создано одно зеркало, что, в сущности, создаст две копии данных в файловой системе — линейный логический том и его копию. Аналогичным образом, если указать **-m2**, будут созданы два зеркала (всего три копии).

Ниже приведен пример создания зеркального логического тома размером 50 Гб с одним зеркалом. Ему будет присвоено имя **mirrorlv**, пространство для его создания будет выделено из группы **vg0**.

```
lvcreate -L 50G -m1 -n gfslv vg0
```

Зеркало LVM разбивает копируемое устройство на регионы, размер которых по умолчанию равен 512 Кб. Чтобы задать другой размер (в мегабайтах), используйте опцию **-R**. LVM поддерживает краткий журнал синхронизации регионов с зеркалами. По умолчанию журнал хранится на диске, поэтому он не теряется при перезагрузке. Если же вы хотите, чтобы журнал находился в памяти, используйте опцию **--corelog**, что отменяет необходимость в устройстве журналирования, но в то же время требует, чтобы зеркало полностью синхронизировалось при каждой перезагрузке.

Приведенная ниже команда создаст логический том **ondiskmirvol** с одним зеркалом в группе **bigvg**. Размер тома равен 12 Мбайт, а журнал зеркала хранится в памяти.

```
# lvcreate -L 12MB -m1 --corelog -n ondiskmirvol bigvg
  Logical volume "ondiskmirvol" created
```

Журнал зеркала будет создан на отдельном устройстве. Возможно создание журнала на том же устройстве, что и секция зеркала, — для этого служит ключ **--alloc anywhere** команды **vgcreate**. Это может отрицательно сказаться на производительности, но позволит создать зеркало, даже если в основу положено всего лишь два устройства.

Ниже приведен пример создания зеркального логического тома **mirrorlv** размером 50 Мбайт с одним зеркалом на основе группы томов **vg0**. При этом журнал зеркала расположен на том же устройстве, что и составляющая зеркала. В этом примере группа томов **vg0** состоит из двух устройств.

```
lvcreate -L 500M -m1 -n mirrorlv -alloc anywhere vg0
```

В момент создания зеркала выполняется синхронизация регионов зеркала. Если компоненты зеркала достаточно велики, процесс синхронизации может занять некоторое время. Если вы создаете новое зеркало, синхронизация которого необязательна, укажите опцию **nosync**.

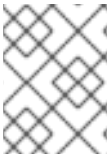
Можно задать, на каких устройствах будут сохраняться журналы и какие экстенты устройств будут использоваться зеркалом. Чтобы ограничить журналирование определенным диском, укажите ТОЛЬКО один экстент на том диске, где должен располагаться журнал. LVM игнорирует порядок, в котором перечислены устройства. Если в списке присутствуют физические устройства, то только они и будут использоваться для выделения пространства; уже занятые физические будут проигнорированы.

Далее приведен пример команды создания зеркального логического тома **mirrorlv** размером 500 Мбайт с одним зеркалом. Том будет создан в составе группы **vg0**. Одна часть зеркала будет располагаться на устройстве **/dev/sda1**, вторая — на **/dev/sdb1**, а журнал будет храниться на **/dev/sdc1**.

```
lvcreate -L 500M -m1 -n mirrorlv vg0 /dev/sda1 /dev/sdb1 /dev/sdc1
```

Следующая команда создаст логический том **mirrorlv** размером 500 Мбайт с одним зеркалом. Том будет создан в составе группы **vg0**. Одна часть зеркала будет занимать экстенты с 0 по 499 устройства **/dev/sda1**, вторая — экстенты с 0 по 499 устройства **/dev/sdb1**, а журнал будет храниться на **/dev/sdc1**, начиная с нулевого экстента. Размер экстента равен 1 Мбайт. Если заданные физические экстенты уже заняты, они будут просто проигнорированы.

```
lvcreate -L 500M -m1 -n mirrorlv vg0 /dev/sda1:0-499 /dev/sdb1:0-499
/dev/sdc1:0
```



ПРИМЕЧАНИЕ

Начиная с версии RHEL 5.3, зеркальные логические тома поддерживаются на уровне кластера.

4.4.1.4. Изменение конфигурации зеркальных томов

С помощью команды **lvconvert** можно преобразовать тип логического тома из зеркального в линейный или из линейного в зеркальный. Эту команду также можно использовать для изменения параметров зеркала существующих логических томов (например, **corelog**).

При преобразовании логического тома в зеркальный, в сущности, вы просто создаете составляющие зеркала для уже существующего тома. Это значит, что группа томов должна иметь достаточно пространства и устройств для организации зеркал и хранения журнала.

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 Раздел 6.3, «Восстановление после сбоя зеркала LVM».

Следующая команда преобразует линейный логический том **vg00/lvol1** в зеркальный.

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

Команда преобразования зеркального логического тома **vg00/lvol1** в линейный с удалением зеркального компонента будет выглядеть так:

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

4.4.2. Постоянные номера устройств

Производительность некоторых приложений зависит от того, не изменяется ли номер активного блочного устройства (основной или второстепенный). Их можно задать с помощью команд **lvcreate** и **lvchange**:

```
--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.

Если вы экспортируете файловую систему по NFS, то указание в файле экспорта параметра **fsid** поможет избежать необходимости в обеспечении постоянства номера устройства в пределах LVM.

4.4.3. Изменение размера логических томов

С помощью команды **lvreduce** можно изменить размер логического тома. Если логический том содержит файловую систему, сначала необходимо уменьшить ее размер (или использовать графический интерфейс LVM, чтобы не делать это вручную), чтобы размер логического тома был эквивалентен будущему размеру файловой системы.

Следующая команда уменьшает размер логического тома **lvol1**, входящего в состав группы **vg00**, на три логических экстента.

```
lvreduce -l -3 vg00/lvol1
```

4.4.4. Изменение параметров группы логических томов

С помощью команды **lvchange** можно изменить параметры логического тома. Страница помощи **lvchange**(8) содержит перечень доступных параметров.

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 Раздел 4.3.7, «Изменение параметров группы томов».

Следующая команда ограничивает доступ к тому **lvol1** в группе **vg00** только чтением.

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

4.4.5. Переименование логических томов

Чтобы переименовать существующий логический том, используйте команду **lvrename**.

Обе приведенные ниже команды изменят имя логического тома **lvold** в группе **vg02** на **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 Раздел 4.8, «Активация логических томов на отдельных узлах кластера».

4.4.6. Удаление логических томов

Удалить неактивный логический том можно с помощью команды **lvremove**. Сначала демонтируйте логический том (**umount**). В кластерном окружении необходимо деактивировать том перед его удалением.

Необходимо сначала демонтировать том.

Приведенная далее команда удалит логический том **/dev/testvg/testlv** из группы **testvg**. В этом примере логический том не был деактивирован заранее.

```
[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
```

Том можно деактивировать отдельно с помощью команды **lvchange -an**, тогда предупреждение об удалении активного логического тома не будет отображено.

4.4.7. Отображение информации о логических томах

Для отображения сведений о логических томах используются команды **lvs**, **lvdisplay** и **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 Раздел 4.9, «Настройка отчетов для LVM».

Вывод команды **lvdisplay** будет содержать информацию о томе в фиксированном формате.

Следующая команда отобразит атрибуты тома **lvol2**, входящего в состав группы **vg00**. Если существуют снимки исходного логического тома, то команда также отобразит перечень снимков и их статус (активен/ неактивен).

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

lvscan отобразит все логические тома в системе. Пример:

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

4.4.8. Увеличение размера логических томов

Команда **lvextend** позволяет увеличить размер логического тома.

После увеличения размера тома потребуется увеличить размер соответствующей файловой системы.

Можно задать точный объем наращивания или точный размер тома уже после увеличения.

Пример наращивания логического тома **/dev/myvg/homevol** до 12 Гбайт:

```
# 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
```

Следующая команда добавит еще один гигабайт к **/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
```

Опция **-l** команды **lvextend** позволяет задать точное число экстентов, на которое следует увеличить размер. Или же можно указать процентную часть группы томов или свободного места в группе. Приведенная далее команда увеличит размер логического тома **testlv** так, чтобы он заполнял все еще не выделенное место в группе **myvg**.

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

Завершив наращивание, необходимо увеличить размер файловой системы.

Большинство стандартных утилит изменения размера файловой системы нарастят файловую систему так, чтобы ее размер соответствовал размеру логического тома по умолчанию.

4.4.9. Увеличение размера тома, использующего чередование

Чтобы увеличить размер логического тома, использующего чередование, необходимо убедиться в наличии достаточного объема пространства на физических томах в его основе. К примеру, в случае двухстороннего чередования, в котором занята целая группа томов, добавление нового физического тома в группу не позволит увеличить размер сегмента чередования. В этом случае придется добавить в группу как минимум два физических тома.

Представим себе группу **vg**, созданную на основе двух физических томов. Вывод **vgs** для этой группы:

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

Можно занять все доступное в группе место при организации чередования.

```
# 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)
```

Обратите внимание, что в группе не осталось свободного пространства.

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

Следующая команда добавит новый физический том в группу, тем самым увеличив ее объем на 135 Гбайт.

```
# 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
```

Следует отметить, что увеличить объем тома с чередованием до максимального размера группы невозможно, так как для организации чередования необходимы два физических устройства.

```
# 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
```

Чтобы увеличить размер логического тома, использующего чередование, добавьте новый физический том, а уже затем нарастите объем логического тома. В приведенном примере добавление двух физических томов позволит увеличить объем логического тома 5A до максимального объема всей группы томов.

```
# vgextend vg /dev/sdd1
  Volume group "vg" successfully extended
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  vg     4   1   0 wz--n- 542.62G 271.31G
# 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
```

Если в вашем распоряжении нет необходимого числа физических устройств, все же можно увеличить объем логического тома, не используя при этом чередование. При наращивании тома по умолчанию используются параметры чередования последнего сегмента существующего логического тома, что, в принципе, можно переопределить. Следующий пример расширяет существующий логический том так, чтобы в случае неудачи команды **lvextend** использовалось все свободное пространство.

```
# 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
# lvextend -i1 -l+100%FREE vg/stripe1
```

4.4.10. Уменьшение размера логических томов

При уменьшении размера логического тома сначала надо размонтировать файловую систему, а уже затем с помощью команды **lvreduce** сжать том. Завершив, смонтируйте файловую систему заново.

ПРЕДУПРЕЖДЕНИЕ

Очень важно сначала уменьшить размер файловой системы до уменьшения размера тома. В противном случае данные могут быть потеряны.

