Red Hat Enterprise Linux 6

LVM

Руководство администратора LVM

Редакция 1

Last Updated: 2017-10-13

Red Hat Enterprise Linux 6 LVM

Руководство администратора LVM
Редакция 1

Landmann
rlandmann@redhat.com

Юридическое уведомление

Аннотация

В данном документе приведена информация об управлении логическими томами, в том числе и в кластерном окружении.

Содержание

ВВЕДЕНИЕ

1. ОБ ЭТОМ РУКОВОДСТВЕ

В данном документе рассматривается управление логическими томами (LVM, Logical Volume Manager) в кластерном окружении.

2. ЦЕЛЕВАЯ АУДИТОРИЯ

Материал ориентирован на опытных администраторов Red Hat Enterprise Linux 6 со знаниями GFS2.

3. ВЕРСИИ

Таблица 1. Версии

Название	Описание
RHEL 6	RHEL 6 или более поздние версии
GFS2	GFS2 для RHEL6 и последующих версий

4. ДОПОЛНИТЕЛЬНАЯ ДОКУМЕНТАЦИЯ

Другие документы:

- *Руководство по установке Red Hat Enterprise Linux 6*
- *Руководство по развертыванию* предоставляет информацию по установке, настройке и администрированию Red Hat Enterprise Linux 6.
- *Руководство по управлению накопителями* содержит инструкции по эффективному управлению устройствами хранения данных и файловыми системами Red Hat Enterprise Linux 6.

Подробную информацию о комплектах распределения нагрузки и отказоустойчивого хранения данных можно найти в следующих документах:

- *Обзор комплекта высокой готовности*.
- *Управление кластером* содержит информацию об установке, настройке и управлении кластерными компонентами Red Hat.
- *Администрирование GFS2* предоставляет информацию об установке, настройке и поддержке Red Hat GFS2 (Global File System 2).
- *DM Multipath* предоставляет информацию о многопутевых возможностях Red Hat Enterprise Linux 6.

- *Распределение нагрузки* предоставляет информацию о настройке высокопроизводительных систем и служб с помощью комплекта распределения нагрузки в группе физических серверов.

- *Примечания к выпуску* содержат краткий обзор последнего выпуска Red Hat.

Полный диапазон документов Red Hat доступен в виде HTML, PDF и RPM на диске документации Red Hat Enterprise Linux и на http://www.redhat.com/docs/.

5. ОТЗЫВЫ И ПРЕДЛОЖЕНИЯ

Если вы обнаружили опечатку, или у вас есть предложения по усовершенствованию руководства, создайте запрос в Bugzilla (http://bugzilla.redhat.com/) для выпуска **Red Hat Enterprise Linux 6** и компонента **doc-Logical_Volume_Manager**. Укажите идентификатор: **`Logical_Volume_Manager_Administration(EN)-6 (2011-05-19-15:20)`**.

Если у вас есть предложения по улучшению руководства, постарайтесь подробно их описать. Для облегчения идентификации ошибок и опечаток укажите номер раздела и окружающий текст.

ГЛАВА 1. ОБЗОР

В этой главе приведена общая информация об LVM и его основных характеристиках в Red Hat Enterprise Linux 6.

1.1. НОВЫЕ И ОБНОВЛЕННЫЕ ВОЗМОЖНОСТИ

Эта секция содержит перечень новых и измененных функций LVM в Red Hat Enterprise Linux 6.

1.1.1. Изменения в Red Hat Enterprise Linux 6.0

Ниже перечислены основные изменения LVM в Red Hat Enterprise Linux 6.0.

- Параметры **mirror_image_fault_policy** и **mirror_log_fault_policy** в секции **activation** файла **lvm.conf** позволяют настроить поведение зеркального тома в случае сбоя устройства. Если **mirror_image_fault_policy** имеет значение **remove**, система попытается исключить проблемное устройство и продолжить работу. Значение **allocate** определяет, что после удаления устройства необходимо выделить пространство для нового устройства. При отсутствии подходящей замены **allocate** работает аналогично **remove** (см. Раздел 4.4.3.1, «Правила работы при сбое зеркального тома»).

- Стек ввода-вывода Linux стал распознавать ограничения ввода-вывода, заданные производителем, что позволяет оптимизировать размещение и доступ к данным. Отключать эту функциональность не рекомендуется, но при необходимости это можно сделать с помощью параметров **data_alignment_detection** и **data_alignment_offset_detection** в **lvm.conf**.

  Информацию о выравнивании данных в LVM и изменении **data_alignment_detection** и **data_alignment_offset_detection** можно можно найти в документации к **/etc/lvm/lvm.conf** (см. Приложение B, *Файлы конфигурации LVM*). Общие сведения о стеке ввода-вывода и его ограничениях в Red Hat Enterprise Linux 6 можно найти в *руководстве по управлению накопителями*.

- Device-mapper напрямую поддерживает интеграцию **udev**, что позволяет синхронизировать проекции устройств, в том числе и для устройств LVM (см. Раздел A.3, «Поддержка udev»).

- Для восстановления зеркала после сбоя диска можно использовать **lvconvert --repair** (см. Раздел 4.4.3.3, «Восстановление зеркального устройства»).

- **lvconvert --merge** позволяет объединить снимок с исходным томом (см. Раздел 4.4.5, «Объединение снимка с оригиналом»).

- **lvconvert --splitmirrors** позволяет разделить образ зеркального тома на части, тем самым состав новый том (см. Раздел 4.4.3.2, «Разбиение образа зеркального тома»).

- Теперь можно создать журнал для зеркального логического устройства, для которого также создано зеркало. Для этого используется параметр **--mirrorlog mirrored** команды **lvcreate** (см. Раздел 4.4.3, «Создание зеркальных томов»).

1.1.2. Изменения в Red Hat Enterprise Linux 6.1

Ниже перечислены основные изменения LVM в Red Hat Enterprise Linux 6.1.

- Допускается создание снимков зеркальных логических томов аналогично тому, как создаются снимки обычных томов (см. Раздел 4.4.4, «Создание снимков»).

- При увеличении размера тома LVM можно использовать параметр **`--alloc cling`** команды **`lvextend`**. При этом место будет выбираться в пределах тех же физических томов, где расположен последний сегмент увеличиваемого тома. Если места не хватает, будет проверен файл **`lvm.conf`** и выбраны диски с тем же тегом.

  Раздел 4.4.12.2, «Увеличение логического тома в режиме **`cling`**» содержит описание наращивания зеркальных томов с помощью **`--alloc cling`**.

- Допускается многократное указание аргументов **`--addtag`** и **`--deltag`** в командах **`pvchange`**, **`vgchange`** и **`lvchange`** (см. Раздел C.1, «Добавление и удаление тегов»).

- Теперь теги объектов LVM могут содержать "/", "=", "!", ":", "#", "&" (см. Приложение C, *Теги объектов LVM*).

- Логический том теперь может состоять из комбинаций RAID0 и RAID1. Если при создании тома было указано число зеркал (**`--mirrors X`**) и звеньев (**`--stripes Y`**), то будет создано зеркальное устройство с чередующимися составляющими (см. Раздел 4.4.3, «Создание зеркальных томов»).

- Для создания резервной копии кластерного логического тома теперь можно специально активировать том и создать его снимок (см. Раздел 4.7, «Активация логических томов на отдельных узлах кластера»).

1.2. ЛОГИЧЕСКИЙ ТОМ

LVM добавляет дополнительный уровень абстракции над физическими дисками, позволяя создавать на их основе логические тома, что обеспечивает бóльшую гибкость по сравнению с управлением физических дисков напрямую. Аппаратная конфигурация накопителей спрятана от программ, поэтому их можно переносить или изменять размер без необходимости остановки приложений и отключения файловых систем.

Преимущества логических томов:

- Возможность изменения размера.

  Размер файловых систем логических томов не ограничивается одним диском, так как том может располагаться на разных дисках и разделах.

- Изменение размера.

  С помощью простых команд можно изменить размер логических томов без необходимости переформатирования или переразбиения физических дисков в их основе.

- Перемещение данных в активной системе.

  С целью создания новых, более быстрых и устойчивых подсистем хранения допускается перемещать данные в активной системе. Данные можно переносить, даже если к дискам выполняется обращение. Например, можно на ходу освободить заменяемый диск перед его удалением из набора.

- Простота присвоения имен устройствам.

Логические тома могут объединяться в группы для облегчения управления. Группам могут присваиваться произвольные имена.

- Чередование дисков.

  Для повышения пропускной способности можно создать логический том с чередованием данных на нескольких дисках.

- Зеркалирование томов.

  Использование логических томов позволяет создать зеркальные копии данных.

- Снимки томов.

  Снимки томов представляют собой их точные копии и могут использоваться для тестирования, не подвергая риску исходный том.

Перечисленные возможности будут рассмотрены в этом документе.

1.3. ОБЗОР АРХИТЕКТУРЫ LVM

На смену исходному механизму LVM1, впервые появившемуся в RHEL4, пришел LVM2, архитектура которого более универсальна. LVM2 обладает следующими преимуществами:

- гибкая емкость;
- эффективное хранение метаданных;
- улучшенный формат восстановления;
- новый формат метаданных ASCII;
- выборочное изменение метаданных;
- поддержка избыточных копий метаданных.

LVM2 обратно совместим с LVM1 (за исключением снимков и кластерной поддержки). Группу томов LVM1 можно преобразовать в LVM2 с помощью команды **vgconvert**. Информацию о преобразовании формата метаданных можно найти на справочной странице **vgconvert**(8).

В основе логического тома лежит блочное устройство — раздел или целый диск. Это устройство инициализируется как *физический том*.

Физические тома объединяются в *группы томов*, тем самым создавая единое пространство для организации логических томов. Этот процесс аналогичен разбиению дисков на разделы. Логический том будет доступен файловым системам и приложениям.

Рисунок 1.1, «Компоненты логического тома» демонстрирует примеры логических томов.

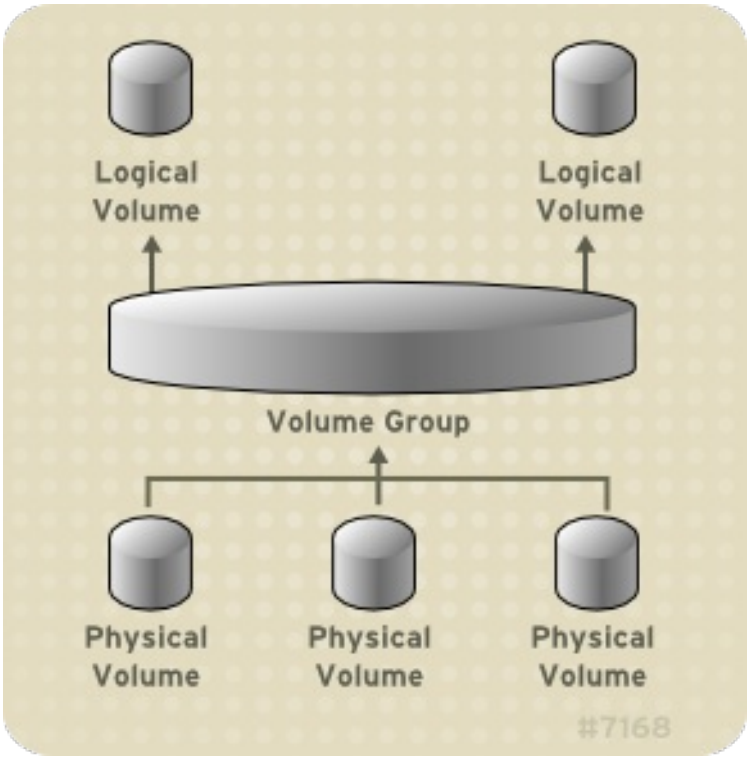


Рисунок 1.1. Компоненты логического тома

Глава 2, *Компоненты LVM* содержит подробную информацию о составляющих логического тома.

1.4. CLVM

Кластерное управление логическими томами (CLVM, Clustered Logical Volume Manager) представляет собой набор кластерных расширений для LVM, позволяющих кластеру управлять общим пространством хранения, например в сети SAN. CLVM входит в состав комплекта отказоустойчивого хранилища RHEL 6.

Выбор CLVM также определяется требованиями, перечисленными ниже.

- Если доступ к хранилищу необходим лишь одному узлу, можно ограничиться возможностями LVM. Создаваемые логические тома будут локальными.
- Если для восстановления после отказа используется кластерная структура, где только один узел может обращаться к хранилищу в заданный момент времени, потребуются агенты HA-LVM (High Availability Logical Volume Management). Подробную информацию можно найти в документе под названием *Конфигурация и управление кластером Red Hat*.
- Если пространство хранения должно быть доступно нескольким активным компьютерам, потребуется прибегнуть к помощи CLVM. CLVM позволит пользователю настроить логические тома в общем хранилище, блокируя доступ к физическому хранилищу во время настройки логического тома и используя кластерные службы блокирования для управления доступом.

Для работы CLVM необходимо, чтобы в системе работали комплекты высокой готовности и отказоустойчивого хранилища. На всех компьютерах в кластере должен выполняться процесс **`clvmd`**, который является главным кластерным расширением LVM. **`clvmd`** передает компьютерам обновления метаданных LVM, тем самым поддерживая постоянную структуру логических томов. Подробную информацию об комплекте высокой готовности можно найти в руководстве *Конфигурация и управление кластером Red Hat*.

Чтобы проверить, был ли запущен процесс **clmvd** во время загрузки компьютера, выполните:

```
# chkconfig clvmd on
```

Если **clvmd** не выполняется, для запуска выполните:

```
# service clvmd start
```

Создание логических томов в кластере аналогично созданию томов на одном узле (см. Глава 4, *Администрирование LVM в текстовом режиме* и Глава 7, *Администрирование LVM в графическом режиме*). Чтобы активировать созданные в кластере тома, необходимо, чтобы функционировала инфраструктура кластера.

По умолчанию логические тома, созданные в общем хранилище с помощью CLVM, будут видны всем компьютерам, у которых есть доступ к этому хранилищу. Можно создать целые группы томов, устройства хранения которых будут видны только одному компьютеру в кластере. Дополнительно можно изменить статус группы с «локальной» на «кластерную» (см. Раздел 4.3.2, «Создание групп томов в кластере» и Раздел 4.3.7, «Изменение параметров группы томов»).

ПРЕДУПРЕЖДЕНИЕ

При создании групп томов в общем хранилище с помощью CLVM необходимо убедиться, что у всех компьютеров в кластере есть доступ к физическим томам в составе группы. Асимметричные схемы кластеров с неравномерными правами доступа к хранилищу не поддерживаются.

Рисунок 1.2, «Структура CLVM» демонстрирует CLVM в кластере Red Hat.

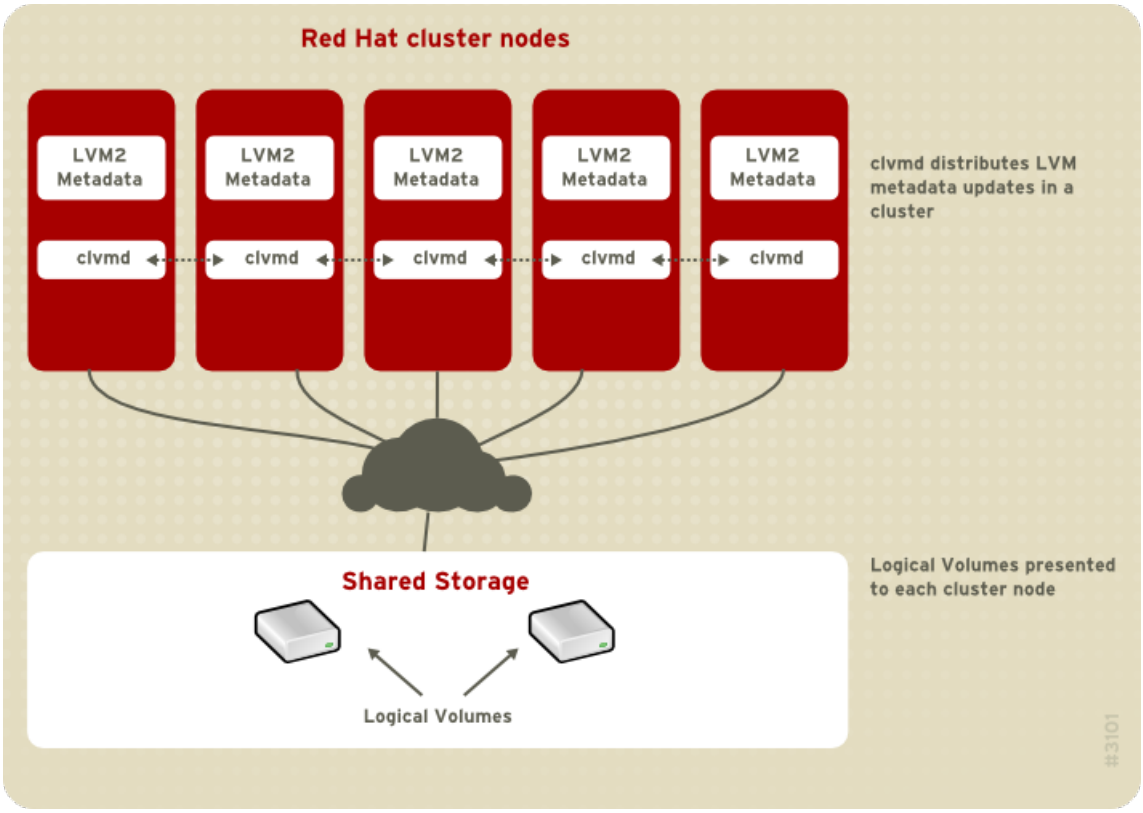


Рисунок 1.2. Структура CLVM

ПРИМЕЧАНИЕ

Для блокирования в пределах кластера потребуется внести изменения в **`lvm.conf`**. Информацию о настройке кластерного блокирования можно найти в файле **`lvm.conf`** (см. Приложение B, *Файлы конфигурации LVM*).

1.5. СОДЕРЖАНИЕ ДОКУМЕНТА

Содержание:

- Глава 2, *Компоненты LVM* описывает составляющие компоненты логического тома.
- Глава 3, *Обзор администрирования LVM* содержит обзор основных шагов по настройке логических томов с помощью команд или графического интерфейса LVM.
- Глава 4, *Администрирование LVM в текстовом режиме* содержит обзор некоторых административных задач по созданию и управлению логическими томами.
- Глава 5, *Примеры конфигурации LVM* приводит примеры конфигурации LVM.
- Глава 6, *Диагностика проблем* содержит описание методов диагностики и решения типичных проблем LVM.
- Глава 7, *Администрирование LVM в графическом режиме* содержит обзор графического интерфейса LVM.
- Приложение A, *Device Mapper* содержит описание модуля Device-mapper, который LVM использует для создания соответствий между логическими и физическими томами.

