ALT Linux 4.0 Server
руководство пользователя

Александр Боковой,
Антон Бояршинов,
Александр Блохин,
Олег Власенко,
Андрей Горев,
Владимир Жуков,
Михаил Забалуев,
Артём Золочевский,
Станислав Иевлев,
Георгий Курячий,
Дмитрий Левин,
Кирилл Маслинский,
Михаил Шигорин

Москва, ALT Linux,
2007

Составление, вёрстка: Кирилл Маслинский

Оглавление

Условные обозначения

Некоторые фрагменты текста набраны `моноширинным шрифтом`: это названия объектов системы (имена файлов, программ и утилит), которые можно увидеть на экране и вводить с клавиатуры, а также клавиатурные комбинации. Названия элементов управления графического интерфейса заключены в кавычки.

Команды, предназначенные для ввода, следуют после знака «приглашения» командной строки «`$`». В случае, если команду следует ввести от имени администратора (суперпользователя `root`) — «`#`».

В тексте встречаются ссылки на дополнительные источники информацции: другие разделы данного руководства (с указанием названия раздела и номера страницы), адреса ресурсов Интернет и отсылки к экранной документации, входящией в состав дистрибутива ALT Linux. Ссылки на страницы руководства (manual pages) оформляются традиционным для Linux способом: название утилиты, за которым в скобках следует номер раздела, к которому относится данное руководство, например, `mount(8)`. Чтобы просмотреть соответствующее руководство, необходимо выполнить команду `man mount`.

Встречающиеся в тексте термины при первом употреблении выделены жирным шрифтом, как и другая информация, заслуживающая повышенного внимания.

Часть I

Установка

Начало установки: загрузка системы

Загрузка с установочного диска начинается с меню, в котором перечислено несколько вариантов загрузки, причём установка системы — это только одна из возможностей. Из этого же меню можно запустить программу для восстановления системы или проверки памяти. Можно получить справку по любому пункту меню, выбрав этот пункт и нажав `F1`. Кроме установки с диска доступно несколько вариантов сетевой установки (об этом рассказано ниже).

Выбор языка меню и справки осуществляется нажатием `F2`. От выбора языка в загрузчике зависит, во-первых, язык интерфейса загрузчика и программы установки, и во-вторых, какие языки будут доступны в списке языков установки — кроме конкретного выбранного языка, в список будут включены и другие языки данной территории. По умолчанию предлагается английский язык, если принять этот вариант, в списке языков установки будут перечислены все возможные языки.

По нажатию `F3` открывается меню доступных видеорежимов (разрешений экрана). Это разрешение будет использоваться во время установки и загрузки установленной системы. По умолчанию выбирается максимальное разрешение из возможных.

Чтобы начать процесс установки, нужно выбрать пункт меню «Установка» и нажать `Enter`. В начальном загрузчике установлено небольшое время ожидания: если в этот момент не предпринимать никаких действий, то будет загружена та система, которая уже установлена на жестком диске. Если вы пропустили нужный момент, перезагрузите компьютер и вовремя выберите пункт «Установка».

Начальный этап установки не требует вмешательства пользователя: происходит автоматическое определение оборудования и запуск компонентов программы установки. Сообщения о том, что происходит на этом этапе, отображаются на экране.

Сетевая установка

Установка ALT Linux возможна не только с лазерного диска, ее можно производить и по сети. Обязательное условие для этого — наличие на сервере дерева файлов, аналогичного содержимому установочного диска, и внешний носитель с начальным загрузчиком.

Кнопка `F4` позволяет выбрать источник установки: FTP, HTTP или NFS-сервер. Нужно указать имя или IP-адрес сервера и каталог, в котором размещен дистрибутив ALT Linux. В случае установки по протоколу FTP может понадобиться также ввести имя пользователя и пароль.

При сетевой установке может понадобиться определить параметры соединения с сервером, в этом случае на экране будут запросы, например, с предложением выбрать сетевую карту (если их несколько) или указать тип IP-адреса: статический (потребуется вписать его самостоятельно) или динамический (DHCP).

После успешного соединения с сервером в память компьютера будет загружен образ установочного диска, после чего начнется установка системы так же, как при установке с диска.

Последовательность установки

Внимание

До того, как будет произведена установка базовой системы на жёсткий диск, программа установки работает с образом системы, загруженном в оперативной памяти компьютера.