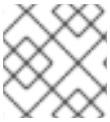
Уменьшив размер логического тома, освобожденное пространство может выделено другим томам в группе.

В следующем примере размер тома **lvol1** в группе **vg00** будет сокращен на 3 логических экстента.

```
lvreduce -l -3 vg00/lvol1
```

4.5. СОЗДАНИЕ ТОМОВ-СНИМКОВ

Для того чтобы создать том-снимок, используйте команду **lvcreate** с опцией **-s**.



ПРИМЕЧАНИЕ

Снимки LVM не поддерживаются узлами в пределах кластера.

Since LVM snapshots are not cluster-aware, they require exclusive access to a volume. For information on activating logical volumes on individual nodes in a cluster, see Раздел 4.8, «Активация логических томов на отдельных узлах кластера».

Приведенная ниже команда создаст снимок исходного тома **/dev/vg00/lvol1** размером 100 Мбайт с именем **/dev/vg00/snap**. Если исходный том содержит файловую систему, то можно смонтировать снимок в любую точку, тем самым получив доступ к содержимому файловой системы для создания резервной копии, в то время как исходная файловая система будет продолжать получать обновления.

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

Выполнив команду **lvdisplay** для исходных логических томов, можно отобразить список всех снимков и их статус (активный или неактивный).

В следующем примере вывод будет отображать состояние логического тома **/dev/new_vg/lvol0**, для которого ранее был создан снимок **/dev/new_vg/newvgsnap**.

```
# 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
```

Команда **lvs** по умолчанию отображает исходный том и процентную часть тома-снимка. Следующий пример демонстрирует стандартный вывод команды **lvs** для системы с логическим томом **/dev/new_vg/lvol0** с соотвествующим ему томом-снимком **/dev/new_vg/newvgsnap**.

```
# 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
```

ПРИМЕЧАНИЕ

Так как размер снимка увеличивается с изменением исходного тома, достаточно важно следить за заполнением снимка. Для этого можно использовать команду **lvs**. При заполнении снимка на 100% он будет потерян, так как запись в неизменяемые участки исходного ресурса разрушит сам снимок.

4.6. УПРАВЛЕНИЕ СКАНИРОВАНИЕМ УСТРОЙСТВ LVM С ПОМОЩЬЮ ФИЛЬТРОВ

При запуске выполняется команда **vgscan**, которая выполнит поиск меток LVM на блочных устройствах системы с целью определения того, какие из них представляют собой физические тома, а также получения метаданных и создания списков групп томов. Имена физических томов хранятся в файле **/etc/lvm/.cache** на каждом узле в системе. Последующие команды будут обращаться к этому файлу, при этом не будет необходимости в повторном сканировании.

С помощью фильтров, определяемых в **lvm.conf**, можно управлять тем, какие устройства будут сканироваться. Фильтры представляют собой набор регулярных выражений, применяемых к именам устройств в каталоге **/dev** с целью разрешения или запрета определения блочного устройства.

Приведенные ниже примеры демонстрируют использование фильтров. Следует отметить, что некоторые примеры не являются лучшими решениями, так как регулярные выражения свободно сопоставляются с полными путями. Например, **a/.*loop.*/** соответствует не только **a/loop/**, но и **/dev/solooperation/lvol1**.

Следующий фильтр добавит все найденные устройства, что является поведением по умолчанию в случае, если фильтры не заданы.

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

Следующий фильтр исключит устройство CD-ROM, если оно не содержит диск, чтобы избежать задержек.

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

Фильтр добавления всех петлевых устройств и удаления всех блочных устройств будет выглядеть так:

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

Пример фильтра, добавляющего все IDE- и петлевые устройства и удаляющего все остальные блочные устройства:

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

Пример фильтра, добавляющего только восьмой раздел на первом диске IDE и удаляющего все остальные блочные устройства:

```
filter = [ "a|^/dev/hda8$|", "r/.*/" ]
```

For more information on the **lvm.conf** file, see Приложение B, *Файлы конфигурации LVM* and the **lvm.conf**(5) man page.

4.7. ПЕРЕМЕЩЕНИЕ ДАННЫХ В АКТИВНОЙ СИСТЕМЕ

Команда **pvmove** позволяет перемещать данные в активной системе.

pvmove разделяет перемещаемые данные на секции и создает временное зеркало для переноса каждой секции. Страница помощи **pvmove**(8) содержит более подробную информацию.

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 Раздел 4.8, «Активация логических томов на отдельных узлах кластера».

Следующая команда переместит все выделенное пространство с физического тома **/dev/sdc1** на другие тома в группе:

```
pvmove /dev/sdc1
```

Команда перемещения экстентов логического тома **MyLV** будет выглядеть так:

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

Поскольку процесс переноса может быть достаточно длительным, можно его запустить на фоне. Приведенная далее команда переместит все выделенные на физическом томе **/dev/sdc1** сегменты на **/dev/sdf1** в фоновом режиме.

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

Следующая команда отобразит индикатор прогресса перемещения (в процентах) с пятисекундным интервалом:

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

4.8. АКТИВАЦИЯ ЛОГИЧЕСКИХ ТОМОВ НА ОТДЕЛЬНЫХ УЗЛАХ КЛАСТЕРА

Если LVM используется в кластерном окружении, то иногда может понадобиться активировать логические тома для одного узла. Например, команда **pvmove** не работает с кластерами, ей необходим эксклюзивный доступ к тóму, так же как и снимкам LVM.

Для активации логических томов для отдельного узла используется команда **lvchange -aey**. Или же с помощью **lvchange -aly** можно активировать логические тома для одного тома, но не эксклюзивно.

You can also activate logical volumes on individual nodes by using LVM tags, which are described in Приложение C, *Теги объектов LVM*. You can also specify activation of nodes in the configuration file, which is described in Приложение B, *Файлы конфигурации LVM*.

4.9. НАСТРОЙКА ОТЧЕТОВ ДЛЯ LVM

С помощью команд **pvs**, **lvs**, **vgs** можно создавать подробные отчеты о состоянии объектов LVM. Каждая строка в таком отчете содержит информацию об одном объекте — набор полей его параметров. Вывод можно отфильтровать по физическим томам, по группе томов, по логическим томам, сегментам физических или логических томов.

Последующие секции включают следующие обзоры:

- Обзор аргументов команд, которые используются для настройки формата генерируемого отчета.
- Перечень полей, которые можно выбрать для каждого объекта LVM.
- Обзор аргументов команд, которые используются для сортировки генерируемого отчета.
- Инструкции по указанию единиц в выводе отчета.

4.9.1. Настройка формата

Независимо от того, используете ли вы **pvs**, **lvs** или **vgs**, выбранная команда отображает стандартный набор полей, что можно переопределить с помощью различных опций.

- Опция **-o** позволяет выбрать поля для отображения. Например, стандартный вывод команды **pvs**, отображающий сведения о физических томах, выглядит так:

```
# 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
```

  Следующая команда отобразит только имя и размер физического тома.

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

- Дополнительное поле можно добавить с помощью символа "+", который используется в комбинации с опцией "-o".

Пример стандартного отображения полей физического тома с его UUID:

```
# 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-X08M-mcpsVe
  /dev/sdd1  new_vg lvm2 a-   17.14G 17.14G yvfvZK-Cf31-j75k-dECm-
0RZ3-0dGW-UqkCS
```

- Добавление **-v** позволяет включить другие поля. Например, **pvs -v** отобразит не только стандартные поля, но и **DevSize** и **PV UUID**.

```
# 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
```

- **--noheadings** спрячет строку заголовков, что используется при создании сценариев.

  Следующий пример использует аргумент **--noheadings** в комбинации с **pv_name** для отображения списка всех физических томов.

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

- **--separator** ***разделитель*** позволяет отделить поля друг от друга. Используется при обработке вывода с помощью **grep**.

  В следующем примере поля вывода команды **pvs** разделены знаком равенства.

```
# 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
```

  Чтобы обеспечить выравнивание полей при использовании разделителя, можно дополнительно указать **--aligned**.

```
# 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 Раздел 6.2, «Отображение информации о сбойных устройствах».

Страницы помощи **pvs**(8), **vgs**(8) и **lvs**(8) содержат полный перечень опций.

Поля группы томов могут быть смешаны с полями физического или логического тома (или их сегментов), но поля физического тома не могут быть смешаны с полями логического тома. Например, следующая команда отобразит по одной строке для каждого физического тома:

```
# 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
```

4.9.2. Выбор объектов

В данной секции рассматриваются таблицы, отображающие информацию, получаемую с помощью команд **pvs**, **vgs**, **lvs**.

Префикс названия поля может быть опущен, если он соответствует стандартному имени, используемому командой. Например, при указании **name** с командой **pv_name** подразумевается **pv_name**, а с **vgs** — **vg_name**.

Выполнение следующей команды эквивалентно **pvs -o pv_free**.

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

4.9.2.1. Команда pvs

Таблица 4.1, «Поля вывода команды pvs» 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.

Таблица 4.1. Поля вывода команды pvs

Аргумент	Заголовок	Описание
dev_size	DevSize	Размер устройства в основе физического тома
pe_start	1st PE	Смещение начала первого физического экстента физического устройства
pv_attr	Attr	Статус физического тома: (a)llocatable или e(x)ported
pv_fmt	Fmt	Формат метаданных физического тома (**lvm2** или **lvm1**)
pv_free	PFree	Свободное место в пределах физического тома

Аргумент	Заголовок	Описание
pv_name	PV	Имя физического тома
pv_pe_alloc_count	Alloc	Число занятых физических экстентов
pv_pe_count	PE	Число физических экстентов
pvseg_size	SSize	Размер сегмента физического тома
pvseg_start	Start	Начальный физический экстент сегмента физического тома
pv_size	PSize	Размер физического тома
pv_tags	PV Tags	Теги физического тома
pv_used	Used	Занятый объем физического тома
pv_uuid	PV UUID	UUID физического тома

По умолчанию **pvs** отображает поля **pv_name**, **vg_name**, **pv_fmt**, **pv_attr**, **pv_size**, **pv_free**, отсортированные по **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
```

Опция **-v** команды **pvs** добавит поля **dev_size**, **pv_uuid**.

```
# 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 17.13G  17.14G yvfvZK-Cf31-j75k-dECm-
0RZ3-0dGW-tUqkCS
```

Аргумент **--segments** команды **pvs** позволяет отобразить информацию о каждом сегменте физического тома. Сегмент представляет собой набор экстентов. Возможность просмотра сегментов может помочь при определении фрагментации логического тома.

pvs --segments по умолчанию отобразит поля **pv_name**, **vg_name**, **pv_fmt**, **pv_attr**, **pv_size**, **pv_free**, **pvseg_start**, **pvseg_size**. Вывод будет отсортирован по **pv_name** и **pvseg_size** для каждого физического тома.

```
# pvs --segments
  PV         VG         Fmt  Attr PSize  PFree  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
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G   126    24
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G   150    22
  /dev/sda1  vg         lvm2 a-   17.14G 16.75G   172  4217
  /dev/sdb1  vg         lvm2 a-   17.14G 17.14G     0  4389
  /dev/sdc1  vg         lvm2 a-   17.14G 17.14G     0  4389
  /dev/sdd1  vg         lvm2 a-   17.14G 17.14G     0  4389
  /dev/sde1  vg         lvm2 a-   17.14G 17.14G     0  4389
  /dev/sdf1  vg         lvm2 a-   17.14G 17.14G     0  4389
  /dev/sdg1  vg         lvm2 a-   17.14G 17.14G     0  4389
```

Для просмотра устройств, которые были определены LVM, но не инициализированы в виде физических томов LVM можно использовать команду **pvs -a**.

```
# 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
```

4.9.2.2. Команда vgs

Таблица 4.2, «Поля вывода команды vgs» 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.