- Приложение B, *Файлы конфигурации LVM* содержит описание файлов конфигурации LVM.
- Приложение C, *Теги объектов LVM* рассказывает о тегах объектов и узлов LVM.
- Приложение D, *Метаданные группы томов* рассматривает структуру метаданных групп томов и приводит их образец.

ГЛАВА 2. КОМПОНЕНТЫ LVM

В этой главе рассматриваются компоненты LVM.

2.1. ФИЗИЧЕСКИЙ ТОМ

В основе логического тома лежит блочное устройство — раздел или целый диск. Устройство инициализируется как *физический том*. В начале тома размещается специальная метка.

Метка LVM по умолчанию размещается во втором 512-байтном секторе. По желанию ее можно разместить в любом из первых четырех секторов, что позволяет томам использовать эти сектора параллельно с другими пользователями.

Метка определяет порядок устройств, так как очередность их обнаружения в процессе загрузки может меняться. Она не изменяется между перезагрузками в пределах кластера.

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

Метаданные LVM содержат настройки групп томов в системе. По умолчанию в секции метаданных каждого физического тома в группе хранится копия метаданных. Метаданные не занимают много места и хранятся в формате ASCII.

В настоящее время LVM позволяет сохранить 1-2 копии метаданных в каждом физическом томе. По умолчанию сохраняется одна копия. Задав число копий один раз, его нельзя будет изменить. Первая копия хранится в начале устройства — вскоре после метки. Вторая копия (если она существует) располагается в конце устройства. Если область в начале диска была случайно перезаписана, вторая копия поможет восстановить метаданные.

Приложение D, *Метаданные группы томов* содержит подробную информацию.

2.1.1. Структура физического тома

Рисунок 2.1, «Структура физического тома» демонстрирует организацию физического тома. Метка LVM расположена во втором секторе, за ней следует область метаданных. Остальное пространство доступно для использования.

ПРИМЕЧАНИЕ

В этом документе предполагается, что размер секторов составляет 512 байт (как и в ядре Linux).

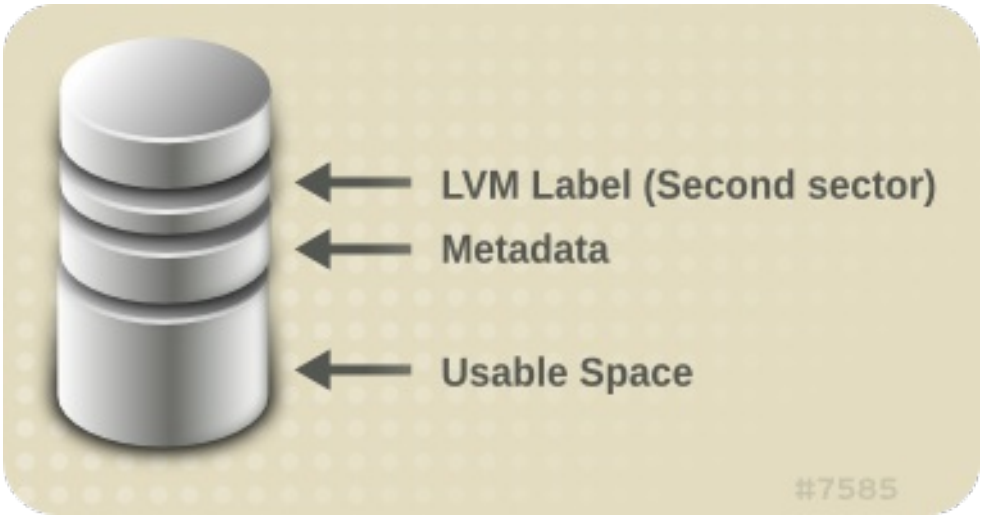


Рисунок 2.1. Структура физического тома

2.1.2. Диски с несколькими разделами

LVM позволяет создавать физические тома на основе дисковых разделов. В этом случае рекомендуется создать единственный раздел, охватывающий весь диск, и присвоить ему метку физического тома. Достоинства такого подхода:

- Облегчение администрирования.

  Так намного легче следить за отдельными дисками, особенно в случае сбоя. Кроме того, несколько физических томов на одном диске при загрузке приведут к появлению предупреждения во время загрузки о неизвестных типах разделов.

- Производительность при чередовании.

  LVM не различает, расположены ли два физических тома на одном и том же диске. Если логический том с чередованием был создан на основе двух физических томов, расположенных на одном диске, может оказаться так, что области чередования расположены в различных разделах одного диска. Это приведет к снижению производительности вместо ожидаемой оптимизации.

Хоть это и не рекомендуется, но может случиться так, что диск все же необходимо разделить на несколько физических томов. Например, в системе с несколькими дисками может понадобиться перенести данные между разделами при миграции существующей системы в формат LVM. При наличии диска большого размера потребуется разбить его на разделы, если на его основе планируется создать несколько групп томов. Если есть такой диск с несколькими разделами, и разделы принадлежат одной группе томов, при создании томов с чередованием уделите особое внимание тому, какие разделы будут включены в логический том.

2.2. ГРУППА ТОМОВ

Физические тома объединяются в группы, что позволяет создать единое дисковое пространство, из которого будет выделяться место для логических томов.

В пределах группы пространство разделяется на блоки фиксированного размера — экстенты. Размер экстента является минимальным размером, который может быть выделен тому. На уровне физических томов используется понятие физических экстентов.

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

2.3. ЛОГИЧЕСКИЙ ТОМ

Группа состоит из логических томов. Существует три типа логических томов: *линейные*, *с чередованием* и *зеркальные*.

2.3.1. Линейный том

Линейный том объединяет несколько физических томов. Например, при наличии двух дисков размером 60 гигабайт можно создать логический том размером 120 гигабайт.

При создании линейного тома ему выделяется непрерывный диапазон физических экстентов. Например, как иллюстрирует Рисунок 2.2, «Организация экстентов», логические экстенты с 1 до 99 могут принадлежать одному физическому тому, а логические экстенты с 100 до 198 — второму. С точки зрения приложения существует только одно устройство размером 198 экстентов.

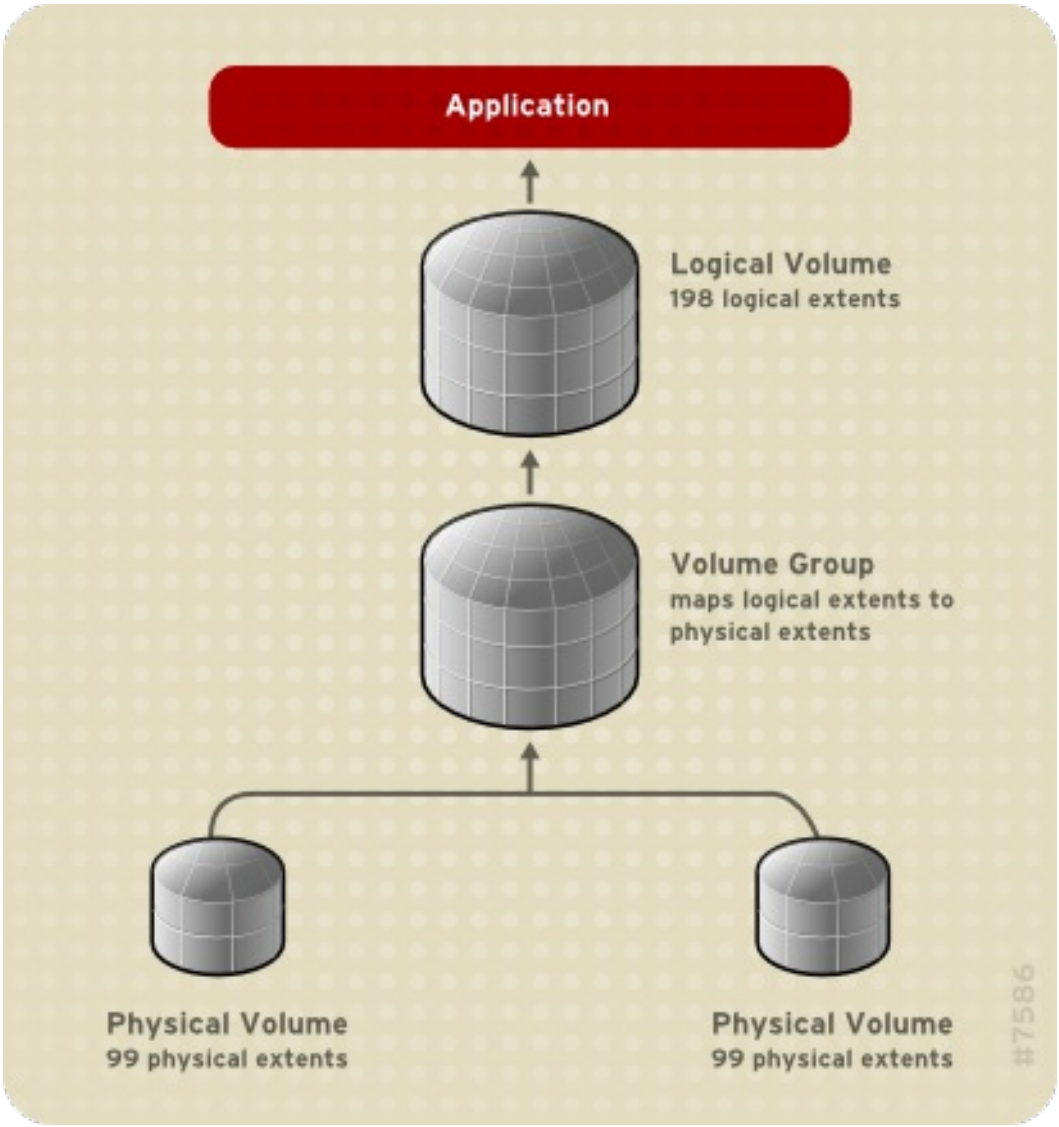


Рисунок 2.2. Организация экстентов

Физические тома, на основе которых строится логический том, могут быть разных размеров.

Рисунок 2.3, «Линейный том с физическими томами разного размера» демонстрирует группу VG1, размер физических экстентов которой равен 4 МБ. Эта группа включает физические тома **PV1** и **PV2**, которые в свою очередь разделены на блоки размером 4 МБ (в соответствии с размером экстентов). Так, **PV1** включает 100 экстентов (общий размер — 400 МБ), а **PV2** — 200 экстентов (общий размер — 800 МБ). На их основе можно создать линейный том размером 4—1200 МБ, содержащий от 1 до 300 экстентов. В этом примере том **LV1** содержит 300 экстентов.

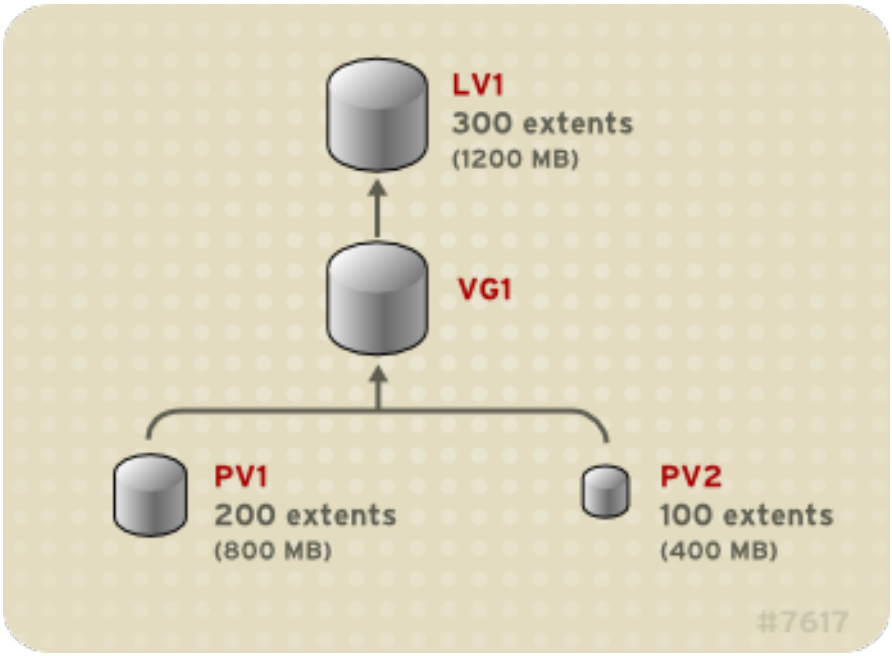


Рисунок 2.3. Линейный том с физическими томами разного размера

На основе доступных физических экстентов можно создать один или несколько логических томов произвольного размера. Рисунок 2.4, «Несколько логических томов» демонстрирует ту же группу, что и Рисунок 2.3, «Линейный том с физическими томами разного размера», но в этом примере в группе создано два логических тома — **LV1** (250 экстентов, общий размер — 1000 МБ) и **LV2** (50 экстентов, общий размер — 200 МБ).

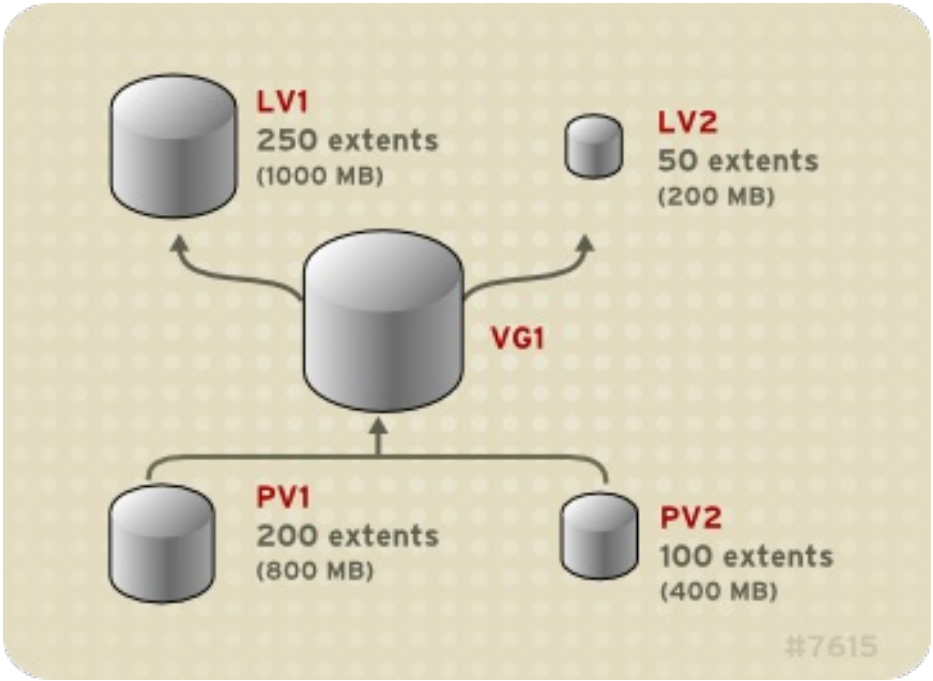


Рисунок 2.4. Несколько логических томов

2.3.2. Том с чередованием

При записи данных в логический том файловая система размещает их в физических томах в его основе. Чередование позволяет повысить производительность при выполнении большого объема последовательных операций ввода-вывода.

При чередовании данные последовательно распределяются между заранее определенным числом физических томов, поэтому операции ввода и вывода могут выполняться параллельно. В некоторых случаях производительность даже может сравниться с линейной организацией.

Приведенная ниже схема демонстрирует чередование данных между тремя физическими томами:

- 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 чередования не может превышать размер экстента.

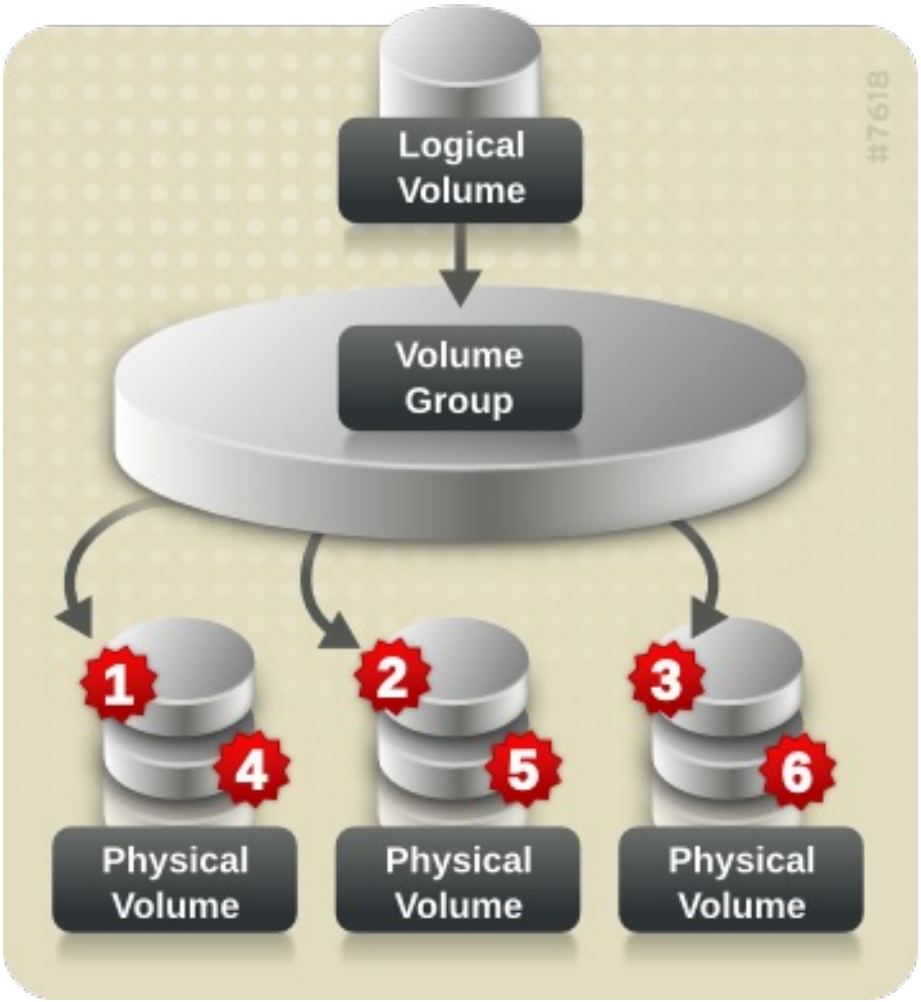


Рисунок 2.5. Чередование данных между тремя физическими томами

В конец существующего набора томов с чередованием можно добавлять дополнительные физические тома. Прежде чем приступить к увеличению размера, необходимо убедиться, что физические тома обладают достаточным свободным пространством. К примеру, если

используется двухстороннее чередование для целой группы томов, одного дополнительного тома будет недостаточно — надо добавить как минимум два физических тома (см. Раздел 4.4.12.1, «Увеличение тома с чередованием»).