Если инициализация оборудования завершилась успешно, будет запущен графический интерфейс программы-установщика и откроется меню выбора **основного языка** — языка интерфейса программы установки и устанавливаемой системы. В списке, помимо доступных языков региона (выбранного на этапе начальной загрузки), указаны английский язык и **POSIX** — стандартное языковое окружение (на основе английского), которое всегда присутствует в Linux. Выберите язык из списка и нажмите «Применить» для перехода к началу установки.

Процесс установки разделен на шаги: каждый шаг посвящен настройке или установке определенного свойства системы. Шаги нужно проходить поседовательно, переход к следующему шагу происходит по нажатию кнопки «Далее». При помощи кнопки «Назад» при необходимости можно вернуться к уже пройденному шагу и изменить настройки. Однако на этом этапе установки возможность перехода к пердыдущему шагу ограничена теми шагами, где нет зависимости от данных, введённых ранее.

Если по каким-то причинам возникла необходимость прекратить установку, воспользуйтесь сочетанием клавиш *Ctrl+Alt+Del* или нажимте `Reset` на системном блоке компьютера. Помните, что совершенно *безопасно* прекращать установку только до шага «Подготовка диска», поскольку до этого момента не производится никаких изменений на жёстком диске. Если прервать установку между шагами «Подготовка диска» и «Установка загрузчика», вероятно, что после этого с жёсткого диска не сможет загрузиться ни одна из установленных систем.

Технические сведения о ходе установки можно посмотреть, нажав *Ctrl+Alt+F1*, вернуться к программе установки — *Ctrl+Alt+F7*. По нажатию *Ctrl+Alt+F2* откроется отладочная виртуальная консоль.

Во время установки системы выполняются следующие шаги:

1) Лицензионное соглашение
2) Настройка клавиатуры
3) Подготовка диска
4) Установка базовой системы
5) Установка загрузчика
6) Администратор системы
7) Системный пользователь
8) Дополнительные пакеты
9) Настройка сети
10) Дата/Время
11) Завершение установки

Лицензионное соглашение

Перед продолжением установки следует внимательно прочитать условия лицензии. Лицензия относится ко всему дистрибутиву ALT Linux. Если вы согласны с условиями лицензии, отметьте пункт «Принимаю» и нажмите «Далее».

Настройка клавиатуры

Раскладка клавиатуры — это привязка букв, цифр и специальных символов к клавишам на клавиатуре. Помимо ввода символов на основном языке, в любой системе Linux необходимо иметь возможность вводить латинские символы (имена команд, файлов и т. п.), для чего обычно используется стандартная английская раскладка клавиатуры. Переключение между раскладками осуществляется при помощи специально зарезервированных для этого клавиш. Для русского языка доступны следующие варианты переключения раскладки:

- Клавиша *Capslock*
- Клавиши *Control* и *Shift* одновременно
- Клавиша *Control*
- Клавиша *Alt*

В случае, если выбранный основной язык имеет всего одну раскладку (например, при выборе английского языка в качесве основного), эта единственная раскладка будет принята авоматически, а сам шаг не будет отображен в интерфейсе.

Подготовка диска

Переход к этому шагу может занять некоторое время. Время ожидания может быть разным и зависит от производительности компьютера, объема жесткого диска, количества разделов на нем и т. д.

На этом этапе подготавливается площадка для установки ALT Linux, в первую очередь — выделяется свободное место на диске. Для серверной установки потребуется не менее 6144 Мб на одном или нескольких жёстких дисках компьютера.

Выбор профиля разбиения диска

В списке устройств перечислены диски, на которые возможно установить Linux, рядом с каждым диском указан его размер и объём свободного места. Если дисков несколько, выберите те из них, на которых вы хотите разместить будущую операционную систему. Ниже перечислены доступные профили разбиения диска. Профиль — это шаблон распределения места на диске для установки Linux. Можно выбрать один из трех профилей:

- Установка рабочей станции
- Установка сервера
- Вручную

Первые два профиля предполагают автоматическое разбиение диска. Они ориентированы на среднестатистические рабочие станции и серверы и должны подойти для большинства пользователей. Если вы хотите иметь возможность ознакомиться с предоженным вариантом и, возможно, отредактировать его, убедитесь, что выбран пункт «Предложить сделать мои изменения после применения профиля». Для того, чтобы применить профиль, выберите соответствующий пункт и нажмите «Далее».

От возможности редактировать результат разбиения можно отказаться, сняв выделение с соответствующего пункта. В этом случае никакой информации о распределении дискового пространства на экране отображаться не будет. После осуществления физических изменений на жестком диске начнется установка базовой системы. Этот вариант отлично подойдет для установки на чистый диск.