Таблица 4.2. Поля вывода команды vgs

Аргумент	Заголовок	Описание
lv_count	#LV	Число логических томов в группе
max_lv	MaxLV	Максимальное допустимое число логических томов в группе (0, если не ограничено)
max_pv	MaxPV	Максимально допустимое число томов в группе (0, если не ограничено)
pv_count	#PV	Число физических томов в основе группы
snap_count	#SN	Число снимков в группе томов
vg_attr	Attr	Статус группы томов. Допустимые значения: (w)riteable, (r)eadonly, resi(z)eable, e(x)ported, (p)artial, (c)lustered
vg_extent_count	#Ext	Число физических экстентов в группе томов
vg_extent_size	Ext	Размер физических экстентов в группе томов
vg_fmt	Fmt	Формат метаданных группы томов (**lvm2** или **lvm1**)
vg_free	VFree	Объем свободного пространства в группе томов
vg_free_count	Free	Число свободных физических экстентов в группе томов
vg_name	VG	Имя группы томов
vg_seqno	Seq	Номер версии группы томов
vg_size	VSize	Размер группы томов
vg_sysid	SYS ID	Системный идентификатор LVM1
vg_tags	VG Tags	LVM-теги группы томов
vg_uuid	VG UUID	UUID группы томов

vgs по умолчанию отображает поля **vg_name**, **pv_count**, **lv_count**, **snap_count**, **vg_attr**, **vg_size**, **vg_free**. Вывод отсортирован по **vg_name**.

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

Опция **-v** команды **vgs** позволяет дополнительно отобразить поля **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
```

4.9.2.3. Команда lvs

Таблица 4.3, «Поля вывода команды lvs» 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.

Таблица 4.3. Поля вывода команды lvs

Аргумент	Заголовок	Описание
chunksize **chunk_size**	Chunk	Размер сегментов снимка тома
copy_percent	Copy%	Процентная часть синхронизации зеркального логического тома. Также используется при перемещении физических экстентов с помощью **pv_move**
devices	Devices	Устройства в основе логического тома: физические устройства, логические тома и начальные физические и логические экстенты
lv_attr	Attr	Статус логического тома. Его составляющие: Бит 1: Тип тома. Допустимые значения: (m)irrored, (M)irrored (без исходной синхронизации), (p)vmove, (s)napshot, (S)napshot (неверный), (v)irtual Бит 2: Разрешения. Допустимые значения: (w)riteable, (r)ead-only Бит 3: Политика выделения. Допустимые значения: (c)ontiguous, (n)ormal, (a)nywhere, (i)nherited. Бит 4: (m)inor Бит 5: Статус. Допустимые значения: (a)ctive, (s)uspended, (I)nvalid snapshot, invalid (S)uspended snapshot, mapped (d)evice (без таблиц) или (i)nactive (устройство с неактивной таблицей). Бит 6: (o)pen (открытое устройство)

Аргумент	Заголовок	Описание
lv_kernel_major	KMaj	Действующий основной номер устройства логического тома (-1, если устройство не активно)
lv_kernel_minor	KMIN	Действующий вспомогательный номер устройства логического тома (-1, если не активно)
lv_major	Maj	Постоянный основной номер устройства логического тома (-1, если не задан)
lv_minor	Min	Постоянный вспомогательный номер устройства логического тома (-1, если не задан)
lv_name	LV	Имя логического тома
lv_size	LSize	Размер логического тома
lv_tags	LV Tags	LVM-теги логического тома
lv_uuid	LV UUID	UUID логического тома
mirror_log	Log	Устройство, на котором размещен журнал зеркал
modules	Modules	Модуль ядра соответствий устройств, необходимый для использования логического тома
move_pv	Move	Исходный физический том временного логического тома, созданного с помощью команды **pvmove**
origin	Origin	Исходное устройство тома-снимка
regionsize **region_size**	Region	Размер сегментов зеркального логического тома
seg_count	#Seg	Число сегментов логического тома
seg_size	SSize	Размер сегментов логического тома
seg_start	Start	Смещение сегментов логического тома
seg_tags	Seg Tags	LVM-теги сегментов логического тома
segtype	Type	Тип сегмента логического тома (например, mirror, striped, linear)

Аргумент	Заголовок	Описание
snap_percent	Snap%	Процентная часть занятого тома-снимка
stripes	#Str	Число сегментов чередования или зеркал логического тома
stripesize **stripe_size**	Stripe	Размер сегментов чередования

По умолчанию команда **lvs** отображает поля **lv_name**, **vg_name**, **lv_attr**, **lv_size**, **origin**, **snap_percent**, **move_pv**, **mirror_log**, **copy_percent**, отсортированные по **vg_name** и **lv_name** в пределах группы.

```
# 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
```

Опция **-v** команды **lvs** позволяет дополнительно отобразить поля **seg_count**, **lv_major**, **lv_minor**, **lv_kernel_major**, **lv_kernel_minor**, **lv_uuid**.

```
# lvs -v
    Finding all logical volumes
  LV         VG     #Seg Attr   LSize  Maj Min KMaj KMin Origin Snap%
Move Copy%  Log LV UUID
  lvol0      new_vg    1 owi-a- 52.00M  -1  -1 253  3
LBy1Tz-sr23-OjsI-LT03-nHLC-y8XW-EhCl78
  newvgsnap1 new_vg    1 swi-a-  8.00M  -1  -1 253  5    lvol0    0.20
1ye1OU-1cIu-o79k-20h2-ZGF0-qCJm-CfbsIx
```

Аргумент **--segments** команды **lvs** позволяет отобразить стандартный набор столбцов с информацией о сегментах. При этом префикс **seg** указывать не обязательно. Так, **lvs --segments** по умолчанию отобразит поля **lv_name**, **vg_name**, **lv_attr**, **stripes**, **segtype**, **seg_size**, отсортированные по **vg_name**, **lv_name** в пределах группы томов и по **seg_start** — в пределах логического тома. Если логические тома фрагментированы, вывод это отобразит.

```
# lvs --segments
  LV       VG         Attr   #Str Type   SSize
  LogVol00 VolGroup00 -wi-ao    1 linear  36.62G
  LogVol01 VolGroup00 -wi-ao    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
```

Дополнительное указание **-v** в команде **lvs --segments** позволяет добавить поля **seg_start**, **stripesize**, **chunksize**.

```
# lvs -v --segments
    Finding 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
```

Следующий пример демонстрирует стандартный вывод команды **lvs** в системе с одним настроенным логическим томом, а также вывод **lvs** с аргументом **segments**.

```
# 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
```

4.9.3. Сортировка отчетов LVM

Обычно вывод команд **lvs**, **vgs**, **pvs** сохраняется и отдельно сортируется, так чтобы столбцы были корректно выровнены. Аргумент **--unbuffered** позволяет отобразить исходный вывод сразу после его генерации.

С помощью опции **-O** можно указать список столбцов, определяющий порядок сортировки. При этом включение этих столбцов в вывод необязательно.

Пример вывода **pvs**, отображающего имена физических томов, их размер и объем свободного пространства:

```
# 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
```

Пример аналогичного вывода, отсортированного по объему свободного пространства:

```
# 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
```

Следующий пример демонстрирует, что нет необходимости в выводе поля, по которому осуществляется сортировка.

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

Чтобы выполнить обратную сортировку, перед полем, которое служит критерием сортировки, надо указать **-**.

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

4.9.4. Указание единиц

Чтобы указать единицы для использования в отчете LVM, используйте аргумент **--units**. Допустимые значения: (b)ytes, (k)ilobytes, (m)egabytes, (g)igabytes, (t)erabytes, (e)xabytes, (p)etabytes и (h)uman-readable. По умолчанию используется упрощенный формат, (h)uman-readable, что можно переопределить в файле **lvm.conf**, изменив параметр **units** в секции **global**.

Пример вывода **pvs**, использующего в качестве единиц мегабайты:

```
# pvs --units m
  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
```

По умолчанию числовые значения кратны 1024, что можно заменить на 1000 посредством указания единиц в верхнем регистре (B, K, M, G, T, H).

Пример стандартного вывода (значения кратны 1024)

```
# 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
```

Вывод, отображающий значения, кратные 1000:

```
#  pvs --units G
  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
```

В качестве единиц также можно использовать сектора ((s)ectors), размер которых 512 б, или даже произвольные единицы.

Пример вывода команды **pvs**, где в качестве единиц используются сектора:

```
# pvs --units s
  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
```

Пример вывода команды **pvs**, где в качестве единиц используются блоки размером 4 Мбайт:

```
# 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
```

ГЛАВА 5. ПРИМЕРЫ КОНФИГУРАЦИИ LVM

Этот раздел содержит простые примеры конфигурации LVM.

5.1. СОЗДАНИЕ ЛОГИЧЕСКОГО ТОМА LVM НА ТРЕХ ДИСКАХ

Данный пример демонстрирует создание логического тома LVM с именем **new_logical_volume** на дисках **/dev/sda1**, **/dev/sdb1**, **/dev/sdc1**.

5.1.1. Создание физических томов

Для использования дисков в группе томов необходимо обозначить их как физические тома LVM.

ПРЕДУПРЕЖДЕНИЕ

Следующая команда удалит все данные на **/dev/sda1**, **/dev/sdb1**, **/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
```

5.1.2. Создание группы томов

Пример команды создания группы томов **/dev/sda1**, **/dev/sdb1**, and **/dev/sdc1**:

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

Отобразить атрибуты новой группы томов можно с помощью команды **vgs**.