2.3.3. Зеркальный том

Зеркало содержит устройства с идентичными копиями данных. При записи данных на одно устройство их копия также записывается на другое, что облегчает восстановление в случае выхода из строя одного из устройств. В случае сбоя одной составляющей зеркала логический том будет преобразован в линейный и продолжит работу.

При создании зеркала LVM записывает копию данных в отдельный физический том. LVM позволяет создать несколько зеркал для логических томов.

LVM подразделяет зеркалируемое устройство на секции, размер которых обычно составляет 512 КБ и ведет журнал синхронизации секций с зеркалами. Журнал может храниться на диске (в таком случае он будет сохраняться между перезагрузками) или временно находиться в памяти.

Рисунок 2.6, «Зеркальный логический том» демонстрирует логический том с одним зеркалом. Его журнал хранится на диске.

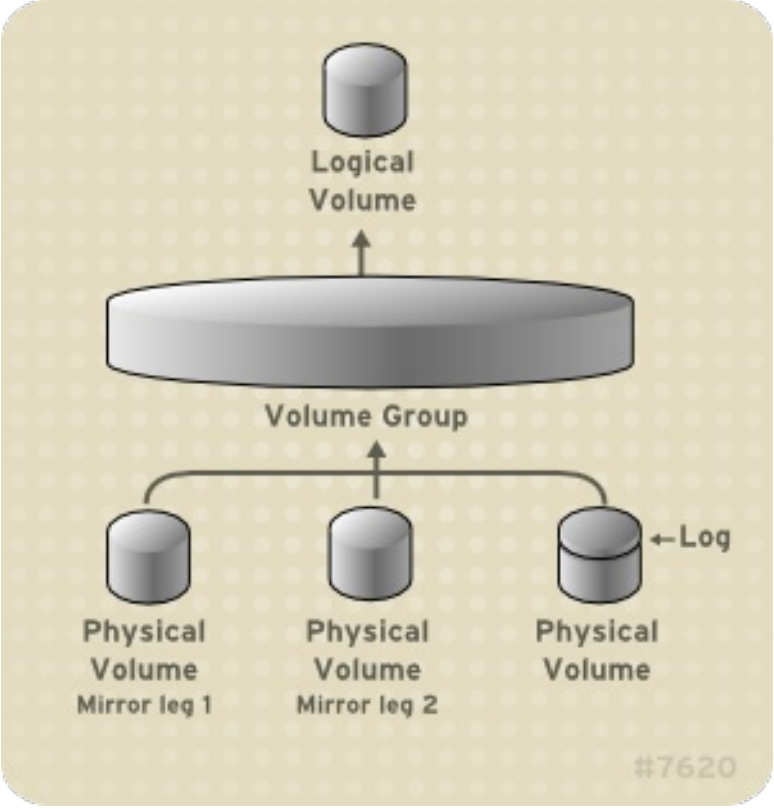


Рисунок 2.6. Зеркальный логический том

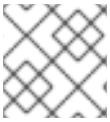
Раздел 4.4.3, «Создание зеркальных томов» содержит информацию о создании и изменении зеркал.

2.3.4. Снимки

LVM позволяет создавать снимки — виртуальные образы устройств в любой момент, не прибегая к остановке служб. Если исходное устройство было изменено после создания снимка, будет создана копия измененных данных, чтобы впоследствии можно было воссоздать состояние устройства.

ПРИМЕЧАНИЕ

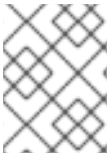
Создание снимков в кластерных группах томов не поддерживается.



ПРИМЕЧАНИЕ

Создание снимков для зеркальных логических томов не поддерживается.

Поскольку копируются только те данные, которые были изменены уже после создания снимка, для хранения снимков не требуется много пространства. Например, для хранения снимка редко обновляемого устройства будет достаточно 3-5% от его исходного размера.



ПРИМЕЧАНИЕ

Снимки не являются полнофункциональными резервными копиями, а всего лишь виртуальными. Поэтому заменять резервное копирование они не могут.

Размер снимка определяет объем пространства, который должен быть отведен для хранения будущих изменений. Например, если том был полностью перезаписан после создания снимка, размер копии изменений будет равен размеру исходного тома. Поэтому при расчете пространства для снимка следует учитывать ожидаемый объем изменений. Так, для снимка практически неизменяемого тома **/usr** потребуется меньше пространства, чем для **/home**, который подвергается изменениям довольно часто.

Если снимок полностью заполнен, он не сможет регистрировать будущие изменения исходного тома. Рекомендуется периодически проверять размер снимка и увеличивать его при необходимости. И наоборот, если вы обнаружили, что размер снимка слишком большой, можно его уменьшить и освободить место.

При создании файловой системы снимка права чтения и записи к исходному тому сохраняются. Если же часть снимка изменена, она будет отмечена и в дальнейшем уже не будет копироваться из исходного тома.

Когда же используются снимки?

- Если необходимо создать резервную копию логического тома, не останавливая систему, данные в которой постоянно обновляются.
- Для проверки целостности файловой системы исходного тома можно выполнить **fsck** в файловой системе снимка.
- Так как снимок доступен для чтения и записи, в его рамках можно тестировать приложения. При этом действительные данные остаются нетронутыми.
- В окружении виртуализации снимки виртуальных систем могут создаваться для их последующей модификации и создания на их основе других гостей. *Руководство по виртуализации Red Hat Enterprise Linux* содержит подробную информацию.

Раздел 4.4.4, «Создание снимков» содержит информацию о создании и изменении снимков

Начиная с Red Hat Enterprise Linux 6, команда **lvconvert** включает параметр **--merge**, предназначенный для слияния снимка с исходным томом. Это может использоваться для создания точки восстановления системы. Восстановленному путем слияния логическому тому

будет присвоено имя исходного тома, его вспомогательный номер и UUID, а снимок будет удален (см. Раздел 4.4.5, «Объединение снимка с оригиналом»).

ГЛАВА 3. ОБЗОР АДМИНИСТРИРОВАНИЯ LVM

Данная глава содержит обзор настройки логических томов LVM. Глава 5, *Примеры конфигурации LVM* рассматривает более детальные примеры.

Глава 4, *Администрирование LVM в текстовом режиме* включает описание команд, используемых для администрирования LVM. Глава 7, *Администрирование LVM в графическом режиме* содержит обзор графического интерфейса.

3.1. СОЗДАНИЕ ТОМОВ LVM В КЛАСТЕРЕ

Для создания логических томов в кластерном окружении используется набор расширений CLVM, которые позволяют управлять общим хранилищем кластера. Для работы CLVM необходимо, чтобы в системе работали комплекты высокой готовности и отказоустойчивого хранилища. В процессе загрузки на всех компьютерах в кластере должна быть запущена служба **clvmd** (см. Раздел 1.4, «CLVM»).

Процесс создания логических томов LVM в кластерном окружении аналогичен созданию томов на отдельном компьютере. Используемые команды не отличаются, и при работе с графическим интерфейсом ничего не меняется. Для активации созданных томов требуется, чтобы была активна кластерная инфраструктура, а для кластера определен кворум.

Для блокирования в пределах кластера потребуется внести изменения в **lvm.conf**. Информацию о настройке кластерного блокирования можно найти в самом файле **lvm.conf** (см. Приложение B, *Файлы конфигурации LVM*).

По умолчанию логические тома, созданные в общем хранилище с помощью CLVM, будут доступны всем компьютерам, у которых есть доступ к этому хранилищу. Можно создать целые группы томов, устройства хранения которых будут видны только одному узлу в кластере. Также можно изменить статус группы с «локальной» на «кластерную» (см. Раздел 4.3.2, «Создание групп томов в кластере» и Раздел 4.3.7, «Изменение параметров группы томов»).

ПРЕДУПРЕЖДЕНИЕ

При создании групп томов в общем хранилище с помощью CLVM необходимо убедиться, что у всех компьютеров в кластере есть доступ к физическим томам в составе группы. Асимметричные конфигурации кластеров с неравномерными правами доступа к хранилищу не поддерживаются.

Информацию об установке комплекта высокой степени готовности и настройке кластера можно найти в руководстве по администрированию кластера.

Раздел 5.5, «Создание зеркального логического тома в кластере» содержит пример создания тома в кластере.

3.2. СОЗДАНИЕ ЛОГИЧЕСКОГО ТОМА

Далее приведен порядок действий при создании логического тома.

1. Инициализация физических томов, на основе которых будет создан логический том.
2. Создание группы томов.
3. Создание логического тома.

После создания логического тома можно создать и подключить файловую систему. Приводимые в данном документе примеры подразумевают использование файловой системы GFS2.

ПРИМЕЧАНИЕ

GFS2 может быть создана отдельно в одной системе или как часть кластерной конфигурации, но Red Hat Enterprise Linux 6 не поддерживает GFS2 в отдельных системах. Их поддержка Red Hat ограничивается подключением снимков кластерных файловых систем с целью создания резервных копий.

1. С помощью **gfs_mkfs** создайте файловую систему GFS2 на логическом томе.
2. С помощью **mkdir** создайте точку подключения. В кластерной системе точку подключения надо создать на каждом узле.
3. Подключите файловую систему. Для каждого узла в кластере можно добавить отдельную строчку в файле **fstab**.

Перечисленные действия также можно выполнить в окне графического интерфейса LVM.

Процесс создания томов не зависит от оборудования, так как информация о настройках LVM расположена в физических томах, а не на компьютере, где создается том. Использующие хранилище серверы обычно содержат локальные копии, но имеют возможность их воссоздания из содержимого физических томов. Физические тома можно подключить к другому серверу при условии совместимости их версий LVM.

3.3. УВЕЛИЧЕНИЕ ФАЙЛОВОЙ СИСТЕМЫ ЛОГИЧЕСКОГО ТОМА

Порядок действий для увеличения файловой системы логического тома.

1. Создайте новый физический том.
2. Добавьте его в группу томов, в которую входит увеличиваемый логический 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, шаги 1 и 2 можно опустить.

3.4. РЕЗЕРВНОЕ КОПИРОВАНИЕ ЛОГИЧЕСКОГО ТОМА

Резервные копии и архивы метаданных создаются автоматически при изменении настроек логического тома и группы (это можно отключить в файле **lvm.conf**). Копия метаданных по умолчанию сохраняется в **/etc/lvm/backup**, а архивы — в **/etc/lvm/archive**. Параметры в

lvm.conf определяют продолжительность хранения архивов в **/etc/lvm/archive** и число хранимых файлов. Ежедневная резервная копия должна включать содержимое каталога **/etc/lvm**.

Обратите внимание, что резервная копия метаданных не содержит системные данные и данные пользователя в логических томах.

Создать архив метаданных в **/etc/lvm/backup** можно с помощью команды **vgcfgbackup**. Восстановить метаданные можно с помощью **vgcfgrestore** (см. Раздел 4.3.12, «Создание резервной копии метаданных группы томов»).

3.5. ЖУРНАЛ СОБЫТИЙ

Вывод сообщений обрабатывается модулем журналирования. При этом события подразделяются на следующие категории:

- стандартный вывод или ошибка;
- syslog;
- файл журнала;
- внешняя функция журнала.

Уровни журналирования задаются в **/etc/lvm/lvm.conf** (см. Приложение B, *Файлы конфигурации LVM*).

ГЛАВА 4. АДМИНИСТРИРОВАНИЕ LVM В ТЕКСТОВОМ РЕЖИМЕ

В этой главе приведены наборы команд для решения некоторых практических задач управления LVM.

ПРИМЕЧАНИЕ

Для управления томами в кластерном окружении необходимо, чтобы выполнялся процесс **clvmd** (см. Раздел 3.1, «Создание томов LVM в кластере»).

4.1. ИСПОЛЬЗОВАНИЕ КОМАНД

Сначала стоит упомянуть об основных возможностях команд LVM.

Если аргумент команды содержит величину объема информации, единицы можно указать вручную. Если единицы не указаны, по умолчанию будут подразумеваться килобайты или мегабайты. Сами значения должны быть целыми числами.

По умолчанию регистр единиц не имеет значения. Например, M и m эквивалентны, а сами значения будут кратны 1024. Но если задан аргумент **--units**, то обозначение единиц в нижнем регистре будет означать, что они кратны 1024, а в верхнем — 1000.

Если команды в качестве аргументов принимают имена отдельных томов или целых групп, можно указать их полный путь. Например, том **lvol0** в группе **vg0** можно определить как **vg0/lvol0**. Если ожидается список групп, но при этом он не указан, по умолчанию будет подставлен список ВСЕХ групп томов. Если команда ожидает список томов, но при этом задана группа томов, будет выполнена подстановка всех логических томов в заданной группе. К примеру, команда **lvdisplay vg0** покажет список всех логических томов в группе **vg0**.

Аргумент **-v** усиливает степень подробности вывода. Его можно использовать каскадно для увеличения детализации. К примеру, стандартный вывод **lvcreate** выглядит так:

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

lvcreate с аргументом **-v**:

```
# 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
```

Степень подробности вывода можно усилить, указав **-vv**, **-vvv** или даже **-vvvv**. Максимально подробный вывод будет достигнут при указании **-vvvv**. Следующий пример демонстрирует лишь несколько первых строк вывода **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 просмотра краткой справки по интересующей команде используется аргумент **--help**.

```
commandname --help
```

man открывает справочную страницу команды:

```
man commandname
```

Например, **man lvm** покажет информацию об LVM.

В рамках LVM обращение к объектам осуществляется при помощи уникального идентификатора UUID, который назначается при создании объектов. Например, представим, что из группы томов удален физический том **/dev/sdf**. При повторном его подключении он будет определен как **/dev/sdk**. LVM корректно его определит, так как для его идентификации используется UUID, а не имя устройства. Раздел 6.4, «Восстановление метаданных физического тома» содержит информацию о присвоении идентификаторов физическим томам при их создании.

4.2. УПРАВЛЕНИЕ ФИЗИЧЕСКИМИ ТОМАМИ

В этой секции рассматриваются команды управления физическими томами.

4.2.1. Создание физических томов

Далее рассматриваются команды, используемые для создания физических томов.

4.2.1.1. Настройка типов разделов

Если физический том занимает весь диск, на этом диске НЕ должно быть таблицы разделов. Разделам DOS соответствует идентификатор 0x8e (задается с помощью **fdisk**, **cfdisk** или их аналога). При удалении таблицы разделов данные на диске будут удалены автоматически. Существующую таблицу можно удалить путем заполнения первого сектора нулями:

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

4.2.1.2. Инициализация физических томов

pvcreate позволяет инициализировать блочное устройство как физический том. 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 отдельных разделов применяется команда **pvcreate**. Следующий пример инициализирует **/dev/hdb1** как физический том для дальнейшего включения в логический том LVM.

```
pvcreate /dev/hdb1
```

4.2.1.3. Поиск блочных устройств

С помощью **lvmdiskscan** можно выполнить поиск блочных устройств для создания физических томов на их основе. 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. Просмотр физических томов

Для просмотра информации о физических томах LVM используются команды **pvs**, **pvdisplay** и **pvscan**.

pvs позволяет настроить формат вывода, показывая по одному тому в каждой строке (см. Раздел 4.8, «Настройка отчетов LVM»).

pvdisplay формирует подробный отчет для каждого физического тома, включая информацию о размере, экстентах, группе томов и пр. Формат вывода фиксирован.

Пример вывода **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 проверяет все поддерживаемые блочные устройства в системе на предмет наличия физических томов.

Пример:

```
# 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]
```

Чтобы выборочно проверить устройства, можно создать фильтр в **lvm.conf** (см. Раздел 4.5, «Определение устройств LVM с помощью фильтров»).

4.2.3. Запрет выделения пространства физического тома

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

Следующая команда запрещает выделение экстентов на **/dev/sdk1**:

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

Аргументы **-xy** разрешат выделение экстентов там, где раньше это было запрещено.

4.2.4. Изменение размера физического тома

pvresize изменяет размер физического тома, если изменился размер блочного устройства в его основе.

4.2.5. Удаление физических томов

pvremove удаляет устройство, в котором больше нет необходимости, заполняя его метаданные нулями.

Если удаляемый том входит в состав группы, сначала исключите его из группы с помощью **vgreduce** (см. Раздел 4.3.6, «Удаление физических томов из группы»).

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

4.3. УПРАВЛЕНИЕ ГРУППАМИ

В этой секции описываются команды администрирования групп томов.

4.3.1. Создание групп томов

Команда **vgcreate** создаст новую группу томов с заданным именем и добавит в нее как минимум один физический том.

Следующая команда создаст группу **vg1**, включающую в свой состав физические тома **/dev/sdd1** и **/dev/sde1**.

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

Если группа создается на основе физических томов, их пространство по умолчанию подразделяется на экстенты размером 4 мегабайт. Этот размер и определяет минимальную величину изменения размера логического тома. Число экстентов не оказывает влияния на эффективность ввода-вывода для логического тома.

Размер экстента можно задать с помощью аргумента **-s** команды **vgcreate**. Дополнительно можно ограничить число томов в группе — для этого используется **-p** (для физических томов) или **-l** (для логических томов).

По умолчанию группа томов выделяет физические экстенты согласно стандартным правилам, то есть чередующиеся секции не будут располагаться на одном физическом томе. Стандартная политика распределения обозначена как **normal**. Аргумент **--alloc** команды **vgcreate** позволяет изменить значение на **contiguous**, **anywhere** или **cling**.

contiguous требует, чтобы новые экстенты размещались следом за существующими. Если есть достаточное число экстентов, чтобы удовлетворить запрос выделения пространства, а политика **normal** не может их использовать, можно выбрать **anywhere**. Экстенты будут заняты, даже если при этом чередующиеся сегменты будут включены в один том, тем самым оказав отрицательный эффект на производительность. Команда **vgchange** позволяет изменить метод выделения экстентов.

Раздел 4.4.12.2, «Увеличение логического тома в режиме **cling**» содержит информацию о **cling** и выборе томов с помощью тегов.

В целом, любой метод, отличный от **normal**, используется только в особых, нестандартных случаях.

Файлы групп томов и их логических томов расположены в **/dev**. Путь:

-

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

Например, если есть группы **myvg1** и **myvg2** с томами **lvo1**, **lvo2** и **lvo3** в составе каждой из них, им будет соответствовать шесть файлов:

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

Максимально допустимый 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 **vgcreate**.

По умолчанию группы, созданные с помощью CLVM в общем хранилище, доступны всем компьютерам, обладающим доступом к этому хранилищу. В принципе, можно создать локальные группы томов, которые будут видны только одному компьютеру в кластере. Для этого используется **vgcreate** с опцией **-c n**.