Рядом с названием каждого профиля указан минимальный объём свободного места на диске, требуемый для установки в соответствии с данным профилем (3312 Мб для рабочей станции и 6144 Мб для сервера). Если на выбранных дисках свободного места недостаточно, будет выведено сообщение об ошибке: «Невозможно применить профиль, недостаточно места на диске». Если вы готовы очистить место, удалив данные, которые уже есть на диске, выберите пункт «Очистить все диски перед применением профиля».

Внимание

Будьте осторожны! В этом случае будут удалены **все** данные со **всех** дисков без возможности восстановления. Рекомендуется использовать эту возможность только в том случае, если вы уверены, что диски не содержат **никаких ценных данных**.

При необходимости освободить **часть** дискового пространства следует воспользоваться профилем разбиения вручную. Вы сможете удалить некоторые из существующих разделов или содержащиеся в них файловые системы. После этого можно создать необходимые разделы самостоятельно или вернуться к шагу выбора профиля и применить один из автоматических профилей. Выбор этой возможности требует знаний об устройстве диска и технологиях его разбиения, поэтому сначала рекомендуется внимательно прочитать главу «Планирование жесткого диска» данного руководства.

После выбора одного из профилей автоматического разбиения по умолчанию (при сохранении возможности редактирования результата) будет показано дерево устройств, отражающее результат применения профиля: предложенную схему разбиения жёсткого диска на разделы и распределение точек монтирования. Если результат вас по каким-то причинам не устраивает, прямо сейчас можно его отредактировать.

Наиболее широкие возможности по организации дискового пространства предлагает профиль ручного разбиения. При его выборе будет предложен тот же самый интерфейс для работы с разделами. Разница только в том, что в случае ручного разбиения на диске ничего не будет изменено автоматически и вы увидите то расположение разделов, которое было до установки. Необходимую информацию о работе с диском и принципах ручного разбиения можно найти в разделе «Разбиение диска средствами программы установки» [стр. 39].

Внимание

Окончательное применение всех изменений к дискам произойдёт по нажатии «Далее» после просмотра результатов разбиения (если был выбран автоматический профиль) или по завершении ручного разбиения. До этого момента никаких физических изменений на диске не производится.

Основные сведения, которые необходимы, чтобы спланировать разбиение диска самостоятельно, приведены в разделе «Планирование диска» [стр. 28], там же разобрано несколько типичных способов разбиения диска.

По нажатию «Далее» будет произведена запись новой таблицы разделов на диск и форматирование разделов. Разделы, только что созданные на диске программой установки, пока не содержат данных и поэтому форматируются без предупреждения. Уже существовашие, но измененные разделы, которые будут отформатированы, помечаются специальным значком в колонке «Файловая система» слева от названия. Если вы уверены в том, что подготовка диска завершена, подтвердите переход к следующему шагу нажатием кнопки «ОК».

Не следует форматировать разделы с теми данными, которые вы хотите сохранить, например, с пользовательскими данными (`/home`) или с другими операционными системами. С другой стороны, отформатировать можно любой раздел, который вы хотите «очистить» (т. е. удалить все данные).

Установка базовой системы

На этом этапе происходит установка стартового набора программ, необходимых для запуска и первоначальной настройки Linux. Далее в процессе установки у вас будет возможность выбрать и установить все необходимые вам для работы приложения.

Установка происходит автоматически в два этапа:

- Получение пакетов
- Установка пакетов

Получение пакетов осуществляется с источника, выбранного на этапе начальной загрузки. При сетевой установке (по протоколу FTP или HTTP) время выполнения этого шага будет зависеть от скорости соединения и может быть значительно большим, чем при установке с лазерного диска.

Когда базовая система будет установлена, вы сможете произвести первичную настройку, в частности — настроить сетевое обрудование и сетевые подключения. Изменить свойства системы, которые были заданы при установке (например, язык системы) можно будет в любой момент как через веб-интерфейс, так и при помощи стандартных для Linux средств и специализированных модулей управления, включённых в дистрибутив.

Установка базовой системы может занять некоторое время, которое можно посвятить, например, чтению руководства.

Установка загрузчика

Внимание

Начиная с этого шага программа установки работает с файлами только что установленной базовой системы. Все последующие изменения можно будет совершить после завершения установки посредством редактирования соответствующих конфигурационных файлов.

Загрузчик Linux — программа, которая позволяет загружать Linux и другие операционные системы. Если на вашем компьютере будет установлен только Linux, то здесь не нужно ничего изменять, просто нажмите «Далее».