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

5.1.3. Создание логического тома

В следующем примере будет создан логический том **new_logical_volume** размером 2 Гб из группы томов **new_vol_group**.

```
[root@tng3-1 ~]# lvcreate -L2G -n new_logical_volume new_vol_group
  Logical volume "new_logical_volume" created
```

5.1.4. Создание файловой системы

Пример команды создания файловой системы GFS для логического тома:

```
[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
```

Следующие команды смонтируют логический том и отобразят информацию об использовании дискового пространства.

```
[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
```

5.2. СОЗДАНИЕ ЛОГИЧЕСКОГО ТОМА С ЧЕРЕДОВАНИЕМ

В этом примере будет создан логический том **new_vol_group**, данные которого чередуются между дисками **/dev/sda1**, **/dev/sdb1**, **/dev/sdc1**.

5.2.1. Создание физических томов

Отметьте диски в составе группы томов, которые вы планируете использовать, как физические тома LVM.

ПРЕДУПРЕЖДЕНИЕ

Следующая команда удалит все данные на **/dev/sda1**, **/dev/sdb1**, **/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
```

5.2.2. Создание группы томов

Пример команды создания группы томов **striped_vol_group**:

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

Отобразить атрибуты новой группы томов можно с помощью команды **vgs**.

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

5.2.3. Создание логического тома

Следующая команда создаст логический том **striped_logical_volume** размером 2 Гбайт из группы томов **striped_vol_group**. При этом он будет содержать 3 сегмента чередования размером 4 Кбайт.

```
[root@tng3-1 ~]# lvcreate -i3 -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
```

5.2.4. Создание файловой системы

Пример команды создания файловой системы GFS для логического тома:

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

Следующие команды смонтируют логический том и отобразят информацию об использовании дискового пространства.

```
[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-LogVol00
```

```
                      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
```

5.3. РАЗБИЕНИЕ ГРУППЫ ТОМОВ

Представим, что существующая группа томов включает три физических тома. Если имеется достаточно незанятого пространства, то новая группа может быть добавлена без необходимости добавления новых дисков.

Изначально логический том **mylv** создается из группы **myvol**, которая включает 3 логических тома: **/dev/sda1**, **/dev/sdb1**, **/dev/sdc1**.

После завершения группа **myvg** будет состоять из **/dev/sda1** и **/dev/sdb1**, а **/dev/sdc1** будет включен в состав второй группы, **yourvg**.

5.3.1. Определение наличия свободного пространства

С помощью команды **pvscan** можно определить объем незанятого пространства в группе томов.

```
[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   ]
```

5.3.2. Перемещение данных

Представим, что надо перенести все занятые физические экстенты **/dev/sdc1** на **/dev/sdb1**. Это можно сделать с помощью команды **pvmove**. Процесс может занять достаточно долго времени.

```
[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%
```

После перемещения данных можно убедиться, что все пространство **/dev/sdc1** теперь свободно.

```
[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   ]
```

5.3.3. Разделение группы томов

С помощью команды **vgsplit** разделим группу **myvg** и создадим новую группу томов **yourvg**.

Прежде чем приступить к разделению группы томов, сначала необходимо убедиться, что файловая система не смонтирована, а затем деактивировать логический том.

lvchange и **vgchange** позволяют деактивировать логический том. Приведенная команда деактивирует том **mylv**, затем выделит пространство для группы **yourvg** из **myvg** и переместит физический том **/dev/sdc1** в новую группу **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"
```

Просмотреть атрибуты обоих групп можно с помощью **vgs**.

```
[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
```

5.3.4. Создание логического тома

Создав новую группу томов, можно создать логический том **yourlv**:

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

5.3.5. Создание файловой системы и монтирование нового логического тома

Теперь для нового логического тома можно создать и смонтировать файловую систему.

```
[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
```

5.3.6. Активация и монтирование исходного логического тома

Поскольку логический том **mylv** был деактивирован, его нужно активировать, а затем смонтировать.

```
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
```

5.4. УДАЛЕНИЕ ДИСКА ИЗ ЛОГИЧЕСКОГО ТОМА

Данный пример демонстрирует, как можно удалить диск из существующего логического тома либо с целью его замены, либо при его включении в другой том. Чтобы удалить диск, сначала необходимо переместить экстенты на другой диск или набор дисков.

5.4.1. Перенос экстентов на существующий логический том

В этом примере логический том расположен на четырех физических томах группы **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
```

Предположим, что требуется перенести экстенты с **/dev/sdb1**, чтобы его можно было затем удалить из группы томов.

Если на других физических томах в группе достаточно свободных экстентов, на удаляемом устройстве можно выполнить команду **pvmove** без аргументов, чтобы распределить свободные экстенты между другими устройствами.

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

После завершения **pvmove** распределение экстентов будет выглядеть так:

```
[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
```

С помощью команды **vgreduce** можно удалить том **/dev/sdb1** из группы.

```
[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
```

Теперь диск можно физически удалить из системы или использовать его для других целей.

5.4.2. Перенос экстентов на новый диск

В следующем примере логический том расположен на трех физических томах группы **myvg**:

```
[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
```

Допустим, мы хотим переместить экстенты **/dev/sdb1** на новое устройство **/dev/sdd1**.

5.4.2.1. Создание физического тома

Пример создания нового физического тома на основе **/dev/sdd1**:

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

5.4.2.2. Добавление нового физического тома в группу томов

Добавим **/dev/sdd1** в существующую группу **myvg**.

```
[root@tng3-1 ~]# vgextend myvg /dev/sdd1
  Volume group "myvg" successfully extended
[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
  /dev/sdd1   myvg lvm2 a-   17.15G 17.15G     0
```

5.4.2.3. Перемещение данных

С помощью команды **pvmove** переместим данные **/dev/sdb1** на **/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
  /dev/sdd1   myvg lvm2 a-   17.15G 15.15G  2.00G
```

5.4.2.4. Удаление физического тома из группы томов

Завершив перенос данных с **/dev/sdb1**, можно удалить том из группы.

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

Теперь можно переназначить диск другой группе томов или удалить его из системы.

ГЛАВА 6. РЕШЕНИЕ ПРОБЛЕМ LVM

Этот раздел содержит инструкции по решению различных проблем с LVM.

6.1. ДИАГНОСТИКА

Если команда не функционирует как ожидается, осуществить сбор информации можно следующим образом:

- Использовать опции **-v**, **-vv**, **-vvv**, **-vvvv** для отображения подробного вывода исполняемой команды.
- 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.
- Выполните команду **lvmdump**. Ее вывод будет содержать информацию, которая может помочь при определении проблем (см. страницу помощи **lvmdump**(8)).
- Команды **lvs -v**, **pvs -a**, **dmsetup info -c** позволяют получить дополнительную информацию о системе.
- Просмотрите последнюю резервную копию метаданных в **/etc/lvm/backup** и архивы в **/etc/lvm/archive**.
- Команда **lvm dumpconfig** позволит проверить текущую информацию конфигурации.
- Файл **.cache** в **/etc/lvm** должен включать запись, перечисляющую устройства, на которых расположены физические тома.

6.2. ОТОБРАЖЕНИЕ ИНФОРМАЦИИ О СБОЙНЫХ УСТРОЙСТВАХ

Опция **-P** команд **lvs** и **vgs** позволяет отобразить информацию о сбойном томе. К примеру, если в составе группы **vg** перестало функционировать одно устройство, следующая команда об этом сообщит:

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

При указании опции **-P** команды **vgs** можно отобразить информацию о сбойном устройстве, даже если невозможно использовать группу томов, в состав которой оно входит.

```
[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)
```

В этом примере сбой устройства привел к сбою линейного тома и тома с чередованием в группе. **lvs** без опции **-P** отобразит следующее:

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

При указании опции **-P** вывод будет включать список сбойных логических томов.

```
[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)
```

Следующие примеры демонстрируют вывод **pvs** и **lvs**, использующих опцию **-P**, в случае сбоя компонента зеркального логического тома.

```
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/sdd1(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/sdb1(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),my_mirror_mimage_1(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)
```

6.3. ВОССТАНОВЛЕНИЕ ПОСЛЕ СБОЯ ЗЕРКАЛА LVM

В этой секции рассматривается пример того, как осуществляется восстановление в случае сбоя одного компонента зеркального логического тома как следствие сбоя физического тома. Тогда LVM преобразует зеркальный том в линейный, сохранив функциональность, но уже без избыточности. Затем можно добавить новое дисковое устройство и пересоздать зеркало.

Следующая команда создаст физические тома, которые будут использоваться при организации зеркала.

```
[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
```

Примеры команд создания группы томов **vg** и зеркального тома **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
```

С помощью команды **lvs** можно осуществить проверку структуры зеркального тома, устройств в его основе и журнала зеркала. Обратите внимание, что в первом примере (см. ниже) синхронизация зеркала еще не завершена, поэтому сначала надо дождаться, пока индикатор прогресса **Copy%** не отобразит 100.00.

```
[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)
```

В этом примере предполагается, что произошел сбой основного компонента зеркала **/dev/sda1**. Все попытки записи в зеркальный том приводят к определению сбойного зеркала, в случае чего LVM преобразует зеркало в один линейный том. Команда **dd** начнет преобразование.

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

Чтобы убедиться, что устройство теперь является линейным, используйте команду **lvs**. При этом будет отображена ошибка ввода/ вывода, так как присуствует сбойный диск.

```
[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)
```

Можно продолжать использовать том, но уже как линейный.

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.

```
[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           lvm2 [603.94 GB]
  PV /dev/sda2           lvm2 [603.94 GB]
  Total: 16 [2.11 TB] / in use: 14 [949.65 GB] / in no VG: 2 [1.18 TB]
```

Далее нарастите размер исходной группы томов, добавив новый физический том.

```
[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   ]
```

Преобразуйте линейный том в зеркальный.

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

Чтобы убедиться, что зеркало было восстановлено, используйте команду **lvs**.