Следующая команда создаст локальную группу **vg1** с томами **/dev/sdd1** и **/dev/sde1**.

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

vgchange -c позволяет изменить тип группы с локальной на кластерную (см. Раздел 4.3.7, «Изменение параметров группы томов»).

Тип группы можно проверить с помощью команды **vgs**. Приведенная далее команда вернет атрибуты групп **VolGroup00** и **testvg1**. В этом примере **testvg1** является кластерной группой, а **VolGroup00** — нет, о чем свидетельствует значение **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
```

Раздел 4.3.4, «Просмотр групп томов», Раздел 4.8, «Настройка отчетов LVM» и справочная страница **vgs** содержат подробную информацию о **vgs**.

4.3.3. Добавление физических томов в группу

vgextend позволяет добавить физические тома в существующую группу, тем самым увеличив ее размер.

Команда добавления тома **/dev/sdf1** в группу **vg1**:

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

4.3.4. Просмотр групп томов

vgs и **vgdisplay** позволяют просмотреть информацию о группах томов LVM.

Основной целью команды **vgscan** является поиск групп на дисках и создание файла кэша LVM, но с ее помощью можно дополнительно показать список групп томов (см. Раздел 4.3.5, «Поиск групп томов на дисках»).

Команда **vgs** выведет информацию для каждой группы в отдельной строке. Формат вывода можно изменять, что широко применяется в сценариях (см. Раздел 4.8, «Настройка отчетов LVM»).

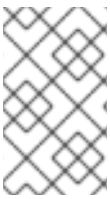
vgdisplay показывает параметры группы в фиксированном формате: размер, экстенты, число физических томов и пр. Приведенный далее пример демонстрирует вывод **vgdisplay** для группы **new_vg**. Если группа не указана, будет показана информация для всех групп.

```
# 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

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

LVM выполняет **vgscan** автоматически при загрузке системы, при запуске **vgcreate** и при обнаружении несоответствий LVM.



ПРИМЕЧАНИЕ

Если конфигурация оборудования была изменена, можно выполнить **vgscan** вручную. Это необходимо, например, при добавлении новых дисков в систему из SAN или при горячем подключении нового диска, отмеченного как физический том.

В **lvm.conf** можно определить фильтр для исключения устройств из проверки (см. Раздел 4.5, «Определение устройств LVM с помощью фильтров»).

Пример вывода **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

Команда **vgreduce** удаляет неиспользуемые физические тома из группы, тем самым уменьшая ее размер. Освобожденные физические тома могут быть удалены или добавлены в другие группы.

Прежде чем удалить физический том из группы, надо убедиться, что он не используется логическими томами. Это можно сделать с помощью **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, надо перенести данные в другой том (с помощью **pvmove**). После этого можно будет уменьшить размер тома с помощью **vgreduce**.

Следующая команда удалит том **/dev/hda1** из **my_volume_group**:

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

4.3.7. Изменение параметров группы томов

vgchange позволяет включить и отключить группы томов (см. Раздел 4.3.8, «Активация и деактивация групп томов»), а также изменить некоторые параметры.

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

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

Справочная страница **vgchange**(8) содержит описание доступных параметров.

4.3.8. Активация и деактивация групп томов

При создании группы томов она будет активна по умолчанию, то есть логические тома в ее составе будут сразу доступны.

Существует множество причин, по которым может понадобиться отключить группу томов. Для это цели служит аргумент **-a** (**--available**) команды **vgchange**.

Пример отключения группы **my_volume_group**:

```
vgchange -a n my_volume_group
```

Если включено кластерное блокирование, добавьте «e» для активации или деактивации группы на выбранном узле в кластере или «l» — только на локальном узле. Логические тома со снимком на одном узле всегда активируются отдельно.

Отключение отдельных логических томов осуществляется с помощью **lvchange** (см. Раздел 4.4.8, «Изменение параметров группы логических томов»). Раздел 4.7, «Активация логических томов на отдельных узлах кластера» содержит информацию об активации логических томов на индивидуальных 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. Разбиение группы томов

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

Логические тома, принадлежащие разным группам, не могут быть разделены. Каждый логический том должен полностью принадлежать физическим томам, входящим в состав либо уже существующей, либо новой группы томов. Для принудительного разбиения используется **pvmove**.

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

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

4.3.11. Объединение групп томов

Для объединения двух групп в одну используется команда **vgmerge**. Возможно объединение двух неактивных томов, если размеры их физических экстентов равны и общий объем физических и логических томов обеих групп не превышает допустимый размер полученной группы.

Следующая команда добавит неактивную группу **my_vg** в группу **databases** с выводом отчета 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. Это можно отключить в **lvm.conf**. Резервная копия метаданных сохраняется в **/etc/lvm/backup**, а архивы — в **/etc/lvm/archives**. Команда **vgcfgbackup** позволяет создать резервную копию метаданных в **/etc/lvm/backup** в любое время.

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

Раздел 6.4, «Восстановление метаданных физического тома» содержит пример использования **vgcfgrestore** для восстановления метаданных.

4.3.13. Переименование группы томов

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

Обе приведенные 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 **vgexport** и **vgimport** можно осуществлять перемещение целых групп томов между системами.

vgexport закрывает доступ к неактивной группе томов, что позволяет отключить ее физические тома, а **vgimport** снова разрешает доступ.

Порядок действий при переносе группы томов между системами:

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

2. Чтобы отметить группу томов как неактивную, выполните команду **vgchange** с аргументом **-a n**.

3. **vgexport** позволяет экспортировать группу, что закроет к ней доступ из системы, откуда она будет удалена.

   После завершения экспортирования группы, 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]
     ...
   ```

   После следующего выключения системы можно отсоединить диски, на основе которых была построена эта группа томов, и переподключить их в новую систему.

4. Подключив диски в новую систему, выполните **vgimport** для импорта группы томов, тем самым открыв к ней доступ из системы.

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

6. Подключите 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**, необходимые для работы активных логических томов. При этом будут созданы недостающие специальные файлы, а лишние файлы будут удалены.

Аналогичного результата можно добиться с помощью **vgscan**, указав аргумент **--mknodes**.

4.4. УПРАВЛЕНИЕ ЛОГИЧЕСКИМИ ТОМАМИ

В этой секции будут рассмотрены команды управления логическими томами.

4.4.1. Создание линейных логических томов

Новый логический том можно создать с помощью **lvcreate**. Если имя тома не указано, по умолчанию будет использоваться обозначение **lvol#**, где # — внутренний номер логического тома.

Для нового линейного тома выделяются свободные экстенты из группы физических томов. Обычно логические тома используют все доступное пространство. При изменении размера логического тома экстенты в их основе будут переорганизованы или освобождены.

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

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

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

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

Далее из свободных экстентов в группе **vg0** будет создан логический том **gfslv** размером 50 гигабайт.

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

lvcreate -l позволяет задать размер логического тома в экстентах. Также можно указать процентную часть группы томов, используемую для создания логического тома. Приведенная ниже команда создаст том **mylv**, занимающий 60% группы **testvol**.

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

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

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

С помощью **-l** можно также создать логический том, который будет использовать целую группу томов. Другой способ создания логического тома, занимающего всю группу, состоит в передаче команде **lvcreate** значения «Total PE», найденного с помощью **vgdisplay**.

Пример создания логического тома **mylv**, который полностью заполнит группу **testvg**:

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

При создании логического тома на основе физических томов стоит учесть вероятность того, что в будущем физический том может потребоваться удалить (см. Раздел 4.3.6, «Удаление физических томов из группы»).

Чтобы создать логический том на основе определенных физических томов в группе, надо их перечислить в командной строке **lvcreate**. Далее будет создан логический том **testlv** на основе физического тома **/dev/sdg1** в группе **testvg**.

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

Можно указать, какие экстенты будут задействованы для образования логического тома. В следующем примере будет создан линейный том, в состав которого войдут экстенты 0 — 24 физического тома **/dev/sda1** и 50 — 124 тома **/dev/sdb1**. Оба физических тома принадлежат группе **testvg**.

```
lvcreate -l 100 -n testlv testvg /dev/sda1:0-24 /dev/sdb1:50-124
```

Следующий пример демонстрирует создание линейного тома на основе экстентов 0 — 25 физического тома **/dev/sda1** и затем продолжая с экстента 100.

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

По умолчанию правила выделения экстентов наследуются от группы томов (**inherit**). Это можно изменить с помощью **lvchange** (см. Раздел 4.3.1, «Создание групп томов»).

4.4.2. Создание томов с чередованием

Для обработки больших объемов последовательных операций чтения и записи рекомендуется использовать логические тома с чередованием, так как при этом можно достичь высокой эффективности чтения и записи (см. Раздел 2.3.2, «Том с чередованием»).

При создании тома с чередованием число сегментов (физических томов) задается с помощью аргумента **-i** команды **lvcreate**. Это значение не должно превышать общее число физических томов в группе (за исключением использования **--alloc anywhere**).

Если размеры физических устройств, на основе которых создан логический том, отличаются, то максимальный объем тома с чередованием будет определяться размером наименьшего устройства. Например, если используются два физических тома, 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 том **gfslv** размером 50 ГБ на основе двух физических томов в группе **vg0**; при этом размер сегмента чередования будет равен 64 КБ.

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

Так же как и в случае с линейными томами, можно явно указать число выделяемых экстентов. В следующем примере на основе двух физических томов будет создан том **stripelv**, содержащий 100 экстентов. Этот том будет входить в состав группы **testvg** и занимать секторы 0 — 49 тома **/dev/sda1** и 50 — 99 тома **/dev/sdb1**.

```
# lvcreate -l 100 -i2 -nstripelv testvg /dev/sda1:0-49 /dev/sdb1:50-99
  Using default stripesize 64.00 KB
  Logical volume "stripelv" created
```

4.4.3. Создание зеркальных томов

ПРИМЕЧАНИЕ

Создание зеркального логического тома в кластере осуществляется по тем же правилам, что и на отдельном компьютере. Отличия состоят в том, что в кластере должна работать кластерная инфраструктура, кластер должен обладать кворумом, а тип блокирования в **lvm.conf** должен разрешать кластерное блокирование. Раздел 5.5, «Создание зеркального логического тома в кластере» содержит примеры создания зеркального тома в кластере.

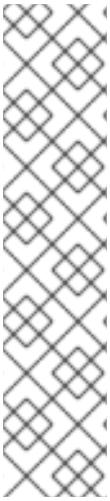
Одновременные попытки выполнения команд создания и преобразования зеркал с разных компьютеров в кластере могут привести к их задержкам и сбоям. Поэтому команды создания зеркал рекомендуется выполнять с одного компьютера.

При создании зеркального тома необходимо указать число копий, для чего служит аргумент **-m** команды **lvcreate**. Так, **-m1** создаст одно зеркало, то есть в файловой системе будет две копии — линейный логический том и его копия. Аналогичным образом, если указать **-m2**, будет создано два зеркала (всего три копии).

Ниже приведен пример создания тома **mirrorlv** размером 50 гигабайт с одним зеркалом. Пространство будет выделено из группы **vg0**.

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

Копируемое устройство разбивается на сегменты, размер которых по умолчанию составляет 512 килобайт. Аргумент **-R** команды **lvcreate** позволяет указать другой размер в мегабайтах, или его можно изменить напрямую с помощью параметра **mirror_region_size** в файле **lvm.conf**.



ПРИМЕЧАНИЕ

В силу ограничений кластерной инфраструктуры, размер сегментов зеркала, размер которого превышает 1.5 ТБ, должен быть больше 512 МБ. В противном случае может нарушиться функциональность LVM.

Размер сегмента рассчитывается так: размер зеркала в терабайтах округляется до следующего значения, кратного 2. Так, например, для зеркала размером 1.5 ТБ можно указать **-R 2**, для 3 ТБ — **-R 4**, для 5 ТБ — **-R 8**.

Следующая команда создаст зеркальный том с сегментами размером 2 МБ:

```
lvcreate -m1 -L 2T -R 2 -n mirror vol_group
```

LVM поддерживает краткий журнал синхронизации сегментов с зеркалами. По умолчанию журнал хранится на диске и сохраняется между перезагрузками. Если же требуется, чтобы журнал находился в памяти, используйте аргумент **--corelog**. Это отменяет необходимость в устройстве журналирования, но в то же время требует полной синхронизации зеркала при каждой перезагрузке.

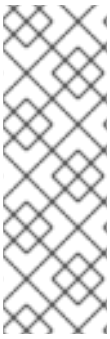
Ниже будет создан логический том **ondiskmirvol** с одним зеркалом в группе **bigvg**. Размер тома равен 12 МБ, а журнал зеркала хранится в памяти.

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

Журнал зеркала будет создан на отдельном устройстве. Возможно создание журнала на том же устройстве, где расположена секция зеркала, — для этого служит аргумент **--alloc anywhere** команды **vgcreate**. Это может отрицательно сказаться на производительности, но позволит создать зеркало даже при наличии всего двух устройств.

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

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



ПРИМЕЧАНИЕ

В кластерном окружении управление журналами зеркал осуществляется компьютером с наименьшим кластерным идентификатором. Если отдельные узлы кластера потеряли доступ к устройству, где хранится журнал, функциональность зеркала не нарушится при условии, что журнал доступен компьютеру с наименьшим идентификатором. Но если компьютер с наименьшим идентификатором потеряет доступ к журналу, процедура восстановления начнется, даже если журнал доступен с других узлов.

Для создания зеркала журнала служит аргумент **--mirrorlog mirrored**. Приведенная ниже команда создаст том **twologvol** с одним зеркалом в группе **bigvg**. Размер тома равен 12 МБ, а журналы зеркал хранятся на отдельных устройствах.

```
# lvcreate -L 12MB -m1 --mirrorlog mirrored -n twologvol bigvg
  Logical volume "twologvol" created
```

Параметр **--alloc anywhere** позволяет хранить копии журнала на том же устройстве, где расположено звено зеркала. Как уже упоминалось, это может отрицательно сказаться на производительности, но позволит создать дополнительный журнал, даже если общее число журналов превышает число физических дисков.

Составляющие зеркала синхронизируются в момент его создания. В зависимости от их размера процесс синхронизации может занять некоторое время. Если создается новое зеркало, синхронизация которого необязательна, можно добавить параметр **nosync**.

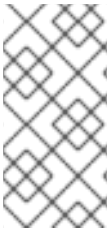
Можно явно указать, на каких устройствах будут располагаться журналы, и какие экстенты будут заняты зеркалом. Чтобы разместить журнал на конкретном диске, укажите всего ОДИН экстент на этом диске. LVM игнорирует порядок, в котором перечислены устройства. Если в списке присутствуют физические устройства, они и будут использоваться для выделения пространства; занятые физические экстенты будут пропущены.

Далее приведен пример создания зеркального тома **mirrorlv** размером 500 МБ с одним зеркалом и одним журналом. Том будет создан в группе **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 том **mirrorlv** размером 500 МБ с одним зеркалом. Том будет создан в составе группы **vg0**. Одна часть зеркала будет занимать экстенты 0 — 499 на **/dev/sda1**, вторая — экстенты 0 — 499 на **/dev/sdb1**, а журнал будет храниться на **/dev/sdc1**, начиная с нулевого экстента. Размер экстента равен 1 МБ. Если указанные экстенты уже заняты, они будут пропущены.

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



ПРИМЕЧАНИЕ

Начиная с Red Hat Enterprise Linux 6.1, допускается объединять массивы RAID0 и RAID1 в один логический том. Если при создании логического тома указано число зеркал (**--mirrors X**) и число секций (**--stripes Y**), будет создано зеркальное устройство с чередующимися составляющими.

4.4.3.1. Правила работы при сбое зеркального тома

Поведение зеркального тома в случае сбоя устройства можно определить с помощью параметров **mirror_image_fault_policy** и **mirror_log_fault_policy** в секции **activation** файла **lvm.conf**. Если этим параметрам присвоено значение **remove**, неисправные устройства будут исключены из системы. Значение **allocate** означает, что после исключения устройства будет предпринята попытка выделения пространства на другом устройстве. Если пространство выделить не удалось, **allocate** работает так же как **remove**.

По умолчанию **mirror_log_fault_policy** имеет значение **allocate**, то есть синхронизация поддерживается между перезагрузками и в случае сбоя. Если задан режим **remove**, то в случае сбоя устройства, где хранится журнал, зеркало будет поддерживать журнал в памяти без

сохранения его между перезагрузками. Как следствие, после сбоя зеркало надо будет синхронизировать полностью.

По умолчанию **mirror_image_fault_policy** имеет значение **remove**, то есть если после сбоя звена зеркала осталась только одна копия, зеркало будет преобразовано в другой тип. **allocate** приводит к повторной синхронизации устройств с сохранением характеристик зеркала, что может занять некоторое время.

ПРИМЕЧАНИЕ

При выходе из строя устройства в составе зеркала начнется процесс восстановления. Сначала проблемные устройства будут исключены, что может привести к тому, что зеркало превратится в обычное линейное устройство. Далее начнется замена этих устройств при условии, что параметр **mirror_log_fault_policy** имеет значение **allocate**. Однако нет гарантий, что на втором этапе не будут выбраны устройства, которые раньше служили причиной сбоя зеркала.

Раздел 6.3, «Восстановление после сбоя зеркала» содержит дальнейшую информацию о восстановлении зеркал вручную.

4.4.3.2. Разбиение образа зеркального тома

Образ зеркального тома можно разделить на части с целью создания нового тома. Для этой цели служит **lvconvert --splitmirrors** с указанием числа копий. Аргумент **--name** задает имя нового тома.

В следующем примере от тома **vg/lv** будет отделен новый том **copy**, содержащий две копии зеркала. LVM сам выбирает, какие именно устройства будут отделены.

```
lvconvert --splitmirrors 2 --name copy vg/lv
```

Можно отдельно указать устройства. Ниже приведена команда отделения нового тома **copy** от **vg/lv**. Составляющие зеркала будут располагаться на двух устройствах: **/dev/sdc1** и **/dev/sde1**.