Если же вы планируете использовать и другие операционные системы, уже установленные на этом компьютере, тогда имеет значение, на каком жёстком диске или разделе будет расположен загрузчик. В большинстве случаев программа установки правильно подберёт расположение загрузчика, однако чтобы быть уверенным, что все операционные системы будут загружаться правильно, обратитесь к разделу «Настройка загрузки» [стр. 45].

Опытным пользователям может пригодиться возможность тонкой настройки загрузчика (кнопка «Дополнительные настройки»). Параметры, которые можно здесь изменять, напрямую соотносятся с соответствующими параметрами конфигурационного файла загрузчика LILO (`/etc/lilo.conf`). Для простоты сохранены латинские названия параметров, об их значении можно справиться в документации по LILO (`lilo.conf(5)`).

Пользователи

Linux — это многопользовательская система. На практике это означает, что для работы в системы нужно в ней *зарегистрироваться*, т. е. дать понять системе, кто именно находится за монитором и клавиатурой. Наиболее распространенный способ регистрации на сегодняшний день — использование **системных имен** (login name) и паролей. Это надежное средство убедиться, что с системой работает тот, кто нужно, если пользователи хранят свои пароли в секрете и если пароль достаточно сложен и не слишком короток (иначе его легко угадать или подобрать).

Администратор системы

В любой системе Linux всегда присутствует один специальный пользователь — администратор, он же **суперпользователь**, для него зарезервировано стандартное системное имя — root.

Внимание

Стоит запомнить пароль root — его нужно будет вводить, чтобы получить право изменять настройки системы с помощью стандартных средств настройки ALT Linux.

При наборе пароля вместо символов на экране высвечиваются звёздочки. Чтобы избежать опечатки при вводе пароля, его предлагается ввести дважды. Можно воспользоваться автоматическим созданием пароля, выбрав «Создать автоматически». Вам будет предложен случайно сгенерированный и достаточно надёжный вариант пароля. Можно принять автоматически сгенерированный пароль, (не забудьте при этом запомнить пароль!), или запросить другой вариант пароля при помощи кнопки «Сгенерировать».

Администратор отличается от всех прочих пользователей тем, что ему позволено производить **любые**, в том числе самые разрушительные, изменения в системе. Поэтому выбор пароля администратора — очень важный момент для **безопасности**: любой, кто сможет ввести его правильно (узнать или подобрать), получит неограниченный доступ к системе. Даже ваши собственные неосторожные действия от имени root могут иметь катастрофические последствия для всей системы.

Системный пользователь

Помимо администратора (root) в систему необходимо добавить по меньшей мере одного **обычного пользователя**. Работа от имени администратора считается опасной (можно по неосторожности повредить систему), поэтому повседневную работу в Linux следует выполнять от имени обычного пользователя, полномочия которого ограничены.

При добавлении пользователя предлагается ввести имя учётной записи (login name) пользователя. Имя учётной записи всегда представляет собой одно слово, состоящее только из строчных латинских букв (заглавные запрещены), цифр и символа подчёркивания «_» (причём «_» не может стоять в начале слова). Дважды вводится пароль пользователя, чтобы исключить опечатки. Так же, как при выборе пароля администратора (root), можно создать пароль автоматички.

В процессе установки предлагается создать только одну учётную запись обычного пользователя — чтобы от его имени системный администратор мог выполнять задачи, которые не требуют привилегий суперпользователя.

Учётные записи для всех прочих пользователей системы можно будет создать в любой момент после установки системы.

Дополнительные пакеты

В любом дистрибутиве ALT Linux доступно значительное количество программ (до нескольких тысяч), часть из которых составляет саму операционную систему, а все остальные — это прикладные программы и утилиты.

В операционной системе Linux все операции установки и удаления производятся над **пакетами** — отдельными компонентами системы. Пакет и программа соотносятся неоднозначно: иногда одна программа состоит из нескольких пакетов, иногда один пакет включает несколько программ.

В процессе установки системы обычно не требуется детализированный выбор компонентов на уровне пакетов — это требует слишком много времени и знаний от проводящего установку. Тем более, что комплектация дистрибутива подбирается таким образом, чтобы из имеющихся программ можно было составить полноценную рабочую среду для соответствующей аудитории пользователей. Поэтому в процессе установки системы пользователю предлагается выбрать из небольшого списка *групп пакетов*, объединяющих пакеты, необходимые для решения наиболее распространённых задач.