```
[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
68.62 groupfs_mimage_0(0),groupfs_mimage_1(0)
  [groupfs_mimage_0] vg   iwi-ao 752.00M
/dev/sdb1(0)
  [groupfs_mimage_1] vg   iwi-ao 752.00M
/dev/sda1(0)
  [groupfs_mlog]     vg   lwi-ao   4.00M
/dev/sdc1(0)
```

6.4. ВОССТАНОВЛЕНИЕ МЕТАДАННЫХ ФИЗИЧЕСКОГО ТОМА

Представим, что область метаданных группы томов была случайно перезаписана или разрушена. В этом случае будет отображена ошибка, сообщающая о проблеме в области метаданных или о неудаче при попытке нахождения физического тома с заданным UUID. Чтобы восстановить данные, можно попробовать заново переписать область метаданных, включив прежний UUID потерянных метаданных.

ПРЕДУПРЕЖДЕНИЕ

Не следует этого выполнять, если логический том активно используется, так как в случае указания неверного UUID все данные будут потеряны.

Пример вывода в случае потери или разрушения области метаданных:

```
[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.
  ...
```

Потерянный UUID можно найти в последней секции архивных метаданных файла ***ИмяГруппыТомов_xxxx*.vg** в каталоге **/etc/lvm/archive**.

Другой способ нахождения идентификатора UUID предполагает деактивацию тома и установку опции **-P** (**partial**).

```
[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'.
  ...
```

С помощью опций **--uuid** и **--restorefile** команды **pvcreate** можно восстановить физический том. В приведенном далее примере устройству **/dev/sdh1** соответствует UUID **FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk**. Сама команда восстановит информацию метаданных, получив содержимое из последнего успешного архива метаданных **VG_00050.vg**. **pvcreate** перезапишет только области метаданных, что не окажет эффекта на существующие данные.

```
[root@link-07 backup]# pvcreate --uuid "FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-
NYM5Sk" --restorefile /etc/lvm/archive/VG_00050.vg /dev/sdh1
  Physical volume "/dev/sdh1" successfully created
```

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

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

После этого можно вывести информацию о логических томах.

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

Следующие команды активируют тома и отображают все активные тома.

```
[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
  stripe VG   -wi-a- 300.00G                                /dev/sdh1
(0),/dev/sda1(0)
  stripe VG   -wi-a- 300.00G                                /dev/sdh1
(34728),/dev/sdb1(0)
```

Эта команда может восстановить физический том, если хранимые на диске метаданные были полностью перезаписаны. Если область перезаписи вышла за пределы области метаданных, то возможна потеря данных. Попробуйте восстановить данные с помощью **fsck**.

6.5. ЗАМЕНА ОТСУТСТВУЮЩЕГО ФИЗИЧЕСКОГО ТОМА

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 Раздел 6.4, «Восстановление метаданных физического тома». 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.

6.6. УДАЛЕНИЕ ПОТЕРЯННЫХ ФИЗИЧЕСКИХ ТОМОВ ИЗ ГРУППЫ

В случае потери физического тома можно активировать оставшиеся физические тома в группе с помощью опции **--partial** команды **vgchange**. Опция **--removemissing** команды **vgreduce** позволяет удалить из группы все логические тома, использующие этот физический том.

Рекомендуется сначала выполнить команду **vgreduce** с опцией **--test**, чтобы знать, что именно будет удалено.

Аналогично большинству действий LVM, результат команды **vgreduce** обратим, если сразу же выполнить **vgcfgrestore** для восстановления метаданных. Например, если вы выполнили **vgreduce** с аргументом **--removemissing**, не указав при этом **--test**, будут удалены логические тома. В этом случае можно заменить физический том и выполнить **vgcfgrestore** для восстановления группы.

6.7. НЕДОСТАТОЧНО СВОБОДНЫХ ЭКСТЕНТОВ ДЛЯ ЛОГИЧЕСКОГО ТОМА

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.

Вывод **vgdisplay** по умолчанию включает следующую строку, отображающую число свободных физических экстентов:

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

Или же можно передать опции **vg_free_count** и **vg_extent_count** команде **vgs** для отображения числа свободных экстентов и общего числа экстентов.

```
[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 34.30G 8780 8780
```

При наличии 8780 свободных физических экстентов можно выполнить следующую команду (укажите опцию "l" для использования в качестве единиц экстентов вместо байтов):

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

В этом случае будут заняты все свободные экстенты в группе томов.

```
# 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 Раздел 4.4.1.1, «Создание линейных томов».

ГЛАВА 7. АДМИНИСТРИРОВАНИЕ LVM ПРИ ПОМОЩИ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА

Помимо текстового интерфейса LVM предоставляет графический интерфейс (GUI, Graphical User Interface), с помощью которого можно выполнить настройку логических томов LVM. Для запуска графической утилиты исполните команду **system-config-lvm**. Глава LVM *Руководства по развертыванию Red Hat Enterprise Linux* содержит пошаговые инструкции по графической настройке логического тома.

LVM GUI также доступен в составе управляющего интерфейса Conga. Информация об использовании графического интерфейса LVM с Conga может быть найдена на странице помощи Conga в Интернете.

ПРИЛОЖЕНИЕ A. DEVICE MAPPER

Device Mapper представляет собой драйвер ядра, реализующий основную инфраструктуру для управления томами. Он позволяет создавать устройства с проецированием, которые могут использоваться в качестве логических томов. Device Mapper не обладает информацией о группах томов или форматах метаданных.

Device Mapper служит основой для некоторых технологий высокого уровня и используется командой **dmraid**, а также Device-Mapper Multipath.

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 supports a variety of mapping targets, including linear mapping, striped mapping, and error mapping. So, for example, two disks may be concatenated into one logical volume with a pair of linear mappings, one for each disk. When LVM2 creates a volume, it creates an underlying device-mapper device that can be queried with the **dmsetup** command. For information about the format of devices in a mapping table, see Раздел A.1, «Соответствия таблицы устройств». For information about using the **dmsetup** command to query a device, see Раздел A.2, «Команда dmsetup».

A.1. СООТВЕТСТВИЯ ТАБЛИЦЫ УСТРОЙСТВ

Проецируемое устройство определяется с помощью таблицы, которая содержит соответствия для диапазонов логических секторов. Строки таблицы следуют формату:

```
start length mapping [mapping_parameters...]
```

Параметр ***начало*** обычно равен 0. Сумма параметров ***начало*** и ***длина*** в одной строке должна быть равна величине ***начало*** на следующей строке. Параметры проецирования определяются типом ***соответствия***.

Размер для Device Mapper всегда указывается в секторах (размер сектора — 512 байт).

Если в Device Mapper устройство задано с помощью параметра соответствия, в файловой системе к нему все же можно обращаться по имени (например, **/dev/hda**) или по номеру в формате ***основной*:*вспомогательный***.

Пример таблицы соответствий, в которой определены 4 линейные составляющие:

```
0 35258368 linear 8:48 65920
35258368 35258368 linear 8:32 65920
70516736 17694720 linear 8:16 17694976
88211456 17694720 linear 8:16 256
```

Первые два параметра в строке обозначают начальный блок или длину сегмента. Следующий параметр определяет тип проецирования (**linear**).

Тип соответствий может принимать следующие значения:

- linear
- striped
- mirror

- snapshot и snapshot-origin
- error
- zero
- multipath
- crypt

A.1.1. Тип linear

Создает соответствие между непрерывным диапазоном блоков и другим блочным устройством.

```
start length linear device offset
```

start

начальный блок виртуального устройства

length

длина сегмента

device

блочное устройство, которое можно указать с помощью имени устройства или основного и вспомогательного номера в формате ***основной*:*вспомогательный***

offset

смещение соответствия

В следующем примере будет создано линейное соответствие начиная с нулевого блока, общей длиной 1638400. При этом основной и вспомогательный номера — 8:2, а смещение устройства — 41146992.

```
0 16384000 linear 8:2 41156992
```

А в приведенном далее примере будет создано линейное соответствие для устройства **/dev/hda**.

```
0 20971520 linear /dev/hda 384
```

A.1.2. Тип striped

Тип striped обеспечивает чередование между несколькими физическими устройствами и в качестве аргументов принимает число устройств и размер сегментов чередования с последующим списком пар имен устройств и секторов. Формат:

```
start length striped #stripes chunk_size device1 offset1 ... deviceN
offsetN
```

Каждому компоненту соответствует один набор параметров ***устройство*** и ***смещение***.

start

начальный блок виртуального устройства

length

длина сегмента

#stripes

число компонентов для организации чередования

chunk_size

число секторов, которое будет записано на одном устройстве, прежде чем будет выполнен переход к следующему. Должно быть кратно 2, а общий размер должен быть не меньше размера страницы ядра.

device

блочное устройство, заданное с помощью имени устройства или номеров в формате ***основной***:***вспомогательный***.

offset

смещение соответствия

Следующий пример демонстрирует чередование на трех устройствах, при этом размер сегмента равен 128 блокам:

```
0 73728 striped 3 128 8:9 384 8:8 384 8:7 9789824
```

0

начальный блок виртуального устройства

73728

длина сегмента

striped 3 128

чередование на трех устройствах с размером сегмента равным 128

8:9

основной и вспомогательный номера первого устройства

384

смещение на первом устройстве

8:8

основной и вспомогательный номера второго устройства

384

смещение на втором устройстве

8:7

основной и вспомогательный номера третьего устройства

9789824

смещение на третьем устройстве

В следующем примере будет создано чередование на двух устройствах; при этом размер сегмента будет равен 256 КиБ, а устройства будут заданы с помощью имен, а не номеров.

```
0 65536 striped 2 512 /dev/hda 0 /dev/hdb 0
```

A.1.3. Тип mirror

Тип mirror поддерживает проецирование зеркальных логических устройств. Формат:

```
start length mirror log_type #logargs logarg1 ... logargN #devs device1
offset1 ... deviceN offsetN
```

start

начальный блок виртуального устройства

length

длина сегмента

log_type

Допустимые значения типа журнала и аргументов:

core

Журнал локального зеркала хранится в основной памяти. Этот тип принимает от одного до трех аргументов:

размер_сектора [[**no**]**sync**] [**block_on_error**]

disk

Журнал локального зеркала хранится на диске. Этот тип принимает от двух до четырех аргументов:

устройство размер_сегмента [[**no**]**sync**] [**block_on_error**]

clustered_core

Журнал кластерного зеркала хранится в основной памяти. Этот тип принимает от двух до четырех аргументов:

размер_секции UUID [[**no**]**sync**] [**block_on_error**]

clustered_disk

Журнал кластерного зеркала хранится на диске. Этот тип принимает от трех до пяти аргументов:

устройство размер UUID [[**no**]**sync**] [**block_on_error**]

LVM поддерживает небольшой журнал, в котором регистрируется информация о регионах, которые синхронизированы с зеркалом. Параметр *размер* определяет размер регионов.

В кластерном окружении параметр *UUID* определяет уникальный идентификатор устройства журналирования. Это делается для того, чтобы поддерживать журнал в пределах кластера.