```
lvconvert --splitmirrors 2 --name copy vg/lv /dev/sd[ce]1
```

4.4.3.3. Восстановление зеркального устройства

Команда **lvconvert --repair** поможет восстановить зеркало после сбоя диска. **lvconvert --repair** выполняется в интерактивном режиме, предлагая подтвердить замену сбойных устройств.

- Чтобы пропустить запросы подтверждения и заменить все проблемные устройства, добавьте параметр **-y**.
- Чтобы пропустить все запросы и не заменять устройства, добавьте параметр **-f**.
- Чтобы пропустить запросы, но при этом включить доступные режимы замены для зеркального образа и журнала, укажите аргумент **--use-policies**. Он получает значения **mirror_log_fault_policy** и **mirror_device_fault_policy** из **lvm.conf**.

4.4.3.4. Изменение конфигурации зеркальных томов

Преобразование зеркального тома в линейный и наоборот осуществляется с помощью команды **lvconvert**. Она также позволяет изменить параметры существующих логических томов.

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

Если по какой-то причине одна из составляющих зеркала перестала работать, LVM осуществит преобразование тома в линейный, чтобы сохранить доступ к его содержимому. После замены звена зеркало можно восстановить с помощью **lvconvert** (см. Раздел 6.3, «Восстановление после сбоя зеркала»).

Следующая команда преобразует линейный 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:

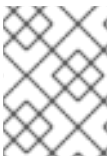
```
lvconvert -m0 vg00/lvol1
```

4.4.4. Создание снимков

Команда **lvcreate** с аргументом **-s** позволяет создать том-снимок с разрешениями записи.

ПРИМЕЧАНИЕ

Создание снимков в кластерной группе томов не поддерживается. Но в Red Hat Enterprise Linux 6.1 можно отдельно активировать кластерный том и создать его снимок (см. Раздел 4.7, «Активация логических томов на отдельных узлах кластера»).



ПРИМЕЧАНИЕ

В Red Hat Enterprise Linux 6.1 добавлена поддержка снимков для зеркальных логических томов.

Ниже будет создан снимок **/dev/vg00/snap** для тома **/dev/vg00/lvol1** размером 100 МБ. Если исходный том содержит файловую систему, можно будет подключить снимок в любой каталог, тем самым получив доступ к содержимому файловой системы для создания резервной копии, в то время как исходная файловая система будет продолжать обновление.

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

Выполнив **lvdisplay** для логических томов, можно получить список его снимков и их статус (активный или неактивный).

Ниже показано состояние тома **/dev/new_vg/lvol0**, для которого 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 показывает исходный том и процентную часть снимка. Следующий пример демонстрирует стандартный вывод **lvs** для системы с логическим томом **/dev/new_vg/lvol0** с соответствующим ему 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 по мере изменения исходного тома, важно следить за его заполнением. Для этого можно использовать команду **lvs**. При заполнении снимка на 100% он будет потерян, так как запись в неизменяемые участки исходного тома повредит сам снимок.

4.4.5. Объединение снимка с оригиналом

В Red Hat Enterprise Linux 6 добавлена возможность объединения снимка с его исходным томом с помощью **lvconvert --merge**. Если не открыт ни снимок, ни исходный том, их слияние будет выполнено автоматически. В противном случае слияние начнется, как только один из томов станет активен или по факту закрытия обоих томов. Объединение снимка с томом, который не может быть закрыт (например, если он содержит корневую файловую систему), будет отложено до момента следующей активации исходного тома. Полученному в результате объединения тому будет присвоено имя исходного тома, его вторичный номер и UUID. Во время слияния все операции чтения и записи исходного тома будут представлены так, как будто они предназначаются снимку. После завершения снимок будет удален.

Следующая команда объединит снимок **vg00/lvol1_snap** с его исходным томом.

```
lvconvert --merge vg00/lvol1_snap"
```

В команде можно указать несколько снимков или их метки. Так, в приведенном ниже примере томам **vg00/lvol1**, **vg00/lvol2** и **vg00/lvol3** соответствует метка **@some_tag**. Команда последовательно осуществит слияние их снимков. Дополнительный параметр **--background** начнет объединение снимков параллельно.

```
lvconvert --merge @some_tag"
```

Приложение C, *Теги объектов LVM* содержит сведения о присвоении меток объектам LVM. Информацию о **lvconvert --merge** можно найти на справочной странице **lvconvert**(8).

4.4.6. Постоянные номера устройств

Старший и младший номер присваивается динамически при загрузке модуля. Производительность некоторых приложений оптимальна при условии, что оба номера остаются неизменными. Их можно задать с помощью команд **lvcreate** и **lvchange**:

```
--persistent y --major старший --minor младший
```

В качестве младшего номера используйте достаточно большое число, чтобы гарантировать, что этот номер не будет назначен устройству автоматически.

Если файловая система подключается по NFS, то указание в файле экспорта параметра **fsid** поможет избежать необходимости в обеспечении постоянства номера устройства в пределах LVM.

4.4.7. Изменение размера логических томов

С помощью команды **lvreduce** можно уменьшить размер логического тома. Если том содержит файловую систему, сначала необходимо уменьшить ее размер, чтобы размер логического тома был эквивалентен размеру файловой системы.

Следующая команда уменьшает размер тома **lvol1** в группе **vg00** на три логических экстента.

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

4.4.8. Изменение параметров группы логических томов

С помощью команды **lvchange** можно изменить параметры логического тома. Полный список параметров можно найти на справочной странице **lvchange**(8).

lvchange также можно использовать для активации и деактивации логических томов. Одновременная активация ВСЕХ логических томов в группе осуществляется с помощью **vgchange** (см. Раздел 4.3.7, «Изменение параметров группы томов»).

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

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

4.4.9. Переименование логических томов

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

Обе приведенные команды изменят имя логического тома **lvold** в группе **vg02** на **lvnew**.

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

```
lvrename vg02 lvold lvnew
```

Раздел 4.7, «Активация логических томов на отдельных узлах кластера» содержит информацию об активации логических томов на отдельных узлах кластера.

4.4.10. Удаление логических томов

Удалить неактивный логический том можно с помощью команды **lvremove**. Сначала выполните **umount** для его отключения из файловой системы. В кластерном окружении необходимо выключить том перед его удалением.

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

```
[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 отключить отдельно с помощью **lvchange -an** — тогда предупреждение об удалении активного тома не появится.

4.4.11. Просмотр логических томов

Для просмотра сведений о логических томах используются команды **lvs**, **lvdisplay** и **lvscan**.

lvs позволяет настроить формат полученных данных и выводит по одному тому на каждой строке, что может применяться при создании сценариев (см. Раздел 4.8, «Настройка отчетов LVM»).

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

Следующая команда покажет атрибуты тома **lvol2** в группе **vg00**. Вывод также будет содержать список снимков выбранного тома и их статус (активен или неактивен).

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

lvscan покажет список логических томов в системе. Пример:

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

4.4.12. Увеличение размера логических томов

lvextend позволяет увеличить размер логического тома.

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

Пример увеличения логического тома **/dev/myvg/homevol** до 12 гигабайт:

```
# 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 в **/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
```

Аргумент **-l** позволяет указать число экстентов для добавления в том. Также можно указать процент от общего размера группы томов или незанятого пространства. Приведенная ниже команда увеличит том **testlv**, выделив ему все нераспределенное пространство 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
```

После увеличения тома необходимо увеличить размер его файловой системы.

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

4.4.12.1. Увеличение тома с чередованием

Чтобы увеличить размер логического тома, использующего чередование, необходимо иметь достаточно 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 — придется добавить как минимум два физических тома.

Возьмем в качестве примера группу **vg**, состоящую из двух физических томов.

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

Новый сегмент чередования может занимать все доступное в группе место.

```
# 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)
```

После этого в группе не останется 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 размер на 135 гигабайт.

```
# 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
```

На данном этапе увеличить объем тома с чередованием до максимального размера группы нельзя, так как для организации чередования необходимо 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
```

Надо добавить еще один физический том и уже после этого можно будет увеличить размер логического тома.

```
# 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 можно переопределить. Следующий пример расширяет существующий 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
```

4.4.12.2. Увеличение логического тома в режиме cling

При увеличении размера тома можно использовать механизм **cling** (параметр **--alloc cling** команды **lvextend**). При этом место будет выбираться в пределах тех же физических томов, где расположен последний сегмент увеличиваемого тома. Если места не хватает, но файл **lvm.conf** содержит список тегов, для увеличения тома будет выбран физический том с тем же тегом.

Например, при наличии логических томов с зеркалами, расположенными в той же группе, им можно присвоить теги @site1 и @site2, а в **lvm.conf** добавить следующее выражение:

```
cling_tag_list = [ "@site1", "@site2" ]
```

Приложение C, *Теги объектов LVM* содержит подробную информацию о присвоении тегов.

В следующем примере **lvm.conf** изменен так:

```
cling_tag_list = [ "@A", "@B" ]
```

В этом примере группа **taft** состоит из физических томов **/dev/sdb1**, **/dev/sdc1**, **/dev/sdd1**, **/dev/sde1**, **/dev/sdf1**, **/dev/sdg1**, **/dev/sdh1**, которые обозначены как **A**, **B** и **C**. Физические тома для организации составляющих зеркала будут выбираться по тегу.

```
[root@taft-03 ~]# pvs -a -o +pv_tags /dev/sd[bcdefgh]1
  PV         VG   Fmt  Attr PSize   PFree   PV Tags
  /dev/sdb1  taft lvm2 a-   135.66g 135.66g A
  /dev/sdc1  taft lvm2 a-   135.66g 135.66g B
  /dev/sdd1  taft lvm2 a-   135.66g 135.66g B
  /dev/sde1  taft lvm2 a-   135.66g 135.66g C
  /dev/sdf1  taft lvm2 a-   135.66g 135.66g C
  /dev/sdg1  taft lvm2 a-   135.66g 135.66g A
  /dev/sdh1  taft lvm2 a-   135.66g 135.66g A
```

Теперь в группе **taft** будет создан зеркальный том размером 100 гигабайт.

```
[root@taft-03 ~]# lvcreate -m 1 -n mirror --nosync -L 100G taft
```

Следующая команда покажет список устройств в составе зеркала.

```
[root@taft-03 ~]# lvs -a -o +devices
  LV                VG        Attr   LSize   Log         Copy%  Devices
  mirror            taft      Mwi-a- 100.00g mirror_mlog 100.00
mirror_mimage_0(0),mirror_mimage_1(0)
  [mirror_mimage_0] taft      iwi-ao 100.00g
/dev/sdb1(0)
  [mirror_mimage_1] taft      iwi-ao 100.00g
/dev/sdc1(0)
  [mirror_mlog]     taft      lwi-ao   4.00m
/dev/sdh1(0)
```

Ниже приведен пример увеличения размера зеркального тома в режиме **cling**, то есть для наращивания звеньев зеркала будут использоваться физические тома с тем же тегом.

```
[root@taft-03 ~]# lvextend --alloc cling -L +100G taft/mirror
  Extending 2 mirror images.
  Extending logical volume mirror to 200.00 GiB
  Logical volume mirror successfully resized
```

Следующая команда покажет результат добавления физических томов с тем же тегом, что и исходный том. Устройства с тегом **C** будут проигнорированы.

```
[root@taft-03 ~]# lvs -a -o +devices
  LV               VG        Attr   LSize   Log         Copy%  Devices
  mirror           taft      Mwi-a- 200.00g mirror_mlog  50.16
mirror_mimage_0(0),mirror_mimage_1(0)
  [mirror_mimage_0] taft      Iwi-ao 200.00g
/dev/sdb1(0)
  [mirror_mimage_0] taft      Iwi-ao 200.00g
/dev/sdg1(0)
  [mirror_mimage_1] taft      Iwi-ao 200.00g
/dev/sdc1(0)
  [mirror_mimage_1] taft      Iwi-ao 200.00g
/dev/sdd1(0)
  [mirror_mlog]     taft      lwi-ao   4.00m
/dev/sdh1(0)
```

4.4.13. Уменьшение размера логических томов

Чтобы уменьшить размер логического тома, сначала надо отключить файловую систему, а уже затем с помощью команды **lvreduce** сжать том. Завершив, снова подключите файловую систему.

ПРЕДУПРЕЖДЕНИЕ

Сначала надо уменьшить размер файловой системы, после чего можно уменьшить сам том. В противном случае данные могут быть потеряны.

Освобожденное пространство может быть выделено другим томам в группе.

Следующая команда уменьшит размер тома **lvol1** в группе **vg00** на три логических экстента.

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

4.5. ОПРЕДЕЛЕНИЕ УСТРОЙСТВ LVM С ПОМОЩЬЮ ФИЛЬТРОВ

При запуске системы выполняется команда **vgscan**, которая выполнит поиск меток LVM на блочных устройствах с целью определения того, какие из них представляют собой физические тома, а также получения метаданных и создания списков групп томов. Названия физических томов хранятся в файле **/etc/lvm/.cache** на каждом узле. Команды будут обращаться к этому файлу во избежание необходимости повторного сканирования.

С помощью фильтров в **lvm.conf** можно управлять тем, какие устройства будут проверяться. Фильтр содержит набор регулярных выражений, применяемых к устройствам в каталоге **/dev**.

Ниже приведены примеры фильтров. Следует отметить, что некоторые примеры не являются лучшими решениями, так как регулярные выражения свободно сопоставляются с полными путями. Например, **a/.*loop.*/** соответствует **a/loop/** и **/dev/solooperation/lvol1**.

Следующий фильтр добавит все обнаруженные устройства, что делается по умолчанию в случае, если фильтры не заданы.

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

Следующий фильтр исключит CD-ROM, если он не содержит диск, что позволит избежать задержек.

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

Фильтр для 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 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/.*/" ]
```

Приложение B, *Файлы конфигурации LVM* и справочная страница **lvm.conf**(5) содержат подробную информацию о **lvm.conf**.

4.6. ПЕРЕНОС ДАННЫХ В АКТИВНОЙ СИСТЕМЕ

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

pvmove разбивает данные на секции и создает временное зеркало для переноса каждой секции. Справочная страница **pvmove**(8) содержит подробную информацию.

Следующая команда переназначит все незанятое пространство физического тома **/dev/sdc1** другим томам в группе:

```
pvmove /dev/sdc1
```

Следующая команда перенесет экстенты логического тома **MyLV**.

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

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

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

Следующая команда будет сообщать о прогрессе (в процентах) с пятисекундным интервалом:

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

4.7. АКТИВАЦИЯ ЛОГИЧЕСКИХ ТОМОВ НА ОТДЕЛЬНЫХ УЗЛАХ КЛАСТЕРА

В кластерном окружении может понадобиться эксклюзивно активировать логические тома на одном узле.

Для этой цели служит команда **lvchange -aey**, а **lvchange -aly** активирует логические тома на локальном узле, но не эксклюзивно.

Активировать логические тома на конкретных узлах также можно с помощью тегов LVM (см. Приложение C, *Теги объектов LVM*) или в файле конфигурации (см. Приложение B, *Файлы конфигурации LVM*).

4.8. НАСТРОЙКА ОТЧЕТОВ LVM

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

Далее будет рассмотрено:

- Обзор аргументов, используемых для настройки формата генерируемого отчета.
- Перечень полей, которые можно выбрать для каждого объекта LVM.
- Обзор аргументов для сортировки генерируемого отчета.
- Определение единиц в отчете.

4.8.1. Изменение формата

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

- Аргумент **-o** позволяет выбрать поля для вывода. Например, стандартный вывод команды **pvs** 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 покажет только имя и размер физических томов.

```
# 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 знака "+" в комбинации с "-o".

  Пример добавления идентификатора UUID в стандартный отчет:

```
# 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** покажет 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.

  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
```

  Для выравнивания 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
```

Для добавления в отчет списка неисправных томов используется параметр **-P** команд **lvs** и **vgs** (см. Раздел 6.2, «Получение информации о неисправных устройствах»).

Справочные страницы **pvs**(8), **vgs**(8) и **lvs**(8) содержат полный перечень параметров.

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 покажет по одному физическому тому в строке:

```
# 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
```

4.8.2. Выбор объектов

Далее приведены таблицы с полями вывода команд **pvs**, **vgs**, **lvs**.

Префикс может быть опущен, если он соответствует стандартному имени, используемому командой. Например, при указании **name** с командой **pvs** 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
```

4.8.2.1. Команда pvs

Таблица 4.1, «Поля вывода pvs» содержит список аргументов команды **pvs** с названиями полей и их описанием.

Таблица 4.1. Поля вывода pvs

Аргумент	Столбец	Описание
dev_size	DevSize	Размер устройства в основе физического тома
pe_start	1st PE	Смещение 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	Столбец	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	Занятое пространство 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**, упорядоченные 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
```

Параметр **-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
```

pvs --segments покажет информацию о каждом сегменте физического тома. Сегмент представляет собой набор экстентов. Просмотр 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 обнаруженных устройств, которые не были инициализированы в виде физических томов 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
```

4.8.2.2. Команда vgs

Таблица 4.2, «Поля вывода vgs» содержит список аргументов **vgs** и названия полей.

Таблица 4.2. Поля вывода vgs

Аргумент	Столбец	Описание
lv_count	#LV	Число логических томов в группе

Аргумент	Столбец	Описание
max_lv	MaxLV	Максимально 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	Версия 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 показывает 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
```

vgs -v позволяет дополнительно показать 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
```

4.8.2.3. Команда lvs

Таблица 4.3, «Поля вывода lvs» содержит список аргументов команды **lvs** с названиями полей и их описанием.

Таблица 4.3. Поля вывода lvs

Аргумент	Столбец	Описание
chunksize **chunk_size**	Chunk	Размер сегментов снимка
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: доступ. Допустимые значения: (w)riteable, (r)ead-only Бит 3: политика выделения пространства. 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)
lv_kernel_major	KMaj	Действующий основной номер устройства логического тома (-1, если устройство неактивно)

Аргумент	Столбец	Описание
lv_kernel_minor	KMIN	Действующий вспомогательный номер устройства логического тома (-1, если неактивно)
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	Теги логического тома
lv_uuid	LV UUID	UUID логического тома
mirror_log	Log	Устройство, где размещен журнал зеркала
modules	Modules	Модуль ядра соответствий устройств
move_pv	Move	Исходный физический том, на основе которого создан временный логический том с помощью **pvmove**
origin	Origin	Исходное устройство 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)
snap_percent	Snap%	Процентная часть занятого пространства снимка
stripes	#Str	Число сегментов чередования или зеркал логического тома

Аргумент	Столбец	Описание
stripesize **stripe_size**	Stripe	Размер сегментов чередования

По умолчанию вывод **lvs** будет содержать 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
```

Параметр **-v** команды **lvs** добавляет в таблицу 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
```

lvs --segments показывает стандартный набор столбцов с информацией о сегментах. При этом префикс **seg** указывать не обязательно. Так, **lvs --segments** по умолчанию покажет 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 будет содержать соответствующую информацию.

```
# 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
```

Параметр **-v** добавляет 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 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
```

4.8.3. Форматирование вывода

Обычно вывод команд **lvs**, **vgs**, **pvs** сохраняется и отдельно форматируется. Аргумент **--unbuffered** покажет исходный вывод сразу после его генерации.

С помощью **-O** можно указать порядок столбцов. При этом включение этих столбцов в вывод необязательно.

Пример вывода **pvs**, содержащего имена физических томов, их размер и свободное пространство:

```
# 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
```

В следующем примере вывод отсортирован по размеру свободного пространства; при этом поле pv_free не будет показано.