Выбрав необходимые группы, следует нажать «Далее». На экране появится информация о количестве устанавливаемых пакетов и объеме дискового пространства, который будет занят после их установки. По нажатию кнопки «Подробности...» откроется список, в котором перечислены конкретные пакеты, которые будут установлены в системе. Если вы готовы подтвердить изменения, нажмите «Да» — сразу после этого начнется установка пакетов. Чтобы вернуться к списку групп пакетов и отредактировать свой первоначальный выбор, нажмите «Нет».

Настройка сети

Существует ряд сетевых параметров, которые являются общими для всех подключений к сети и должны быть определены даже тогда, когда компьютер не подключён ни к какой сети. Для подключения к локальной сети необходимо также настроить **сетевое подключение**, которое в Linux также принято называть **сетевой интерфейс**. Настройки этих двух типов производятся в закладках «Общие сетевые настройки» и «IP-интерфейсы».

Общие настройки сети

Укажите **Имя хоста** (host name) — имя компьютера вида `computer.domain`. Несмотря на то, что этот параметр никому из соседних компьютеров в сети не передаётся (в отличие, скажем, от имени компьютера в Windows-сети), его используют многие сетевые службы, например, почтовый сервер. Если компьютер не подключён к локальной сети, имя хоста может выглядеть как угодно, можно оставить значение по умолчанию (`localhost.localdomain`).

При работе и настройке сетевых служб часто приходится использовать символьные имена других машин в сети. Чтобы система преобразовала их в IP-адреса, используются DNS-серверы. В случае локальной сети преобразования осуществяет локальный DNS-сервер. за пределами локальной сети используются DNS-серверы вышестоящих Интернет-провайдеров. Все необходимые серверы DNS перечисляются в соответствующем поле (по одному на строку или через запятую).

Если в поле «Автопоиск в доменах» перечислить наиболее часто используемые домены (например, `domain`), то можно пользоваться неполными именами машин (`computer` вместо `computer.domain`)

В случае наличия в сети DHCP-сервера общие настройки сети могут быть получены автоматически. Отметьте пункты «Получать имя хоста автоматически (DHCP)» и «Получать настройки DNS автоматически (DHCP)», если хотите воспользоваться этой возможностью. Обратите внимание, что для этого необходимо включить использование DHCP в настройках IP-интерфейса. Если сервера DHCP нет, при подключении к сети все необходимые параметры (имя хоста, шлюз по умолчанию, адреса серверов DNS) нужно выяснить у администратора сети или у Интернет-провайдера и вписать вручную.

IP-интерфейсы

В случае локальной сети каждое подключение обычно привязывается к определённому физическому устройству — сетевой (Ethernet) карте. Чтобы настроить подключение, на закладке «Интерфейс» выберите из списка одно из обнаруженных сетевых устройств. При наличии беспроводных интерфейсов их настройки осуществляются по нажатию кнопки «Настройки для беспроводных интерфейсов».

Для того, чтобы включить выбранный интерфейс, отметьте пункт «Интерфейс включен».

В полях «IP-адрес» и «Маска сети» должны быть указаны обязательные параметры каждого узла IP-сети. Первый параметр — уникальный идентификатор машины, от второго напрямую зависит, к каким машинам локальной сети данная машина будет иметь доступ. Если требуется выход во внешнюю сеть, не забудьте про параметр «Шлюз по умолчанию».

При наличии DHCP-сервера все вышеперечисленные параметры можно получить автоматически, для этого отметьте пункт «Использовать DHCP». Если DHCP-сервера нет, потребуется указать IP-адрес и сетевую маску явно. Параметры настройки сетевого подключения можно узнать у администратора сети.

Дата/Время

Для установки даты и времени достаточно правильно указать часовой пояс. Для этого в соответствующих списках выберите страну, а затем регион, в котором вы находитесь. Поиск по спискам можно ускорить, набирая на клавиатуре первые буквы искомого слова.

Завершение установки

На экране последнего шага установки отображается информация о местонахождении протокола установки (каталог `/root/.install-log`) и адрес веб-интерфейса управления системой вида `https://localhost.localdomain:8080`.

После нажатия кнопки «Завершить» и перезагрузки компьютера можно загрузить установленную систему в обычном режиме.

Удачной работы с ALT Linux!

Часть II

Администрирование системы

Глава 1

Планирование жёсткого диска

Файловая система Linux

Операционные системы хранят данные на диске при помощи **файловых систем**. Классическая файловая система представляет данные в виде вложенных друг в друга **каталогов** (их ещё называют папками), в которых содержатся **файлы**[1]. Один из каталогов является «вершиной» файловой системы (а выражаясь технически — «корнем»[2]), в нём содержатся (или, если угодно, из него растут) все остальные каталоги и файлы.