The optional **[no]sync** argument can be used to specify the mirror as "in-sync" or "out-of-sync". The **block_on_error** argument is used to tell the mirror to respond to errors rather than ignoring them.

#log_args

общее число аргументов журнала

logargs

аргументы для зеркала. Число аргументов определяется параметром ***#число_аргументов***, а допустимые аргументы задаются параметром ***тип_журнала***.

#devs

число компонентов зеркала. Устройство и смещение задаются для каждого компонента.

device

блочное устройство для каждого элемента зеркала, которое задается с помощью имени устройства или пары номеров в формате ***основной***:***вспомогательный***.

offset

исходное смещение. Блочное устройство и смещение задаются для каждого элемента зеркала.

Следующий пример демонстрирует проецирование для кластерного зеркала; при этом журнал зеркала хранится на диске.

```
0 52428800 mirror clustered_disk 4 253:2 1024 UUID block_on_error 3 253:3
0 253:4 0 253:5 0
```

0

начальный блок виртуального устройства

52428800

длина сегмента

mirror clustered_disk

зеркало, тип которого определяет, является ли оно кластерным или хранится на диске

4

далее следуют четыре аргумента

253:2

основной и вспомогательный номера устройства

1024

размер региона, в котором будет регистрироваться иноформация о синхронизации

UUID

UUID устройства журналирования, где будут храниться журналы кластера

block_on_error

зеркало должно реагировать на ошибки

3

число составляющих элементов зеркала

253:3 0 253:4 0 253:5 0

основной и вспомогательный номера и смещение для устройств

A.1.4. Тип snapshot и snapshot-origin

При создании первого LVM-снимка тома, используются четыре устройства Device Mapper:

1. Устройство с линейным проецированием (**linear**), содержащее изначальную таблицу соответствия для исходного тома.
2. Устройство с линейным проецированием, используемое в качестве COW-устройства (copy-on-write) для исходного тома. При каждой операции записи исходные данные сохраняются на COW-устройство каждого снимка с целью сохранения постоянства его видимого содержимого (до тех пор пока COW-устройство не заполнится).
3. Устройство с проецированием **snapshot**, совмещающим #1 и #2, которое представляет собой видимый снимок тома.
4. The "original" volume (which uses the device number used by the original source volume), whose table is replaced by a "snapshot-origin" mapping from device #1.

При создании этих устройств используется фиксированная схема присвоения имен. Например, команды создания LVM-тома с именем **base** и тома снимков **snap** на его основе будут выглядеть так:

```
# lvcreate -L 1G -n base volumeGroup
# lvcreate -L 100M --snapshot -n snap volumeGroup/base
```

В результате мы получим 4 устройства, которые можно просмотреть с помощью команды

```
# dmsetup table|grep volumeGroup
volumeGroup-base-real: 0 2097152 linear 8:19 384
volumeGroup-snap-cow: 0 204800 linear 8:19 2097536
volumeGroup-snap: 0 2097152 snapshot 254:11 254:12 P 16
volumeGroup-base: 0 2097152 snapshot-origin 254:11

# ls -lL /dev/mapper/volumeGroup-*
```

```
brw-------  1 root root 254, 11 29 ago 18:15 /dev/mapper/volumeGroup-base-
real
brw-------  1 root root 254, 12 29 ago 18:15 /dev/mapper/volumeGroup-snap-
cow
brw-------  1 root root 254, 13 29 ago 18:15 /dev/mapper/volumeGroup-snap
brw-------  1 root root 254, 10 29 ago 18:14 /dev/mapper/volumeGroup-base
```

Формат **snapshot-origin**:

```
start length snapshot-origin origin
```

start

начальный блок виртуального устройства

length

длина сегмента

origin

базовый том или снимок

snapshot-origin будет обычно включать один или несколько снимков. При записи исходные данные будут сохранены в COW-устройство каждого снимка. Это делается с целью сохранения постоянства содержимого, до тех пор пока COW-устройство не будет заполнено.

Формат **snapshot**:

```
start length snapshot origin COW-device P|N chunksize
```

start

начальный блок виртуального устройства

length

длина сегмента

origin

базовый том или снимок

COW-device

Устройство, где будут сохраняться измененные секции данных

P|N

Буквы P (Persistent) или N (Not persistent) определяют, будет ли сохраняться снимок после перезагрузки. Если указать P, то снимок будет сохраняться, и если указать N, снимок сохраняться не будет, метаданные не будут записываться на диск, вместо этого они будут храниться в памяти ядра.

chunksize

Размер изменяемых секций данных, которые будут сохраняться в COW-устройстве (в секторах).

Пример **snapshot-origin**:

```
0 2097152 snapshot-origin 254:11
```

В следующем примере исходное устройство — 254:11, COW-устройство — 254:12. Снимок будет сохраняться между перезагрузками, а размер секций данных, сохраняемых в COW-устройстве, равен 16 секторам.

```
0 2097152 snapshot 254:11 254:12 P 16
```

A.1.5. Тип error

Тип error заставит любые запросы ввода и вывода к заданному сектору завершиться неудачей.

«error» обычно используется для тестирования. Скажем, надо проверить поведение устройства в случае сбоя. Для этого просто создается соответствие с поврежденным сектором где-нибудь посередине или подменяется составляющий элемент зеркала.

Или же «error» используется для замены поврежденного устройства во избежание задержек и повтора неудачных попыток или при реорганизации метаданных LVM в случае сбоя.

error не требует указания параметров помимо *начало* и *конец*.

Пример:

```
0 65536 error
```

A.1.6. Тип zero

В качестве цели **zero** используется блочное устройство **/dev/zero**. Запросы чтения для такого соответствия будут возвращать блоки нулей, а запись хоть и будет осуществляться, но записанные данные будут просто удалены. **zero** принимает два параметра — *начало* и *length*.

Пример создания соответствия **zero** для устройства размером 16 Тбайт:

```
0 65536 zero
```

A.1.7. Тип multipath

Тип «multipath» обеспечивает сопоставление для Multipath-устройств. Формат:

```
start length  multipath  #features [feature1 ... featureN] #handlerargs
[handlerarg1 ... handlerargN] #pathgroups pathgroup pathgroupargs1 ...
pathgroupargsN
```

Каждой группе маршрутов соотвествует один набор параметров ***аргумент_группы***.

start

начальный блок виртуального устройства

length

длина сегмента

#features

Число Multipath-функций с их последующим списком. Если число равно нулю, то за этим параметром будет сразу следовать ***#число_аргументов***. В настоящее время поддерживается лишь одна возможность Multipath — **queue_if_no_path**, что обозначает, что если нет доступного маршрута, то операции ввода и вывода будут поставлены в очередь.

К примеру, если опция **no_path_retry** в файле **multipath.conf** ставит операции ввода и вывода в очередь, до тех пор пока все пути не будут отмечены как сбойные, после заданного числа попыток, соответствие будет выглядеть так:

```
0 71014400 multipath 1 queue_if_no_path 0 2 1 round-robin 0 2 1 66:128 \
1000 65:64 1000 round-robin 0 2 1 8:0 1000 67:192 1000
```

После того как все проверки маршрутов завершились неудачей, соответствие будет выглядеть так:

```
0 71014400 multipath 0 0 2 1 round-robin 0 2 1 66:128 1000 65:64 1000 \
round-robin 0 2 1 8:0 1000 67:192 1000
```

#handlerargs

Число аргументов аппаратного обработчика с последующим списком аргументов. Обработчик будет определять модуль для выполнения операций, связанных с оборудованием, при изменении групп путей или при обработке ошибок ввода и вывода. Если число аргументов — 0, то будет следовать параметр ***число_групп***.

#pathgroups

Число групп маршрутов. Группа маршрутов представляет собой набор маршрутов, используемых Multipath-устройством для распределения нагрузки. Каждой группе соответствует один набор параметров ***аргумент_группы***.

pathgroup

Следующая группа маршрутов.

pathgroupsargs

Каждой группе соответствуют следующие аргументы:

```
pathselector #selectorargs #paths #pathargs device1 ioreqs1 ... deviceN
ioreqsN
```

В каждой группе каждому маршруту соответствует один набор аргументов.

pathselector

Алгоритм для определения того, какой маршрут в этой группе будет использоваться для следующей операции ввода и вывода.

#selectorargs

Число аргументов. По умолчанию значение установлено в 0.

#paths

Число маршрутов для группы.

#pathargs

Число аргументов для каждого маршрута в группе. По умолчанию установлено в 1.

device

Номер блочного устройства маршрута в формате ***основной*:*вспомогательный*** номер.

ioreqs

Число запросов ввода и вывода, которые будут переданы этому маршруту, прежде чем будет выполнено переключение на другой маршрут в текущей группе.

Рисунок A.1, «Соответствие multipath» shows the format of a multipath target with two path groups.

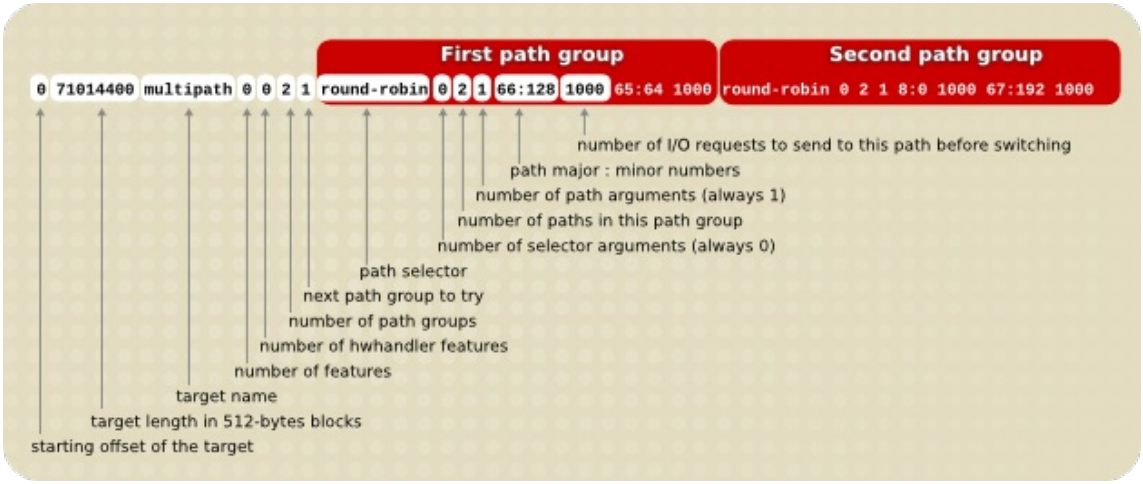


Рисунок A.1. Соответствие multipath

Следующий пример демонстрирует определение резервного устройства на случай сбоя Multipath-устройства. Цель включает четыре группы путей, при этом только один путь в группе может быть открыт. Таким образом Multipath-устройство будет использовать только один маршрут в определенный момент времени.

```
0 71014400 multipath 0 0 4 1 round-robin 0 1 1 66:112 1000 \
round-robin 0 1 1 67:176 1000 round-robin 0 1 1 68:240 1000 \
round-robin 0 1 1 65:48 1000
```

Следующий пример демонстрирует определение цели для того же Multipath-устройства, но в этом случае используется лишь одна группа маршрутов, между которыми загрузка будет распределена равномерно.

```
0 71014400 multipath 0 0 1 1 round-robin 0 4 1 66:112 1000 \
 67:176 1000 68:240 1000 65:48 1000
```

Документ *Использование Multipath Device Mapper* содержит дальнейшую информацию.