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

Чтобы изменить порядок сортировки, 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
```

4.8.4. Выбор единиц

Аргумент **--units** позволяет указать единицы объема данных. Допустимые значения: (b)ytes, (k)ilobytes, (m)egabytes, (g)igabytes, (t)erabytes, (e)xabytes, (p)etabytes и (h)uman-readable. По умолчанию используется 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, указав единицы 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
```

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), размер которых равен 512 байт, или даже произвольные единицы.

Пример, 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, где в качестве единиц используются блоки размером 4 мегабайт:

```
# 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

В этой главе приведены простые примеры конфигурации LVM.

5.1. СОЗДАНИЕ ЛОГИЧЕСКОГО ТОМА НА ТРЕХ ДИСКАХ

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

5.1.1. Создание физических томов

Для использования дисков необходимо обозначить их как физические тома.

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

Команда создания группы томов **new_vol_group**:

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

С помощью **vgs** можно просмотреть атрибуты новой группы:

```
[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 выделен логический том **new_logical_volume** размером 2 гигабайта 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

Теперь для нового тома можно создать файловую систему.

```
[root@tng3-1 ~]# mkfs.gfs2 -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 подключат логический том в файловую систему и покажут 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

Сначала надо обозначить диски 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

Затем можно создать группу **volgroup01**:

```
[root@tng3-1 ~]# vgcreate volgroup01 /dev/sda1 /dev/sdb1 /dev/sdc1
  Volume group "volgroup01" successfully created
```

С помощью **vgs** можно просмотреть атрибуты новой группы:

```
[root@tng3-1 ~]# vgs
  VG                #PV #LV #SN Attr   VSize  VFree
  volgroup01          3   0   0 wz--n- 51.45G 51.45G
```

5.2.3. Создание логического тома

Далее в группе **volgroup01** будет создан том **striped_logical_volume** общим размером 2 гигабайта с сегментами, размер которых равен 4 килобайта.

```
[root@tng3-1 ~]# lvcreate -i3 -I4 -L2G -nstriped_logical_volume volgroup01
  Rounding size (512 extents) up to stripe boundary size (513 extents)
  Logical volume "striped_logical_volume" created
```

5.2.4. Создание файловой системы

Теперь для нового тома можно создать файловую систему.

```
[root@tng3-1 ~]# mkfs.gfs2 -plock_nolock -j 1
/dev/volgroup01/striped_logical_volume
This will destroy any data on /dev/volgroup01/striped_logical_volume.

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

Device:                    /dev/volgroup01/striped_logical_volume
Blocksize:                 4096
Filesystem Size:           492484
Journals:                  1
Resource Groups:           8
Locking Protocol:          lock_nolock
Lock Table:

Syncing...
All Done
```

Следующие команды подключат логический том в файловую систему и покажут информацию об использовании дискового пространства.

```
[root@tng3-1 ~]# mount /dev/volgroup01/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/volgroup01/striped_logical_volume
                        1969936        20   1969916   1% /mnt
```

5.3. РАЗБИЕНИЕ ГРУППЫ ТОМОВ

Если в группе томов достаточно незанятого пространства, то новая группа может быть добавлена без необходимости добавления новых дисков. В качестве примера рассмотрим группу, в состав которой входит 3 физических тома.

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

После разбиения группа **myvg** будет состоять из **/dev/sda1** и **/dev/sdb1**, а **/dev/sdc1** будет включен в состав 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

Для переноса занятых физических экстентов с **/dev/sdc1** на **/dev/sdb1** используется команда **pvmove**. Процесс может занять достаточно много 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 убедитесь, 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 **vgsplit** разделим группу **myvg** и создадим новую группу **yourvg**.

Прежде чем приступить к разделению группы томов, сначала необходимо убедиться, что файловая система отключена, а затем отключить сам логический том.

Отключить том можно с помощью **lvchange** и **vgchange**. Приведенная команда отключает том **mylv**, затем выделяет пространство для группы **yourvg** из **myvg** и переносит физический том **/dev/sdc1** в **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"
```

Атрибуты обеих групп можно просмотреть 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

После создания группы можно создать логический том **yourlv**:

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

5.3.5. Создание файловой системы и подключение нового логического тома

Теперь для нового логического тома можно создать файловую систему.

```
[root@tng3-1 ~]#  mkfs.gfs2 -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 подключение исходного логического тома

Поскольку том **mylv** был отключен, его нужно снова подключить.

```
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 удаление диска из логического тома либо с целью его замены, либо при его переносе в другой том. Чтобы удалить диск, сначала необходимо переместить экстенты на другой диск или набор дисков.

5.4.1. Перенос экстентов в другой физический 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** и затем его удалить из группы томов.

Если на других физических томах в группе достаточно свободных экстентов, на удаляемом устройстве можно выполнить **pvmove** без аргументов, чтобы перераспределить 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 **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 отсоединить 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, надо переместить экстенты **/dev/sdb1** на новое устройство **/dev/sdd1**.

5.4.2.1. Создание физического тома

На основе **/dev/sdd1** создадим физический том:

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

5.4.2.2. Добавление физического тома в группу

Добавим **/dev/sdd1** в 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 **pvmove** переместим данные с **/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

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

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

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

5.5. СОЗДАНИЕ ЗЕРКАЛЬНОГО ЛОГИЧЕСКОГО ТОМА В КЛАСТЕРЕ

Создание зеркального логического тома в кластере осуществляется по тем же правилам, что и на отдельном компьютере. Отличия состоят в том, что в кластере должны работать необходимые кластерные службы, кластер должен обладать кворумом, а тип блокирования в **lvm.conf** должен разрешать кластерное блокирование. Раздел 5.5, «Создание зеркального логического тома в кластере» содержит примеры создания зеркального тома в кластере.

Далее будет создан зеркальный логический том. Предварительно будет проверено, выполняются ли необходимые кластерные службы.

1. Чтобы создать общий зеркальный логический том, который будет доступен всем узлам в кластере, на каждом узле необходимо определить нужный тип блокирования в файле **lvm.conf**.

   ```
   # /sbin/lvmconf --enable-cluster
   ```

2. Следующая команда проверит факт выполнения службы **clvmd** на текущем узле:

   ```
   [root@doc-07 ~]# ps auxw | grep clvmd
   root     17642  0.0  0.1 32164 1072 ?        Ssl  Apr06   0:00
   clvmd -T20 -t 90
   ```

   Далее будут показаны сведения о состоянии кластера:

   ```
   [root@example-01 ~]# cman_tool services
   fence domain
   member count  3
   victim count  0
   victim now    0
   master nodeid 2
   wait state    none
   members       1 2 3

   dlm lockspaces
   name          clvmd
   id            0x4104eefa
   ```

```
flags         0x00000000
change        member 3 joined 1 remove 0 failed 0 seq 1,1
members       1 2 3
```

3. Убедитесь, что пакет **cmirror** установлен.

4. Запустите **cmirrord**.

```
[root@hexample-01 ~]# service cmirrord start
Starting cmirrord:                                         [  OK
]
```

5. Далее будет создано три физических тома. Два будут использоваться для организации зеркала, а на третьем будет храниться журнал зеркала.

```
[root@doc-07 ~]# pvcreate /dev/xvdb1
  Physical volume "/dev/xvdb1" successfully created
[root@doc-07 ~]# pvcreate /dev/xvdb2
  Physical volume "/dev/xvdb2" successfully created
[root@doc-07 ~]# pvcreate /dev/xvdc1
  Physical volume "/dev/xvdc1" successfully created
```

6. Все три тома будут добавлены в группу **vg001**.

```
[root@doc-07 ~]# vgcreate vg001 /dev/xvdb1 /dev/xvdb2 /dev/xvdc1
  Clustered volume group "vg001" successfully created
```

Вывод **vgcreate** демонстрирует, что группа томов является кластерной (атрибут «c»). Команда **vgs** также может это проверить.

```
[root@doc-07 ~]# vgs vg001
  VG       #PV #LV #SN Attr   VSize  VFree
  vg001      3   0   0 wz--nc 68.97G 68.97G
```

7. В следующем примере будет создан логический том **mirrorlv** с одним зеркалом из группы **vg001**. Выделяемые экстенты будут указаны явно.

```
[root@doc-07 ~]# lvcreate -l 1000 -m1 vg001 -n mirrorlv
/dev/xvdb1:1-1000 /dev/xvdb2:1-1000 /dev/xvdc1:0
  Logical volume "mirrorlv" created
```

С помощью **lvs** можно наблюдать за прогрессом создания зеркала (столбец Copy%).

```
[root@doc-07 log]# lvs vg001/mirrorlv
  LV       VG       Attr   LSize Origin Snap%  Move Log
Copy%  Convert
  mirrorlv vg001    mwi-a- 3.91G                    vg001_mlog
47.00
[root@doc-07 log]# lvs vg001/mirrorlv
  LV       VG       Attr   LSize Origin Snap%  Move Log
Copy%  Convert
  mirrorlv vg001    mwi-a- 3.91G                    vg001_mlog
91.00
```

```
[root@doc-07 ~]#  lvs vg001/mirrorlv
  LV       VG       Attr   LSize Origin Snap%  Move Log
Copy%  Convert
  mirrorlv vg001    mwi-a- 3.91G                    vg001_mlog
100.00
```

Сообщение об успешном создании зеркала будет зарегистрировано в системном журнале:

```
May 10 14:52:52 doc-07 [19402]: Monitoring mirror device vg001-
mirrorlv for events
May 10 14:55:00 doc-07 lvm[19402]: vg001-mirrorlv is now in-sync
```

8. Для просмотра структуры зеркала можно выполнить **lvs** с параметрами **-o +devices**. В приведенном ниже примере логический том состоит из двух линейных томов и одного журнала.

```
[root@doc-07 ~]# lvs -a -o +devices
  LV                 VG         Attr   LSize  Origin Snap%  Move
Log           Copy%  Convert Devices
  mirrorlv           vg001      mwi-a-  3.91G
mirrorlv_mlog 100.00
mirrorlv_mimage_0(0),mirrorlv_mimage_1(0)
  [mirrorlv_mimage_0] vg001      iwi-ao  3.91G
/dev/xvdb1(1)
  [mirrorlv_mimage_1] vg001      iwi-ao  3.91G
/dev/xvdb2(1)
  [mirrorlv_mlog]     vg001      lwi-ao  4.00M
/dev/xvdc1(0)
```

Параметр **seg_pe_ranges** покажет диапазон физических экстентов.

```
[root@doc-07 ~]# lvs -a -o +seg_pe_ranges --segments
  PE Ranges
  mirrorlv_mimage_0:0-999 mirrorlv_mimage_1:0-999
  /dev/xvdb1:1-1000
  /dev/xvdb2:1-1000
  /dev/xvdc1:0-0
```

ПРИМЕЧАНИЕ

Раздел 6.3, «Восстановление после сбоя зеркала» содержит инструкции по восстановлению зеркала в случае выхода из строя его составляющих.

ГЛАВА 6. ДИАГНОСТИКА ПРОБЛЕМ

Этот раздел содержит инструкции по исправлению различных проблем с LVM.

6.1. ДИАГНОСТИКА

Если команда не функционирует как ожидается, осуществить сбор информации можно перечисленными ниже способами.

- Использовать параметры **-v**, **-vv**, **-vvv**, **-vvvv** для усиления степени детализации вывода исполняемой команды.
- Если проблема связана с активацией логического тома, в секции "log" файла конфигурации укажите 'activation = 1', затем выполните команду с параметром **-vvvv**. Проверив вывод, не забудьте сбросить значение переменной в 0, чтобы избежать возможных проблем с блокировкой машины в случае нехватки памяти.
- 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 конфигурацию.
- Файл **.cache** в **/etc/lvm** должен включать список устройств, на которых расположены физические тома.

6.2. ПОЛУЧЕНИЕ ИНФОРМАЦИИ О НЕИСПРАВНЫХ УСТРОЙСТВАХ

Аргумент **-P** команд **lvs** и **vgs** покажет информацию о сбойном томе. К примеру, если в составе группы **vg** перестало функционировать одно устройство, следующая команда поможет его определить:

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

vgs -P покажет информацию о сбойном устройстве, даже если группа томов, в состав которой это устройство входит, недоступна.

```
[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 выход из строя одного устройства привел к сбою линейного тома и тома с чередованием в группе. **lvs** без аргумента **-P** покажет:

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

-P добавит в вывод список неисправных 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** с аргументом **-P**.

```
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

В этой секции рассматривается пример восстановления после выхода из строя компонента зеркального логического тома как следствие сбоя физического тома в его основе. При этом параметр **mirror_log_fault_policy** имеет значение **remove** (см. Раздел 6.3, «Восстановление после сбоя зеркала»).

LVM преобразует зеркальный том в линейный, сохранив функциональность, но уже без избыточности. После этого можно добавить новое устройство и пересоздать зеркало.

Следующая команда создаст физические тома, на основе которых впоследствии будет создано зеркало.

```
[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
```

Далее создадим **vg** и зеркальный том **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 **lvs** можно проверить структуру зеркального тома, устройств в его основе и журнала зеркала. В первом примере (см. ниже) синхронизация зеркала еще не завершена, поэтому сначала надо дождаться, пока индикатор прогресса **Copy%** не достигнет 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 звена зеркала **/dev/sda1**. Попытки записи в зеркальный том приводят к определению неисправного зеркала, в случае чего LVM преобразует его 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**. Как следствие наличия неисправного диска, вывод будет сообщать об ошибке ввода-вывода.

```
[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)
```

Том можно продолжать использовать, но уже как линейный.

После подключения нового устройства можно пересоздать физический том, а затем сам зеркальный том. При попытке использования того же диска в процессе выполнения **pvcreate** появятся предупреждающие сообщения.

```
[root@link-08 ~]# pvcreate /dev/sdi[12]
  Physical volume "/dev/sdi1" successfully created
  Physical volume "/dev/sdi2" 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/sdi1           lvm2 [603.94 GB]
  PV /dev/sdi2           lvm2 [603.94 GB]
  Total: 16 [2.11 TB] / in use: 14 [949.65 GB] / in no VG: 2 [1.18 TB]
```

Далее надо добавить созданный том в группу.

```
[root@link-08 ~]# vgextend vg /dev/sdi[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/sdi1   VG vg   lvm2 [603.93 GB / 603.93 GB free]
  PV /dev/sdi2   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   ]
```

Теперь можно преобразовать линейный том в зеркальный.

```
[root@link-08 ~]# lvconvert -m 1 /dev/vg/groupfs /dev/sdi1 /dev/sdb1
/dev/sdc1
  Logical volume mirror converted.
```

Убедитесь, что зеркало было восстановлено.

```
[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/sdi1(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 повреждена. В этом случае появится сообщение об ошибке в области метаданных или о неудачной попытке нахождения физического тома с заданным UUID. Чтобы не потерять данные физического тома, можно попробовать заново переписать область метаданных, включив прежний UUID метаданных.

ПРЕДУПРЕЖДЕНИЕ

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

Пример вывода в случае потери или повреждения 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.
  ...
```

Перезаписанный UUID можно найти в файле ***ГруппаТомов_xxxx*.vg** в каталоге **/etc/lvm/archive**.

Другой способ определения идентификатора UUID состоит в отключении тома и установке параметра **-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 аргументов **--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**. Команда восстановит информацию о метаданных из последнего успешного архива метаданных **VG_00050.vg**. **pvcreate** перезапишет только область метаданных, что не окажет влияния 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
```

Затем с помощью **vgcfgrestore** можно восстановить метаданные.

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

После этого можно получить доступ логическим томам.

```
[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)
```

Первая команда выполнит активацию тома, а вторая покажет список активных томов.

```
[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)
```

vgcfgrestore может восстановить физический том, если метаданные были полностью перезаписаны. Если область перезаписи вышла за пределы области метаданных, возможна потеря данных. Попробуйте их восстановить с помощью **fsck**.

6.5. ЗАМЕНА ФИЗИЧЕСКОГО ТОМА

Для замены физического тома можно придерживаться той же последовательности действий, что и при восстановлении метаданных (см. Раздел 6.4, «Восстановление метаданных физического тома»). Используйте аргументы **--partial** и **--verbose** команды **vgdisplay** для определения UUID и размера уже исключенных физических томов. Если же вашей целью является замена физического тома другим томом того же размера, для инициализации нового устройства с тем же UUID используйте аргументы **--restorefile** и **--uuid** команды **pvcreate**. Затем с помощью **vgcfgrestore** можно восстановить метаданные группы.

6.6. УДАЛЕНИЕ ФИЗИЧЕСКИХ ТОМОВ ИЗ ГРУППЫ

В случае выхода из строя одного физического тома можно активировать остальные тома в группе с помощью аргумента **--partial** команды **vgchange**. Аргумент **--removemissing** команды **vgreduce** позволяет удалить из группы все логические тома, использующие этот физический том.

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

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**, логические тома будут удалены. В этом случае можно заменить физический том и выполнить **vgcfgrestore** для восстановления группы.

6.7. НЕХВАТКА СВОБОДНЫХ ЭКСТЕНТОВ ДЛЯ ЛОГИЧЕСКОГО ТОМА

Если для создания логического тома не хватает экстентов, хотя вывод **vgdisplay** и **vgs** показывает обратное, появится сообщение "Insufficient free extents". Дело в том, что обе команды округляют значения для облегчения чтения. Чтобы указать точный размер, используйте число свободных физических экстентов вместо байтов.

Вывод **vgdisplay** по умолчанию включает число свободных физических экстентов:

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

Или же можно передать параметры **vg_free_count** и **vg_extent_count** команде **vgs** для вывода числа свободных экстентов и их общего числа.

```
[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
```

Всего свободно 8780 экстентов. Теперь можно выполнить следующую команду ("l" использует экстенты в качестве единиц вместо байтов):

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

Том займет 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
```

Можно нарастить логический том так, чтобы он использовал процент незанятого пространства в группе. Для этого используется аргумент **`-l`** команды **`lvcreate`** (см. Раздел 4.4.1, «Создание линейных логических томов»).

ГЛАВА 7. АДМИНИСТРИРОВАНИЕ LVM В ГРАФИЧЕСКОМ РЕЖИМЕ

Помимо текстового интерфейса LVM предоставляет графический интерфейс (GUI, Graphical User Interface) для настройки логических томов LVM. Для запуска графической утилиты выполните **`system-config-lvm`**. В главе «LVM» в *руководстве по развертыванию Red Hat Enterprise Linux* приведены пошаговые инструкции по настройке логического тома.

ПРИЛОЖЕНИЕ A. DEVICE MAPPER

Device Mapper — драйвер ядра, реализующий основную инфраструктуру для управления томами. Он позволяет создавать проекции устройств, которые могут использоваться в качестве логических томов. Device Mapper не обладает информацией о группах томов и форматах метаданных.