Если жёсткий диск разбит на разделы, то на *каждом* разделе организуется отдельная файловая система с собственным корнем и структурой каталогов (ведь разделы полностью изолированы друг от друга).

В Linux корневой каталог называется весьма лаконично — «/». Полные имена (пути) всех остальных каталогов получаются из «/», к которому дописываются справа имена последовательно вложенных друг в друга каталогов. Имена каталогов в пути также разделяются символом «/» («слэш»). Например, запись `/home` обозначает каталог «`home`» в корневом каталоге («/»), а `/home/user` — каталог «`user`» в каталоге «`home`» (который, в свою очередь, в корневом каталоге)[3]. Пе-

[1]Файл — область данных, имеющая собственное имя.

[2]Такой каталог называют **корневым каталогом**, поскольку он служит корнем дерева файловой системе (в математическом смысле слов «дерево» и «корень»).

[3]Весьма похожий способ записи полного пути используется в системах DOS и Windows, с той разницей, что корневой каталог обозначается литерой устройства с последующим двоеточием, а в качестве разделителя используется символ «\» («обратный слэш»).

речисленные таким образом каталоги, завершающиеся именем файла составляют **полный путь** к файлу.

Относительный путь строится точно так же, как и полный — перечислением через «/» всех названий каталогов, встретившихся при движении к искомому каталогу или файлу. Между полным путём и относительным есть только одно существенное различие: относительный путь начинается *от текущего каталога*, в то время как полный путь всегда начинается *от корневого каталога*. Относительный путь любого файла или каталога в файловой системе может иметь любую конфигурацию: чтобы добраться до искомого файла можно двигаться как по направлению к корневому каталогу, так и от него. Linux различает полный и относительный пути очень просто: если имя объекта *начинается* на «/» — это полный путь, в любом другом случае — относительный.

Монтирование

Корневой каталог в Linux всегда только *один*, а все остальные каталоги в него вложены, т. е. для пользователя файловая система представляет собой единое целое[1]. В действительности, разные части файловой системы могут находиться на совершенно разных устройствах: разных разделах жёсткого диска, на разнообразных съёмных носителях (лазерных дисках, дискетах, флэш-картах), даже на других компьютерах (с доступом через сеть). Для того, чтобы организовать из всего перечисленного единое дерево файловой системы, используется процедура **монтирования**.

Монтирование — это подключение в один из каталогов целой файловой системы, находящейся где-то на другом устройстве. Эту операцию можно представить как «прививание» ветки к дереву. Для монтирования необходим пустой каталог — он называется **точкой монтирования**. Точкой монтирования может служить любой каталог, никаких ограничений на этот счёт в Linux нет. При помощи специальной команды (`mount`) мы объявляем, что в данном *каталоге* (пока пустом) нужно отображать файловую систему, доступную на таком-то *устройстве* или же по сети. После этой операции в каталоге (точке монтирования) появятся все те файлы и каталоги, которые находят-

[1] Это отличается от технологии, применяемой в Windows или Amiga, где для каждого устройства, на котором есть файловая система, используется свой корневой каталог, обозначенный литерой, например «`a`», «`c`», «`d`» и т. д.

ся на соответствующем устройстве. В результате пользователь может даже и не знать, на каком устройстве какие файлы располагаются.

Подключённую таким образом («смонтированную») файловую систему можно в любой момент отключить — **размонтировать** (для этого имеется специальная команда `umount`), после чего тот каталог, куда она была смонтирована, снова окажется пустым.

Для Linux самой важной является **корневая файловая система** (root filesystem). Именно к ней затем будут подключаться (монтироваться) все остальные файловые системы на других устройствах. Обратите внимание, что корневая файловая система тоже монтируется, но только не к другой файловой системе, а к «самой Linux», причём точкой монтирования служит «/» (корневой каталог). Поэтому при загрузке системы прежде всего монтируется корневая файловая система, а при останове она размонтируется (в последнюю очередь).

Пользователю обычно не требуется выполнять монтирование и размонтирование вручную: при загрузке системы будут смонтированы все устройства, на которых хранятся части файловой системы, а при останове (перед выключением) системы все они будут размонтированы. Файловые системы на съёмных носителях (лазерных дисках, дискетах и пр.) также монтируются и размонтируются автоматически — либо при подключении носителя, либо при обращении к соответствующему каталогу.