A.1.8. Тип crypt

crypt обеспечит кодирование передаваемых через заданное устройство данных. При этом используется API ядра Crypto.

Формат:

```
start length crypt cipher key IV-offset device offset
```

start

начальный блок виртуального устройства

length

длина сегмента

cipher

Код представляет собой ***cipher[-режим]-ivmode[:iv параметры]***.

cipher

Список кодов можно найти в файле **/proc/crypto** (например, **aes**).

chainmode

Всегда используйте **cbc**, а не **ebc**, так как **ebc** не использует исходный вектор (IV).

ivmode[:iv options]

IV — исходный вектор, используемый для изменения метода шифрования. Режим IV может принимать значения **plain** или **essiv:hash**. ***ivmode*** для **-plain** использует номер сектора (плюс смещение IV) в качестве IV. ***ivmode*** для **-essiv** представляет собой расширение, позволяющее обойти недостатки «водяных знаков».

ключ

Ключ шифрования в шестнадцатиричной форме

IV-offset

Смещение исходного вектора (IV, Initial Vector)

device

блочное устройство, которое можно указать с помощью имени устройства или основного и вспомогательного номера в формате ***основной*:*вспомогательный***

offset

смещение соответствия

Пример использования **crypt**:

-

```
0 2097152 crypt aes-plain 0123456789abcdef0123456789abcdef 0 /dev/hda 0
```

A.2. КОМАНДА DMSETUP

Для работы с Device Mapper используется команда **dmsetup**. Вспомогательные опции **info**, **ls**, **status**, **deps** команды **dmsetup** позволяют получить подробную информацию. Они будут рассмотрены ниже.

За информацией об опциях и возможностях **dmsetup** обратитесь к странице помощи **dmsetup**(8).

A.2.1. Команда dmsetup info

Команда **dmsetup info *устройство*** позволяет получить информацию об устройствах Device Mapper. Если устройство не указано, то будет отображена информация для всех настроенных устройств Device Mapper.

Формат вывода **dmsetup info**:

Name

Имя устройства, заданное как имя группы томов и имя логического тома, разделенные дефисом. Если исходное имя содержит дефис, то он будет указан дважды.

State

Статус устройства. Допустимые значения: **SUSPENDED**, **ACTIVE**, **READ-ONLY**. Например для перевода устройства в состояние **SUSPENDED** выполните команду **dmsetup suspend**, что приостановит все операции ввода и вывода. **dmsetup resume** снова активирует устройство, изменив его статус на **ACTIVE**.

Read Ahead

Число блоков данных, которые система будет считывать заранее для каждого открытого для чтения файла. По умолчанию значение выделяется автоматически; его можно изменить с помощью опции **--readahead** команды **dmsetup**.

Tables present

Possible states for this category are **LIVE** and **INACTIVE**. An **INACTIVE** state indicates that a table has been loaded which will be swapped in when a **dmsetup resume** command restores a device state to **ACTIVE**, at which point the table's state becomes **LIVE**. For information, see the **dmsetup** man page.

Open count

Счетчик попыток открыть устройство с помощью команды **mount**.

Event number

The current number of events received. Issuing a **dmsetup wait *n*** command allows the user to wait for the n'th event, blocking the call until it is received.

Major, minor

Основной и вспомогательный номера устройства.

Number of targets

Число фрагментов в составе устройства. Например, линейное устройство, в основу которого положено 3 диска, будет иметь 3 фрагмента, а линейное устройство, в состав которого входят начало и конец диска, будет иметь 2 фрагмента.

UUID

Идентификатор UUID.

Пример частичного вывода команды **dmsetup info**:

```
[root@ask-07 ~]# dmsetup info
Name:              testgfsvg-testgfslv1
State:             ACTIVE
Read Ahead:        256
Tables present:    LIVE
Open count:        0
Event number:      0
Major, minor:      253, 2
Number of targets: 2
UUID: LVM-K528WUGQgPadNXYcFrrf9LnPlUMswgkCkpgPIgYzSvigM7SfeWCypddNSWtNzc2N
...
Name:              VolGroup00-LogVol00
State:             ACTIVE
Read Ahead:        256
Tables present:    LIVE
Open count:        1
Event number:      0
Major, minor:      253, 0
Number of targets: 1
UUID: LVM-tOcS1kqFV9drb0X1Vr8sxeYP0tqcrpdegyqj5lZxe45JMGlmvtqLmbLpBcenh2L3
```

A.2.2. Команда dmsetup ls

С помощью команды **dmsetup ls** можно отобразить список имен устройств. **dmsetup ls --target *тип*** позволит отобразить устройства, обладающие по крайней мере одним целевым фрагментом заданного типа. Другие опции команды **dmsetup ls** можно просмотреть на странице помощи **dmsetup**.

Пример отображения списка настроенных устройств:

```
[root@ask-07 ~]# dmsetup ls
testgfsvg-testgfslv3    (253, 4)
testgfsvg-testgfslv2    (253, 3)
testgfsvg-testgfslv1    (253, 2)
VolGroup00-LogVol01     (253, 1)
VolGroup00-LogVol00     (253, 0)
```

Пример отображения списка устройств для настроенных зеркальных соответствий:

```
[root@grant-01 ~]# dmsetup ls --target mirror
lock_stress-grant--02.1722      (253, 34)
lock_stress-grant--01.1720      (253, 18)
```

```
lock_stress-grant--03.1718      (253, 52)
lock_stress-grant--02.1716      (253, 40)
lock_stress-grant--03.1713      (253, 47)
lock_stress-grant--02.1709      (253, 23)
lock_stress-grant--01.1707      (253, 8)
lock_stress-grant--01.1724      (253, 14)
lock_stress-grant--03.1711      (253, 27)
```

A.2.3. Команда dmsetup status

Команда **dmsetup status *устройство*** позволяет просмотреть статус каждого целевого фрагмента для заданного устройства. Если устройство не указано, будет отображена информация для всех настроенных устройств. **dmsetup status --target *тип*** позволит отобразить статус устройств, обладающих по крайней мере одним целевым фрагментом заданного типа.

Пример отображения статуса для всех настроенных устройств:

```
[root@ask-07 ~]# dmsetup status
testgfsvg-testgfslv3: 0 312352768 linear
testgfsvg-testgfslv2: 0 312352768 linear
testgfsvg-testgfslv1: 0 312352768 linear
testgfsvg-testgfslv1: 312352768 50331648 linear
VolGroup00-LogVol01: 0 4063232 linear
VolGroup00-LogVol00: 0 151912448 linear
```

A.2.4. Команда dmsetup deps

Команда **dmsetup deps *устройство*** позволяет отобразить список устройств (в виде пары основного и вспомогательного номеров), входящих в таблицу соответствий для заданного устройства. Если устройство не указано, будет выведена информация обо всех устройствах.

Пример команды отображения зависимостей для всех настроенных устройств:

```
[root@ask-07 ~]# dmsetup deps
testgfsvg-testgfslv3: 1 dependencies    : (8, 16)
testgfsvg-testgfslv2: 1 dependencies    : (8, 16)
testgfsvg-testgfslv1: 1 dependencies    : (8, 16)
VolGroup00-LogVol01: 1 dependencies     : (8, 2)
VolGroup00-LogVol00: 1 dependencies     : (8, 2)
```

Пример команды отображения зависимостей для устройства **lock_stress-grant--02.1722**:

```
[root@grant-01 ~]# dmsetup deps lock_stress-grant--02.1722
3 dependencies  : (253, 33) (253, 32) (253, 31)
```

ПРИЛОЖЕНИЕ B. ФАЙЛЫ КОНФИГУРАЦИИ LVM

LVM поддерживает разные файлы конфигурации. При запуске системы из каталога, заданного переменной окружения **LVM_SYSTEM_DIR** (по умолчанию установлена в **/etc/lvm**) будет загружен файл **lvm.conf**.

В **lvm.conf** могут быть указаны дополнительные файлы конфигурации для загрузки. Последующие настройки будут переопределять предыдущие. Чтобы отобразить текущие настройки после загрузки всех файлов конфигурации, выполните команду **lvm dumpconfig**.

For information on loading additional configuration files, see Раздел C.2, «Теги узлов».

B.1. ФАЙЛЫ КОНФИГУРАЦИИ LVM

Для конфигурации LVM используются следующие файлы:

/etc/lvm/lvm.conf

Основной файл конфигурации, к которому обращаются утилиты

etc/lvm/lvm_*тег_узла*.conf

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 tiles to read in. For information on host tags, see Раздел C.2, «Теги узлов».

Также при системной настройке LVM используются следующие файлы:

/etc/lvm/.cache

кэш-файл фильтров имен устройств (настраиваемый)

/etc/lvm/backup/

каталог, куда сохраняются автоматически созданные резервные копии метаданных групп томов (настраиваемый)

/etc/lvm/archive/

каталог, куда сохраняются архивы автоматически созданных резервных копий метаданных групп томов (настраивается в зависимости от глубины истории архивации и пути)

/var/lock/lvm/

Для одного узла используются файлы блокировки, предотвращающие разрушение данных при параллельной работе утилит. В окружении кластера будет применяться DLM.