Device Mapper служит основой для некоторых технологий высокого уровня и активно используется командами **dmraid** и **dm-multipath**.

Device Mapper используется для активации логических томов. Логический том преобразуется в проецируемое устройство, а каждому сегменту будет соответствовать строка с таблице соответствий. Device Mapper поддерживает прямое проецирование, проецирование с чередованием или проецирование с учетом ошибок. Так, два диска могут быть объединены в один логический том с двумя линейными соответствиями — по одному на диск. При создании тома LVM будет также создана проекция, для обращения к которой используется **dmsetup**. Раздел A.1, «Таблица соответствий» содержит описание форматов устройств в таблице соответствий, а Раздел A.2, «dmsetup» предоставляет информацию об **dmsetup**.

A.1. ТАБЛИЦА СООТВЕТСТВИЙ

Проекции устройств определяются в таблице, которая содержит соответствия для диапазонов логических секторов. Формат строк в таблице:

```
начало длина проекция [параметры...]
```

Параметр ***начало*** обычно равен 0. Сумма параметров ***начало*** и ***длина*** в одной строке должна быть равна величине ***начало*** в следующей строке. Параметры зависят от типа ***проекции***.

Размеры указываются в секторах (размер сектора — 512 байт).

Если в параметрах проецирования указано устройство, в файловой системе к нему можно обращаться и по имени (например, **/dev/hda**), и 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 обозначают начальный блок и длину сегмента. Следующий параметр определяет тип проецирования (**linear**).

Тип может принимать следующие значения:

- linear;
- striped;
- mirror;
- snapshot и snapshot-origin;

- error;
- zero;
- multipath;
- crypt.

A.1.1. Тип linear

Создает линейное соответствие между непрерывным диапазоном блоков и другим блочным устройством.

```
начало длина linear устройство смещение
```

начало

первый блок виртуального устройства

длина

длина сегмента

устройство

блочное устройство, которое можно указать по имени или по номеру в формате ***основной*:*вспомогательный***

смещение

смещение проекции

В следующем примере будет создана линейная проекция общей длиной 1638400. При этом основной и вспомогательный номера — 8:2, а смещение устройства — 41146992.

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

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 чередующихся сегментов, за которым следуют пары имен устройств и секторов. Формат:

```
начало длина striped #число размер_сегмента устройство1 смещение1 ...
устройствоN смещениеN
```

Каждому сегменту чередования соответствует одна пара значений ***устройство*** и ***смещение***.

начало

первый блок виртуального устройства

длина

длина сегмента

число

число чередующихся сегментов для организации виртуального устройства

размер_сегмента

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

устройство

блочное устройство, заданное с помощью имени или пары ***основной***:***вспомогательный***.

смещение

смещение проекции

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

первый блок виртуального устройства

73728

длина сегмента

striped 3 128

чередование на трех устройствах с размером сегмента равным 128 блокам

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 организовано чередование на двух устройствах; при этом размер сегмента будет равен 256 КиБ, а устройства будут заданы с помощью имен.

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

A.1.3. Тип mirror

Тип mirror поддерживает проецирование зеркальных логических устройств. Формат:

```
начало длина mirror тип_журнала #число_аргументов аргумент1 ... аргументN
#число_устройств устройство1 смещение1 ... устройствоN смещениеN
```

начало

первый блок виртуального устройства

длина

длина сегмента

тип_журнала

Возможные значения:

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 зеркала хранится на локальном 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, где регистрируется информация о синхронизации регионов с зеркалом. *Размер* определяет размер регионов.

В кластерном окружении *UUID* определяет уникальный идентификатор устройства, где расположен журнал.

Аргумент **[no]sync** не является обязательным; его целью является определение состояния зеркала как "in-sync" или "out-of-sync". Аргумент **block_on_error** заставит зеркало отвечать на ошибки, а не игнорировать их.

#число_аргументов

общее число аргументов журнала

аргумент1...аргументN

аргументы журнала зеркала. Число аргументов определяется параметром ***#число_аргументов***, а допустимые аргументы определяются параметром ***тип_журнала***.

#число_устройств

число составляющих зеркала. Для каждого элемента задается устройство и смещение.

устройство

блочное устройство в основе звена зеркала. Задается с помощью имени устройства или пары номеров ***основной***:***вспомогательный***.

смещение

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

Следующий пример демонстрирует проецирование для кластерного зеркала с поддержкой журнала 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

первый блок виртуального устройства

52428800

длина сегмента

mirror clustered_disk

зеркало и тип журнала, который в данном случае подразумевает, что он хранится на диске.

4

далее следуют четыре аргумента

253:2

основной и вспомогательный номера

1024

размер региона, где будет регистрироваться информация о синхронизации

UUID

UUID устройства с журналами 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

A.1.4. Тип snapshot и snapshot-origin

При создании первого снимка тома используются четыре устройства Device Mapper:

1. Устройство с линейным проецированием (**linear**), содержащее изначальную таблицу соответствий для исходного тома.
2. Устройство с линейным проецированием, используемое в качестве устройства CoW (copy-on-write) для исходного тома. При каждой операции записи данные сохраняются и на COW-устройство, тем самым синхронизируя содержимое (до тех пор пока COW-устройство не заполнится).
3. Устройство с проецированием **snapshot**, совмещающим типы 1 и 2, которое представляет собой видимый снимок тома.
4. Исходный том. Таблица этого тома заменяется соответствием "snapshot-origin" из первого устройства.

При создании этих устройств используется фиксированная схема присвоения имен. Например, команды создания тома **base** и его снимка **snap** будут выглядеть так:

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

В результате будет получено 4 устройства:

```
# 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**:

```
начало длина snapshot-origin исходный
```

начало

первый блок виртуального устройства

длина

длина сегмента

исходный

исходный том, для которого создается снимок

Исходный том **snapshot-origin** обычно имеет один или несколько снимков. При чтении проецирование будет осуществляться напрямую. При записи исходные данные будут сохранены в COW-устройство каждого снимка. Это делается с целью буферизации изменений, до тех пор пока COW-устройство не будет заполнено.

Формат **snapshot**:

```
начало длина snapshot исходный COW-устройство P|N размер_секции
```

начало

первый блок виртуального устройства

длина

длина сегмента

исходный

исходный том, для которого создается снимок

COW-устройство

Устройство, где будут сохраняться измененные секции данных

P|N

P (Persistent) или N (Not persistent) определяют, будет ли сохраняться снимок после перезагрузки. Значение P сохраняет снимок, а N — не сохраняет, в случае чего метаданные будут храниться в памяти.

размер_секции

Размер изменяемых секций на COW-устройстве (в секторах).

Следующий пример демонстрирует **snapshot-origin** для устройства с номером 254:11.

```
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» завершает неудачей любые запросы ввода и вывода к заданному сектору.

«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 работающий компонент зеркала специально заменяется поврежденным.

Проекция «error» может использоваться вместо поврежденного устройства во избежание задержек обращения или при реорганизации метаданных LVM в случае сбоя.

error требует указания только двух параметров — *начало* и *конец*.

Пример:

```
0 65536 error
```

A.1.6. Тип zero

Для подстановки соответствия **zero** используется блочное устройство **/dev/zero**. Запросы чтения будут возвращать блоки нулей, а запись хоть и будет завершена успешно, но записываемые данные не будут сохранены. **zero** принимает два параметра — *начало* и *длина*.

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

```
0 65536 zero
```

A.1.7. Тип multipath

Тип «multipath» обеспечивает сопоставление для многопутевых устройств. Формат:

```
начало длина  multipath  #число_функций [функция1 ... функцияN]
#число_аргументов [аргумент1 ... аргументN] #число_групп след_группа
аргументы_группы1 ... аргументы_группыN
```

Каждой группе путей соответствует один набор ***аргументы_группы***.

начало

первый блок виртуального устройства

длина

длина сегмента

#число_функций

Число функций с их последующим списком. Если равно нулю, далее сразу будет следовать ***#число_аргументов***. В настоящее время поддерживается лишь одна функция — **queue_if_no_path**, которая ставит запросы ввода-вывода в очередь при отсутствии доступных путей.

К примеру, если параметр **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 путей 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
```

#число_аргументов

Число аргументов аппаратного обработчика с последующим списком аргументов. Обработчик определяет модуль для выполнения операций, связанных с оборудованием, при изменении групп путей или при обработке ошибок ввода-вывода. Если значение равно нулю, дальше сразу будет следовать ***число_групп***.

#число_групп

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

след_группа

Следующая группа маршрутов.

аргументы_группы

Каждой группе соответствуют следующие аргументы:

```
алгоритм #число_аргументов #число_маршрутов #число_аргументов
устройство1 число_запросов1 ... устройствоN число_запросовN
```

Каждому маршруту соответствует один набор аргументов.

алгоритм

Алгоритм выбора пути для обработки следующего запроса ввода-вывода.

#число_аргументов

Число аргументов. По умолчанию равно 0.

#число_маршрутов

Число маршрутов в группе.

#число_аргументов

Число аргументов для каждого маршрута. По умолчанию равно 1 (единственный аргумент ***ioreqs***).

устройство

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

ioreqs

Число запросов ввода и вывода, которые будут переданы этому маршруту, прежде чем будет выбран следующий путь.

Рисунок A.1, «Соответствие multipath» демонстрирует проекцию с двумя группами маршрутов.

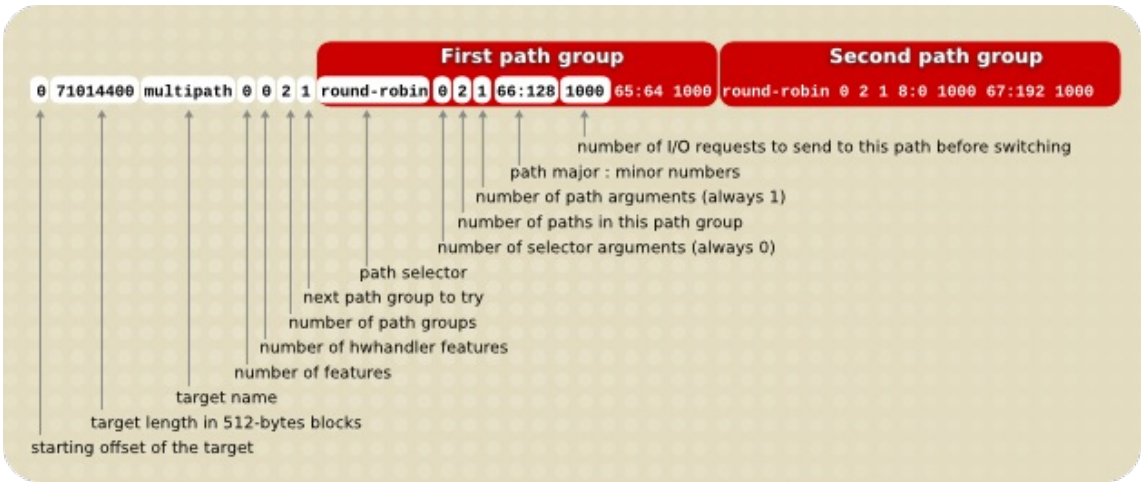


Рисунок A.1. Соответствие multipath

Следующий пример демонстрирует определение резервного устройства на случай сбоя многопутевого устройства. Цель включает четыре группы путей, но только один путь в группе может быть открыт.

```
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
```

Ниже приведен пример того же устройства, но в этом случае используется лишь одна группа маршрутов, между которыми нагрузка 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 *DM-Multipath* содержит дальнейшую информацию.

A.1.8. Тип crypt

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

Формат:

```
начало длина crypt код ключ смещение_IV устройство смещение
```

начало

первый блок виртуального устройства

длина

длина сегмента

код

Формат кода: ***cipher[-режим]-ivmode[:iv параметры]***.

код

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

режим

Надо указать **cbc**, а не **ebc**, так как **ebc** не использует исходный вектор (IV).

ivmode[:iv параметры]

IV — вектор инициализации, позволяющий изменить метод шифрования. Режим IV может принимать значения **plain** или **essiv:hash**. Метод **plain** использует номер сектора (плюс смещение IV) в качестве IV, в то время как **-essiv** представляет собой расширение для усиления защиты от атак Watermarking.

ключ

Ключ шифрования в шестнадцатеричной форме

смещение_IV

Смещение вектора инициализации

устройство

блочное устройство, которое можно указать по имени или по номеру в формате ***основной*:*вспомогательный***

смещение

смещение проекции

Пример:

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

A.2. DMSETUP

Для работы с модулем Device Mapper используется команда **dmsetup** с параметрами **info**, **ls**, **status** и **deps**. Они будут рассмотрены ниже.

За информацией о параметрах обратитесь к справочной странице **dmsetup**(8).

A.2.1. dmsetup info

Команда **dmsetup info *устройство*** позволяет получить информацию об устройствах Device Mapper. Если устройство не указано, будут показаны сведения о всех настроенных устройствах Device Mapper.

Формат вывода **dmsetup info**:

Name

Имя устройства в формате «группа_томов-логический_том». Если исходное имя содержит дефис, он будет указан дважды.

State

Возможные значения статуса: **SUSPENDED**, **ACTIVE**, **READ-ONLY**. Например для перевода устройства в состояние **SUSPENDED** выполните **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 файла. Значение можно изменить с помощью параметра **--readahead** команды **dmsetup**.

Tables present

Допустимые значения: **LIVE**, **INACTIVE**. **INACTIVE** обозначает, что загружена неактивная таблица, а когда команда **dmsetup resume** восстановит работу устройства (переведет в **ACTIVE**), ее статус изменится на **LIVE**. Более подробную информацию можно найти на справочной странице **dmsetup**.

Open count

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

Event number

Число полученных событий. **dmsetup wait *n*** позволит дождаться события с номером n, заблокировав вызов до тех пор, пока оно не будет получено.

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, включающее начало и конец диска, будет иметь 2 фрагмента.

UUID

Идентификатор UUID.

Пример частичного вывода **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. dmsetup ls

С помощью команды **dmsetup ls** можно просмотреть список проекций устройств. **dmsetup ls --target *тип*** покажет устройства, в основе которых лежит по крайней мере один компонент указанного типа. Другие параметры **dmsetup ls** можно просмотреть на справочной странице **dmsetup**.

Получение списка 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)
```

Получение списка зеркальных устройств:

```
[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)
```

Конфигурация LVM может быть довольно сложной. Параметр **--tree** покажет связи устройств в виде дерева.

```
# dmsetup ls --tree
```

```
vgtest-lvmir (253:13)
 ├─vgtest-lvmir_mimage_1 (253:12)
 │  └─mpathep1 (253:8)
 │     └─mpathe (253:5)
 │        ├─ (8:112)
 │        └─ (8:64)
 ├─vgtest-lvmir_mimage_0 (253:11)
 │  └─mpathcp1 (253:3)
 │     └─mpathc (253:2)
 │        ├─ (8:32)
 │        └─ (8:16)
 └─vgtest-lvmir_mlog (253:4)
    └─mpathfp1 (253:10)
       └─mpathf (253:6)
          ├─ (8:128)
          └─ (8:80)
```

A.2.3. dmsetup status

dmsetup status *устройство* покажет статус всех составляющих заданного устройства. Если устройство не указано, будет приведена информация для всех настроенных устройств. **dmsetup status --target *тип*** покажет статус устройств, в основе которых лежит по крайней мере один компонент указанного типа.

Получение статуса 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. dmsetup deps

Команда **dmsetup deps *устройство*** покажет список основных и вспомогательных номеров устройств, входящих в таблицу соответствий для заданного устройства. Если устройство не указано, будет показана информация обо всех устройствах.

Просмотр 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)
```

Получение 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)
```

A.3. ПОДДЕРЖКА UDEV

Основной задачей менеджера устройств **udev** является обслуживание файлов устройств в каталоге **/dev**. Создание файлов устройств осуществляется в соответствии с правилами **udev** в пространстве пользователя. При добавлении, удалении и изменении устройств ядро генерирует события, которые будут обработаны правилами **udev**.

udev может создавать не только файлы устройств, но и символьные ссылки, предоставляя тем самым свободу действий в выборе имен устройств и организации каталогов внутри **/dev**.

Событие **udev** содержит информацию об устройстве — имя, подсистема, которой оно принадлежит, тип устройства, основной и второстепенный номер и тип события. На основе этой информации и данных из **/sys** можно создать фильтры для правил.

udev предоставляет централизованный механизм настройки доступа к файлам устройств. Пользователь может определить собственный набор правил доступа к устройствам.

Правила могут содержать вызовы других программ. В свою очередь, программы могут экспортировать переменные окружения. Полученные результаты могут быть переданы правилам для дальнейшей обработки.

Обработка событий может осуществляться не только службой **udev**, но и другими программами, использующими библиотеку **udev**.

A.3.1. Интеграция udev с Device Mapper

В RHEL 6 обработка событий **udev** синхронизируется с Device Mapper. Синхронизация необходима, так как **udev** применяет правила параллельно с программой, запрашивающей изменение устройства (например, **dmsetup** или LVM). В противном случае могло бы оказаться так, что пользователь пытается удалить устройство, которое еще используется службой **udev**.

RHEL 6 обеспечивает поддержку правил **udev** для устройств Device Mapper. Правила располагаются в **/lib/udev/rules.d** (см. Таблица A.1, «Правила udev»).

Таблица A.1. Правила udev

Файл	Описание
10-dm.rules	Содержит общие правила Device Mapper и создает символьные ссылки в **/dev/mapper** для устройств **/dev/dm-*N***, где N — динамически назначенный ядром номер устройства. Сценарии должны использовать действительные имена устройств из **/dev/mapper**, а не **/dev/dm-*N***, так как номер N назначается динамически и изменяет порядок инициализации устройств.

Файл	Описание
11-dm-lvm.rules	Содержит правила для устройств LVM и создает символьные ссылки в каталоге **/dev/*vgname*** для устройств **/dev/dm-*N***. Правила udev должны следовать формату **11-dm-*подсистема*.rules**. Пользователи **libdevmapper**, предоставляющие правила для **udev**, также должны придерживаться этого формата.
13-dm-disk.rules	Содержит правила для обработки устройств Device Mapper и создает символьные ссылки в каталогах **/dev/disk/by-id**, **/dev/disk/by-uuid** и **/dev/disk/by-uuid**.
95-dm-notify.rules	Содержит правило для уведомления ожидающего процесса. Уведомление будет отправлено после завершения обработки **udev**, то есть после применения всех правил. После этого процесс продолжит работу.

Дополнительные правила доступа можно добавить в **12-dm-permissions.rules**. Этот файл не устанавливается в **/lib/udev/rules** по умолчанию и изначально расположен в каталоге **/usr/share/doc/device-mapper-*версия***. Он содержит шаблон для настройки доступа на базе примера. Отредактировав файл, переместите его в **/etc/udev/rules.d**.