Стандартные каталоги

В корневом каталоге Linux-системы обычно находятся только подкаталоги со *стандартными* именами. Более того, не только имена, но и *тип данных*, которые могут попасть в тот или иной каталог, также регламентированы стандартом[1]. Этот стандарт довольно последовательно соблюдается во всех Linux-системах: так, в любой Linux вы всегда найдёте каталоги `/etc`, `/home`, `/usr/bin` и т. п. и сможете довольно точно предсказать, что именно в них находится.

[1]Этот стандарт называется **Filesystem Hierarchy Standard** («стандартная структура файловых систем»). Стандарт **FHS** регламентирует не только перечисленные каталоги, но и их подкаталоги, а иногда даже приводит список конкретных файлов, которые должны присутствовать в определённых каталогах. Краткое описание стандартной иерархии каталогов Linux можно почерпнуть из руководства `hier(7)`. Полный текст и последнюю редакцию стандарта **FHS** можно найти в пакете `fhs` или прочесть по адресу `http://www.pathname.com/fhs/`.

Стандартное размещение файлов позволяет и человеку, и даже программе предсказать, где находится тот или иной компонент системы. Для человека это означает, что он сможет быстро сориентироваться в любой системе Linux (где файловая система организована в соответствии со стандартом) и найти то, что ему нужно. Для программ стандартное расположение файлов — это возможность организации автоматического взаимодействия между разными компонентами системы.

Параметры монтирования

При выполнении операции монтирования, в том числе при выборе точки монтирования во время инсталляции Linux-системы, можно изменять свойства смонтированной файловой системы. Для этого нужно указать утилите `mount` один или несколько параметров. Существует ряд параметров монтирования, поддерживаемых всеми файловыми системами. Есть параметры, характерные для одной конкретной файловой системы. Подробно о параметрах монтирования можно прочитать в руководстве к утилите `mount` (`mount(8)`).

Типы файловых систем

Существует довольно много разных файловых систем, которые отличаются друг от друга внутренним устройством, однако пользователь везде найдёт привычную структуру из вложенных каталогов и файлов. Файловые системы различаются скоростью доступа, надёжностью хранения данных, степенью устойчивости при сбоях, некоторыми дополнительными возможностями. Современные операционные системы поддерживают по несколько типов файловых систем (помимо файловых систем, используемых для хранения данных на жёстком диске, также файловые системы CD и DVD и пр.). Хотя для каждой операционной системы обычно есть одна «традиционная» файловая система, которая предлагается по умолчанию, является универсальной и подходит абсолютному большинству пользователей.

Важное свойство файловых систем — поддержка журналирования. **Журналируемая файловая система** ведёт постоянный учёт всех операций записи на диск. Благодаря этому после сбоя электропитания файловая система *всегда* автоматически возвращается в рабочее состояние.

Существует несколько типов файловых систем, которые в полной мере поддерживают все возможности, необходимые для полноценной работы Linux (все необходимые типы и атрибуты файлов, в том числе права доступа).

`Ext2/3`

Этот тип файловой системы разработан специально для Linux и традиционно используется на большинстве Linux-систем. Фактически в названии «Ext2/3» объединены названия двух вариантов этой файловой системы. Ext3 отличается от Ext2 только поддержкой **журналирования**, в остальном они одинаковы и легко могут быть преобразованы одна в другую в любой момент без потери данных. Обычно предпочтителен вариант с журналированием (Ext3) в силу его большей надёжности. При высокой параллельной дисковой загрузке производительность Ext3 снижается, что выражается в снижении скорости операций с диском и повышении значения нагрузки на систему (Load Average).

`ReiserFS`

Файловая система этого типа похожа скорее на базу данных: внутри неё используется своя собственная система индексации и быстрого поиска данных, а представление в виде файлов и каталогов — только одна из возможностей использования такой файловой системы. Традиционно считается, что ReiserFS отлично подходит для хранения огромного числа маленьких файлов. Поддерживает журналирование.

`XFS`

Файловая система, наиболее подходящая для хранения очень больших файлов, в которых постоянно что-нибудь дописывается или изменяется. Поддерживает журналирование. Лишена недостатков Ext3 по производительности, но при её использовании выше риск потерять данные при сбоях питания (в том числе и по причине принудительного обнуления повреждённых блоков в целях безопасности; при этом метаданные файла обычно сохраняются и он выглядит как корректный). Рекомендуется использовать эту файловую систему с проверенным аппаратным обеспечением, подключенным к управляемому источнику бесперебойного питания (UPS).