B.2. ПРИМЕР ФАЙЛА LVM.CONF

Далее приведен пример файла конфигурации **lvm.conf**, который используется по умолчанию в RHEL 5.3. Если версия вашей системы отличается, то и настройки могут быть другими.

```
[root@tng3-1 lvm]# cat /etc/lvm/lvm.conf
# 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" ]

    # If several entries in the scanned directories correspond to the
    # same block device and the tools need to display a name for device,
    # all the pathnames are matched against each item in the following
    # list of regular expressions in turn and the first match is used.
    preferred_names = [ ]

    # Try to avoid using undescriptive /dev/dm-N names, if present.
    # preferred_names = [ "^/dev/mpath/", "^/dev/mapper/mpath",
"^/dev/[hs]d" ]

    # 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 -
vvvv'.
```

```
    # By default we accept every block device:
    filter = [ "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 is stored in the /etc/lvm/cache directory
    # in a file called '.cache'.
    # It is safe to delete the contents: the tools regenerate it.
    # (The old setting 'cache' is still respected if neither of
    # these new ones is present.)
    cache_dir = "/etc/lvm/cache"
    cache_file_prefix = ""

    # 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

    # By default, if a PV is placed directly upon an md device, LVM2
    # will align its data blocks with the the chunk_size exposed in sysfs.
    # 1 enables; 0 disables.
    md_chunk_alignment = 1

    # If, while scanning the system for PVs, LVM2 encounters a device-
mapper
    # device that has its I/O suspended, it waits for it to become
accessible.
    # Set this to 1 to skip such devices.  This should only be needed
    # in recovery situations.
    ignore_suspended_devices = 0
```

```
}

# 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.
    verbose = 0

    # Should we send log messages through syslog?
    # 1 is yes; 0 is no.
    syslog = 1

    # Should we log error and debug messages to a file?
    # By default there is no log file.
    #file = "/var/log/lvm2.log"

    # Should we 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 readeable 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/lib64"

    # 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

    # Default value for --units argument
    units = "h"
```

```
    # 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 suffices
    # 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 = 3

    # 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 {
    # How to fill in missing stripes if activating an incomplete volume.
    # Using "error" will make inaccessible parts of the device return
    # I/O errors on access.  You can instead use a device path, in which
    # case, that device will be used to in place of missing stripes.
    # But note that using anything other than "error" with mirrored
    # or snapshotted volumes is likely to result in data corruption.
    missing_stripe_filler = "error"

    # 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/lvol1", "@tag1", "@*" ]

    # Size (in KB) of each copy operation when mirroring
    mirror_region_size = 512

    # Setting to use when there is no readahead value stored in the
metadata.
    #
    # "none" - Disable readahead.
    # "auto" - Use default value chosen by kernel.
    readahead = "auto"

    # '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 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 would not be wise for mirror devices
    #            because it would break the redundant nature of the
    #            mirror.  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"
}


####################
# Advanced section #
####################

# 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.

    # pvmetadatacopies = 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 areas.
    # 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
    # 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. If no mirror library is
    # provided, mirrors are not monitored through dmeventd.

    mirror_library = "libdevmapper-event-lvm2mirror.so"

    # snapshot_library is the library used when monitoring a snapshot
device.
    #
    # "libdevmapper-event-lvm2snapshot.so" monitors the filling of
    # snapshots and emits a warning through syslog, when the use of
    # snapshot exceedes 80%. The warning is repeated when 85%, 90% and
    # 95% of the snapshot are filled.

    snapshot_library = "libdevmapper-event-lvm2snapshot.so"
}
```

ПРИЛОЖЕНИЕ C. ТЕГИ ОБЪЕКТОВ LVM

Теги представляют собой ключевые слова, используемые для группировки объектов LVM2 одного типа. Теги могут быть присвоены физическим томам, группам томов, логическим томам, сегментам, а также узлам при конфигурации кластера. Снимкам теги не присваиваются.

Теги также можно передать команде вместо аргументов PV, VG, LV. Перед тегами необходимо указать символ "@" во избежание неоднозначности. При обработке тега он будет замещен объектами с этим тегом.

Теги LVM могут включать до 128 символов, должны содержать только символы [A-Za-z0-9_+.-] и не могут начинаться с дефиса.

Теги могут быть применены только к объектам в составе группы томов. Физический том лишится тега при его исключении из группы, так как теги сохраняются только в составе метаданных группы и при удалении физического тома они также будут удалены. Теги нельзя присвоить снимкам.

Следующая команда отобразит список всех логических томов с тегом **database**.

```
lvs @database
```

C.1. ДОБАВЛЕНИЕ И УДАЛЕНИЕ ТЕГОВ

Для добавления или удаления тегов физических томов можно использовать соответствующие опции **--addtag** и **--deltag** команды **pvchange**.

Для добавления или удаления тегов групп томов можно использовать соответствующие опции **--addtag** и **--deltag** команд **vgchange** и **vgcreate**.

Для добавления или удаления тегов логических томов можно использовать соответствующие опции **--addtag** и **--deltag** команд **lvchange** и **lvcreate**.

C.2. ТЕГИ УЗЛОВ

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 Приложение B, *Файлы конфигурации LVM*.

Для каждого тега узла будет выполнено обращение к файлу конфигурации lvm_*тег_узла*.conf. Если он содержит определения новых тегов, тогда имена соответствующих файлов конфигурации будут добавлены к списку имен для последующего чтения.

Например, приведенная далее запись в файле конфигурации будет всегда определять **tag1**, а также **tag2**, если имя узла — **host1**.

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

C.3. УПРАВЛЕНИЕ АКТИВАЦИЕЙ С ПОМОЩЬЮ ТЕГОВ

В файле конфигурации можно указать, какие логические тома должны быть активированы на заданном узле. Например, приведенная далее запись используется в качестве фильтра запросов активации (таких как **vgchange -ay**). В данном случае будет активирован том **vg1/lvol0**, а также логические тома и группы с тегом**database** на текущем узле.

```
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.

Представьте ситуацию, когда для каждой машины в пределах кластера существует запись в файле конфигурации:

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

Для активации **vg1/lvol2** только на узле **db2** выполните следующую последовательность шагов:

1. На любом узле кластера выполните команду **lvchange --addtag @db2 vg1/lvol2**.
2. Выполните **lvchange -ay vg1/lvol2**.

При этом имена узлов будут сохранены в метаданных группы томов.

ПРИЛОЖЕНИЕ D. МЕТАДАННЫЕ ГРУППЫ ТОМОВ LVM

Метаданные хранят информацию о настройках группы томов. Их копия по умолчанию хранится в области метаданных каждого физического тома в составе группы томов. Метаданные LVM не занимают много места и хранятся в формате ASCII.

Если группа включает достаточно много физических томов, то хранение многочисленных избыточных копий довольно неэффективно. Тогда с помощью опции **--metadatacopies 0** команды **pvcreate** можно создать физический том без копий метаданных. Стоит помнить, что выбрав число копий метаданных для физического тома один раз, вы уже не сможете его изменить. Нулевое значение позволит ускорить процесс получения обновлений конфигурации. Однако каждая группа томов должна содержать как минимум один физический том с областью метаданных (за исключением случаев использования пользовательских настроек, позволяющих хранить метаданные группы томов в файловой системе). Если вы планируете разделить группу томов, необходимо, чтобы каждой группе соответствовала хотя бы одна копия метаданных.

Основные метаданные хранятся в формате ASCII. Область метаданных представляет собой циклический буфер. Таким образом, новые метаданные добавляются в конец уже существующих, соответственно перемещается указатель начала данных.

Размер области метаданных можно определить с помощью аргумента **--metadatasize** команды **pvcreate**. Размер, используемый по умолчанию, слишком мал для групп с большим числом логических или физических томов.

D.1. МЕТКА ФИЗИЧЕСКОГО ТОМА

По умолчанию команда **pvcreate** размещает метку во втором секторе (размер сектора — 512 Мб). Метку также можно поместить в любой из первых четырех секторов, так как утилиты LVM сканируют первые четыре сектора в поисках метки. Сама метка физического тома начинается со строки **LABELONE**.

Метка PV содержит:

- UUID физического тома.
- Размер блочного устройства в байтах.
- Список, содержащий информацию о расположении областей данных. Список завершается NULL.
- Списки, содержащие информацию о расположении областей метаданных. Списки завершаются NULL.

Информация о расположении метаданных включает смещение и размер (в байтах). Метка может вмещать сведения о 15 местах расположения, в то время как утилиты LVM используют лишь 3 — одна область данных и максимум две области метаданных.

D.2. СОДЕРЖИМОЕ МЕТАДАННЫХ

Метаданные группы томов содержат следующую информацию:

- Информацию о том, как и когда были созданы метаданные
- Информация о группе томов

Информация о группе томов включает:

- Имя и уникальный идентификатор
- Номер версии, который увеличивается на единицу при каждом обновлении метаданных
- Параметры: Чтение/ запись? Возможно ли изменять размер?
- Любое административное ограничение на число физических и логических томов
- Размер экстента, определяемый числом секторов, размер которых составляет 512 Байт
- Неупорядоченный список физических томов в составе группы. Определение каждого тома включает:
  - Его UUID, используемый для определения блочного устройства, содержащего этот физический том
  - Параметры, например, возможность выделения пространства данного тома
  - Смещение начала первого экстента в пределах физического тома (в секторах)
  - Число экстентов
- Неупорядоченный список логических томов. Информация включает:
  - Упорядоченный список сегментов логического тома. Метаданные каждого сегмента включают соответствия упорядоченного списка сегментов логических или физических томов

D.3. ПРИМЕР МЕТАДАННЫХ

Пример метаданных для группы томов с именем **myvg** будет выглядеть так:

```
# 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"]
        extent_size = 8192              # 4 Megabytes
        max_lv = 0
        max_pv = 0

        physical_volumes {
```

```
                pv0 {
                        id = "ZBW5qW-dXF2-0bGw-ZCad-2RlV-phwu-1c1RFt"
                        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-Ur9OF9"
                        status = ["READ", "WRITE", "VISIBLE"]
                        segment_count = 2

                        segment1 {
                                start_extent = 0
                                extent_count = 1280     # 5 Gigabytes

                                type = "striped"
                                stripe_count = 1        # linear

                                stripes = [
                                        "pv0", 0
```

```
                                ]
                        }
                        segment2 {
                                start_extent = 1280
                                extent_count = 1280     # 5 Gigabytes

                                type = "striped"
                                stripe_count = 1        # linear

                                stripes = [
                                        "pv1", 0
                                ]
                        }
                }
        }
}
```

ПРИЛОЖЕНИЕ E. ИСТОРИЯ ИЗМЕНЕНИЙ

Издание 3-6.400 **2013-10-31** **Rüdiger Landmann**
Rebuild with publican 4.0.0

Издание 3-6 **2012-07-18** **Anthony Towns**
Rebuild for Publican 3.0

Издание 1.0-0 **Thu Jan 29 2009**

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

A

B

C

M

O

P