Указанные правила определяют основные переменные, которые могут использоваться другими правилами.

Переменные в **10-dm.rules**:

- **DM_NAME**: имя проекции устройства;
- **DM_UUID**: идентификатор проекции устройства;
- **DM_SUSPENDED**: устройство приостановлено;
- **DM_UDEV_RULES_VSN**: версия. Проверяется другими правилами, чтобы убедиться, что указанные выше переменные определены официальными правилами Device Mapper.

Переменные в **11-dm-lvm.rules**:

- **DM_LV_NAME**: логический том;
- **DM_VG_NAME**: группа томов;
- **DM_LV_LAYER**: уровень LVM.

Для настройки доступа к конкретному устройству Device Mapper перечисленные переменные можно добавить в файл **12-dm-permissions.rules**.

A.3.2. Команды для работы с udev

Таблица A.2, «Команды dmsetup для работы с udev» содержит список команд **dmsetup** для работы с **udev**.

Таблица A.2. Команды dmsetup для работы с udev

Команда	Описание
dmsetup udevcomplete	Сообщает о завершении обработки правил и разрешает продолжение ожидающего процесса (вызывается из **95-dm-notify.rules**).
dmsetup udevcomplete_all	Используется при отладке для возобновления всех ожидающих процессов.
dmsetup udevcookies	Используется при отладке для просмотра существующих cookie-файлов.
dmsetup udevcreatecookie	Используется для создания cookie-файлов вручную.
dmsetup udevreleasecookie	Ожидает завершения обработки всех процессов для выбранного cookie.

Ниже приведен список параметров для перечисленных команд.

--udevcookie

Обязательный параметр. Используется вместе с **udevcreatecookie** и **udevreleasecookie**:

```
COOKIE=$(dmsetup udevcreatecookie)
  dmsetup команда --udevcookie $COOKIE ....
  dmsetup команда --udevcookie $COOKIE ....
  ....
  dmsetup команда --udevcookie $COOKIE ....
dmsetup udevreleasecookie --udevcookie $COOKIE
```

Дополнительно можно экспортировать переменную в окружение процесса:

```
export DM_UDEV_COOKIE=$(dmsetup udevcreatecookie)
  dmsetup команда ...
  dmsetup команда ...
  ...
  dmsetup команда ...
```

--noudevrules

Отключает правила udev. За создание файлов и ссылок будет отвечать **libdevmapper**. Этот аргумент используется для отладки **udev**.

--noudevsync

Отключает синхронизацию **udev**. Используется для отладки.

Более подробную информацию о **dmsetup** можно найти на справочной странице **dmsetup**(8).

Команды LVM поддерживают следующие параметры для интеграции с **udev**:

- **--noudevrules**: отключает правила **udev**.
- **--noudevsync**: отключает синхронизацию **udev**.

Ниже перечислены параметры, которые могут быть добавлены в **lvm.conf**.

- **udev_rules**: включает и отключает **udev_rules** для всех команд LVM2.
- **udev_sync**: включает и отключает синхронизацию **udev** для всех команд LVM.

lvm.conf содержит подробные комментарии.

ПРИЛОЖЕНИЕ B. ФАЙЛЫ КОНФИГУРАЦИИ LVM

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

В **lvm.conf** могут быть указаны дополнительные файлы конфигурации. Последующие настройки будут переопределять предыдущие. Чтобы отобразить текущие настройки, выполните **lvm dumpconfig**.

Раздел C.2, «Теги узлов» содержит инструкции по добавлению других файлов конфигурации.

B.1. ФАЙЛЫ КОНФИГУРАЦИИ LVM

Далее перечислены файлы конфигурации LVM.

/etc/lvm/lvm.conf

Основной файл конфигурации, к которому обращаются утилиты.

etc/lvm/lvm_*тег_узла*.conf

Каждому тегу соответствует отдельный файл **lvm_*тег_узла*.conf**. Если он содержит другие теги, тогда имена соответствующих файлов конфигурации будут добавлены в список для последующего чтения (см. Раздел C.2, «Теги узлов»).

Также используются следующие файлы:

/etc/lvm/.cache

файл кэширования фильтров имен устройств (может быть изменен).

/etc/lvm/backup/

каталог, где хранятся автоматически созданные резервные копии метаданных групп томов (может быть изменен).

/etc/lvm/archive/

каталог, где хранятся архивы автоматически созданных резервных копий метаданных групп томов (настраивается в зависимости от длины истории архивации и пути)

/var/lock/lvm/

В структуре с одним узлом файлы блокировки предотвращают повреждение данных при параллельной работе программ. В кластере будет использоваться блокирование DLM.

B.2. ПРИМЕР LVM.CONF

Пример **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 md device's stripe-width.
    # 1 enables; 0 disables.
    md_chunk_alignment = 1

    # Default alignment of the start of a data area in MB.  If set to 0,
    # a value of 64KB will be used.  Set to 1 for 1MiB, 2 for 2MiB, etc.
    # default_data_alignment = 1

    # By default, the start of a PV's data area will be a multiple of
    # the 'minimum_io_size' or 'optimal_io_size' exposed in sysfs.
    # - minimum_io_size - the smallest request the device can perform
    #   w/o incurring a read-modify-write penalty (e.g. MD's chunk size)
    # - optimal_io_size - the device's preferred unit of receiving I/O
    #   (e.g. MD's stripe width)
```

```
    # minimum_io_size is used if optimal_io_size is undefined (0).
    # If md_chunk_alignment is enabled, that detects the optimal_io_size.
    # This setting takes precedence over md_chunk_alignment.
    # 1 enables; 0 disables.
    data_alignment_detection = 1

    # Alignment (in KB) of start of data area when creating a new PV.
    # md_chunk_alignment and data_alignment_detection are disabled if set.
    # Set to 0 for the default alignment (see: data_alignment_default)
    # or page size, if larger.
    data_alignment = 0

    # By default, the start of the PV's aligned data area will be shifted
by
    # the 'alignment_offset' exposed in sysfs.  This offset is often 0 but
    # may be non-zero; e.g.: certain 4KB sector drives that compensate for
    # windows partitioning will have an alignment_offset of 3584 bytes
    # (sector 7 is the lowest aligned logical block, the 4KB sectors start
    # at LBA -1, and consequently sector 63 is aligned on a 4KB boundary).
    # But note that pvcreate --dataalignmentoffset will skip this
detection.
    # 1 enables; 0 disables.
    data_alignment_offset_detection = 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

    # During each LVM operation errors received from each device are
counted.
    # If the counter of a particular device exceeds the limit set here, no
    # further I/O is sent to that device for the remainder of the
respective
    # operation. Setting the parameter to 0 disables the counters
altogether.
    disable_after_error_count = 0

    # Allow use of pvcreate --uuid without requiring --restorefile.
    require_restorefile_with_uuid = 1
}

# This section allows you to configure the way in which LVM selects
# free space for its Logical Volumes.
#allocation {
#    When searching for free space to extend an LV, the "cling"
#    allocation policy will choose space on the same PVs as the last
#    segment of the existing LV.  If there is insufficient space and a
#    list of tags is defined here, it will check whether any of them are
#    attached to the PVs concerned and then seek to match those PV tags
#    between existing extents and new extents.
#    Use the special tag "@*" as a wildcard to match any PV tag.
#
```

```
#    Example: LVs are mirrored between two sites within a single VG.
#    PVs are tagged with either @site1 or @site2 to indicate where
#    they are situated.
#
#    cling_tag_list = [ "@site1", "@site2" ]
#    cling_tag_list = [ "@*" ]
#}

# 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 {

    # 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"

    # Since version 2.02.54, the tools distinguish between powers of
    # 1024 bytes (e.g. KiB, MiB, GiB) and powers of 1000 bytes (e.g.
    # KB, MB, GB).
    # If you have scripts that depend on the old behaviour, set this to 0
    # temporarily until you update them.
    si_unit_consistency = 1

    # 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 "lvm2".
    # format = "lvm2"

    # 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.
    # Type 4 uses read-only locking which forbids any operations that
might
    # change metadata.
    locking_type = 1

    # Set to 0 to fail when a lock request cannot be satisfied
immediately.
    wait_for_locks = 1

    # If using external locking (type 2) and initialisation fails,
    # with this set to 1 an attempt will be made to use the built-in
    # clustered locking.
```

```
    # If you are using a customised locking_library you should set this to
0.
    fallback_to_clustered_locking = 1

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

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

    # Whenever there are competing read-only and read-write access
requests for
    # a volume group's metadata, instead of always granting the read-only
    # requests immediately, delay them to allow the read-write requests to
be
    # serviced.  Without this setting, write access may be stalled by a
high
    # volume of read-only requests.
    # NB. This option only affects locking_type = 1 viz. local file-based
    # locking.
    prioritise_write_locks = 1

    # 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"

    # Treat any internal errors as fatal errors, aborting the process that
    # encountered the internal error. Please only enable for debugging.
    abort_on_internal_errors = 0

    # If set to 1, no operations that change on-disk metadata will be
permitted.
    # Additionally, read-only commands that encounter metadata in need of
repair
    # will still be allowed to proceed exactly as if the repair had been
    # performed (except for the unchanged vg_seqno).
    # Inappropriate use could mess up your system, so seek advice first!
    metadata_read_only = 0
```

```
}

activation {
    # Set to 0 to disable udev synchronisation (if compiled into the
binaries).
    # Processes will not wait for notification from udev.
    # They will continue irrespective of any possible udev processing
    # in the background.  You should only use this if udev is not running
    # or has rules that ignore the devices LVM2 creates.
    # The command line argument --nodevsync takes precedence over this
setting.
    # If set to 1 when udev is not running, and there are LVM2 processes
    # waiting for udev, run 'dmsetup udevcomplete_all' manually to wake
them up.
    udev_sync = 1

    # Set to 0 to disable the udev rules installed by LVM2 (if built with
    # --enable-udev_rules). LVM2 will then manage the /dev nodes and
symlinks
    # for active logical volumes directly itself.
    # N.B. Manual intervention may be required if this setting is changed
    # while any logical volumes are active.
    udev_rules = 1

    # 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. This applies to automatic repairs (when the mirror is
being
    # monitored by dmeventd) and to manual lvconvert --repair when
    # --use-policies is given.
    #
    # "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.
    #
    # "allocate_anywhere" - Not yet implemented. Useful to place the log
device
    #            temporarily on same physical volume as one of the mirror
    #            images. This policy is not recommended for mirror devices
    #            since 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_image_fault_policy = "remove"

    # 'snapshot_autoextend_threshold' and 'snapshot_autoextend_percent'
define
    # how to handle automatic snapshot extension. The former defines when
the
    # snapshot should be extended: when its space usage exceeds this many
    # percent. The latter defines how much extra space should be allocated
for
```

```
    # the snapshot, in percent of its current size.
    #
    # For example, if you set snapshot_autoextend_threshold to 70 and
    # snapshot_autoextend_percent to 20, whenever a snapshot exceeds 70%
usage,
    # it will be extended by another 20%. For a 1G snapshot, using up 700M
will
    # trigger a resize to 1.2G. When the usage exceeds 840M, the snapshot
will
    # be extended to 1.44G, and so on.
    #
    # Setting snapshot_autoextend_threshold to 100 disables automatic
    # extensions. The minimum value is 50 (A setting below 50 will be
treated
    # as 50).

    snapshot_autoextend_threshold = 100
    snapshot_autoextend_percent = 20

    # While activating devices, I/O to devices being (re)configured is
    # suspended, and as a precaution against deadlocks, LVM2 needs to pin
    # any memory it is using so it is not paged out.  Groups of pages that
    # are known not to be accessed during activation need not be pinned
    # into memory.  Each string listed in this setting is compared against
    # each line in /proc/self/maps, and the pages corresponding to any
    # lines that match are not pinned.  On some systems locale-archive was
    # found to make up over 80% of the memory used by the process.
    # mlock_filter = [ "locale/locale-archive", "gconv/gconv-
modules.cache" ]

    # Set to 1 to revert to the default behaviour prior to version 2.02.62
    # which used mlockall() to pin the whole process's memory while
activating
    # devices.
    use_mlockall = 0

    # Monitoring is enabled by default when activating logical volumes.
    # Set to 0 to disable monitoring or use the --ignoremonitoring option.
    monitoring = 1

    # When pvmove or lvconvert must wait for the kernel to finish
    # synchronising or merging data, they check and report progress
    # at intervals of this number of seconds.  The default is 15 seconds.
    # If this is set to 0 and there is only one thing to wait for, there
    # are no progress reports, but the process is awoken immediately the
    # operation is complete.
    polling_interval = 15
}


####################
# 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

    # Default number of copies of metadata to maintain for each VG.
    # If set to a non-zero value, LVM automatically chooses which of
    # the available metadata areas to use to achieve the requested
    # number of copies of the VG metadata.  If you set a value larger
    # than the the total number of metadata areas available then
    # metadata is stored in them all.
    # The default value of 0 ("unmanaged") disables this automatic
    # management and allows you to control which metadata areas
    # are used at the individual PV level using 'pvchange
    # --metadataignore y/n'.

    # vgmetadatacopies = 0

    # 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
    # the snapshot exceeds 80%. The warning is repeated when 85%, 90% and
    # 95% of the snapshot is filled.

    snapshot_library = "libdevmapper-event-lvm2snapshot.so"

    # Full path of the dmeventd binary.
    #
    # executable = "/sbin/dmeventd"
}
```

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, а также узлам в кластере. Снимкам теги не присваиваются.

Теги можно передавать командам вместо аргументов PV, VG, LV. Перед тегом следует указать"@" — тогда при разборе выражения он будет замещен объектами с этим тегом.

В Red Hat Enterprise Linux 6.1 тег LVM может содержать до 1024 знаков, но не может начинаться с дефиса.

Раньше допустимые знаки включали [A-Za-z0-9_+.-]. Начиная с Red Hat Enterprise Linux 6.1, этот список расширен и включает "/", "=", "!", ":", "#", "&".

Теги могут быть применены только внутри группы томов. При исключении из группы физический том лишится тега, так как теги сохраняются только в метаданных группы и при удалении физического тома они также будут удалены.

Следующая команда вернет список логических томов с тегом **database**.

```
lvs @database
```

C.1. ДОБАВЛЕНИЕ И УДАЛЕНИЕ ТЕГОВ

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

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

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

pvchange, **vgchange**, **lvchange** теперь могут включать несколько аргументов **--addtag** и **--deltag**. Например, следующая команда удалит теги **T9**, **T10** из группы **grant** и добавит **T13**, **T14**.

```
vgchange --deltag T9 --deltag T10 --addtag T13 --addtag T14 grant
```

C.2. ТЕГИ УЗЛОВ

В кластерном окружении теги узлов могут быть определены в файле конфигурации. Например, если в секция **tags** содержит **hosttags = 1**, в качестве тега будет автоматически присваиваться имя компьютера. Это позволит использовать один и тот же файл конфигурации для всех машин; при этом поведение может отличаться в зависимости от имени компьютера.

Приложение B, *Файлы конфигурации LVM* содержит информацию о файлах конфигурации.

При разборе тега будет выполнено обращение к файлу конфигурации lvm_*тег*.conf. Если он содержит определения новых тегов, имена соответствующих файлов конфигурации будут добавлены к списку имен для последующего чтения.

Так, приведенное выражение будет определять **tag1** и дополнительно **tag2**, если имя компьютера — **host1**.

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

C.3. АКТИВАЦИЯ ТОМОВ С ПОМОЩЬЮ ТЕГОВ

В файле конфигурации можно задать логические тома для активации на текущем компьютере. Например, приведенное далее выражение определяет фильтр для 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" ] }
```

"@*" обозначает, что соответствие будет успешно только в том случае, если тег метаданных совпадет с тегом узла.

Представим, что файлы конфигурации на всех компьютерах в пределах кластера содержат выражение

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

Для активации **vg1/lvol2** на узле **db2** надо выполнить следующее:

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

Имена компьютеров будут сохранены в метаданных группы томов.

ПРИЛОЖЕНИЕ D. МЕТАДАННЫЕ ГРУППЫ ТОМОВ

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

Если группа включает много физических томов, хранение всех избыточных копий довольно неэффективно. Тогда с помощью опции **--metadatacopies 0** команды **pvcreate** можно создать физический том без копии 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 в каждой группе должна присутствовать хотя бы одна копия метаданных.

Основные метаданные хранятся в формате ASCII. Область метаданных представляет собой циклический буфер. 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 **pvcreate** размещает метку во втором секторе (размер сектора — 512 байт). 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 значением NULL.
- Списки, содержащие информацию о расположении областей метаданных. Списки завершаются значением 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
- Информацию о группе томов

Информация о группе томов включает:

- Имя и уникальный идентификатор
- Номер версии, который увеличивается на единицу при каждом обновлении метаданных
- Параметры: Чтение или 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, такие как возможность выделения пространства данного тома
  - Смещение первого экстента в пределах физического тома (в секторах)
  - Число экстентов
- Неупорядоченный список логических томов. Информация включает:
  - Упорядоченный список сегментов логического тома. Метаданные сегмента включают соответствия упорядоченного списка сегментов логических или физических томов

D.3. ПРИМЕР МЕТАДАННЫХ

Пример метаданных для группы **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 1-3.400 **2013-10-31** **Rüdiger Landmann**

Rebuild with publican 4.0.0

Издание 1-3 **2012-07-18** **Anthony Towns**

Rebuild for Publican 3.0

Издание 2.0-1 **Thu May 19 2011** **Steven Levine**

Исходная версия документа для Red Hat Enterprise Linux 6.1

BZ#694619
Описание `cling`.

BZ#682649
Добавлено предупреждение о выполнении нескольких команд создания зеркального тома.

BZ#674100
Добавлен пример вывода `dmsetup ls --tree`.

BZ#694607
Добавлена информация о добавлении аргументов --addtag и --deltag в одну команду.

BZ#694604
Добавлена информация о поддержке расширенного списка знаков в тегах.

BZ#694611
Описание поддержки зеркальных томов с чередованием.

BZ#694616
Описание поддержки снимков зеркальных томов.

BZ#694618
Описание поддержки снимков кластерных томов.

BZ#682648
Добавлена информация о переносе журнала при переносе компонента зеркала.

BZ#661530
Обновлен пример `cluster.conf`.

BZ#642400
Добавлено примечание о поддержке журнала кластера на узлом с наименьшим идентификатором.

BZ#663462
Удалена устаревшая информация о виртуальной машине Xen.

Издание 1.0-1 **Wed Nov 10 2010** **Steven Levine**

Исходная версия для Red Hat Enterprise Linux 6

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

Символы

C

L