`SWAPFS`

Этот тип файловой системы находится на особом положении — он используется для организации на диске **области подкачки** (swap). Область подкачки используется в Linux для организации виртуальной памяти: когда программам недостаточно имеющейся в наличии оперативной памяти, часть рабочей информации временно размещается на жёстком диске.

`JFS`

Разработана IBM для файловых серверов с высокой нагрузкой: при разработке особый упор делался на производительность и надёжность, что и было достигнуто. Поддерживает журналирование.

В Linux поддерживается, кроме собственных, немало форматов файловых систем, используемых другими ОС. Если способ записи на эти файловые системы *известен* и не слишком замысловат, то работает и запись, и чтение, в противном случае — только чтение (чего нередко бывает достаточно). Файловые системы перечисленных ниже типов обычно присутствуют на разделах диска, принадлежащих другим операционным системам.

`FAT12/FAT16/FAT32`

Эти файловые системы используются в MS-DOS и разных версиях Windows, а также на многих съёмных носителях (в частности, на дискетах и USB-flash). Linux поддерживает чтение и запись на эти файловые системы.

`NTFS`

Файловая система NTFS изначально появилась в системах Windows NT, но может использоваться и другими версиями Windows (например, Windows 2000). В Linux NTFS поддерживается только на чтение.

Именование дисков и разделов в Linux

Внимание

Файлы, соответствующие устройствам постоянного хранения информации, в том числе жёстким дискам, получают в Linux специальные наименования в зависимости от типа и способа подключения.

Файлы устройств

Многие устройства, в том числе жёсткие диски, лазерные приводы и разнообразные съёмные носители, представлены в системе Linux в виде *файлов* особого типа — файлов устройств (их ещё называют иногда файлами-дырками). Операции чтения и записи на диск система выполняет как чтение/запись файла, соответствующего данному устройству. Все файлы устройств располагаются в специально предназначенном для них каталоге: `/dev`. Полные имена файлов устройств, соответствующих дискам и разделам дисков, складываются из названия каталога `/dev/` и обозначения соответствующего диска или раздела. Например, первому основному разделу первого диска IDE в Linux соответствует файл `/dev/hda1`.

Имена файлов, соответствующих жёстким дискам и разделам, довольно часто встречаются в конфигурационных файлах и в интерфейсе некоторых программ (особенно утилит, предназначенных для администрирования системы).

Устройства IDE

На сегодняшний день один из наиболее распространённых способов подключения жёстких дисков и лазерных (CD/DVD) приводов для IBM-совместимых персональных компьютеров — шина IDE. В Linux первый жёсткий диск на шине IDE обычно называется `hda` (**h**ard **d**isk «**a**»). Второй диск получает имя `hdb`, третий — `hdc` и так далее. Лазерные накопители по имени никак не отличаются от жёстких дисков. Часто бывает, что жёсткий диск — первый в системе (`hda`), а лазерный накопитель — *третий* (`hdc`), второго же вовсе нет. Обычно в персональном компьютере присутствует два канала IDE, на каждом из которых можно разместить до двух дисков.

`hda`

Первый диск на первом канале IDE (Primary master);

`hdb`

Второй диск на первом канале IDE (Primary slave);

`hdc`

Первый диск на втором канале IDE (Secondary master);

`hdd`

Второй диск на втором канале IDE (Secondary slave).

Устройства SCSI/SATA

Другой распространённый способ подключения жёстких дисков — интерфейс SCSI (по-русски произносится как «ска́зи»). В Linux SCSI-диски нумеруются буквами латинского алфавита (так же, как и IDE-диски), в зависимости от порядкового номера диска на шине SCSI: первый SCSI-диск называется `sda` (**s**csi **d**isk «**a**»), второй `sdb` и т. д.

Диски SATA и съёмные USB-устройства (USB флэш-карты, цифровые камеры и т. п.) обычно распознаются системой как SCSI-диски и, соответственно, обозначаются также `sda`, `sdb` и т. д. Аналогично через эмуляцию SCSI в Linux могут работать записывающие лазерные приводы (CD- и DVD-RW), они также получают имена, соответствующие SCSI-дискам, даже если в действительности подключены к шине IDE.

Имена устройств в Linux никогда не дублируются, в том числе при эмуляции: если соответствующее имя (например, `sda`) уже занято каким-то устройством, для вновь подключаемого устройства будет выбрано следующее (первое свободное) имя (например, `sdb`).

Нумерация разделов

Каждый **раздел** на жёстком диске также получает собственное обозначение в Linux. Обозначение раздела складывается из названия соответствующего диска и *номера* этого раздела на диске. Например, первый раздел на первом жёстком диске IDE обозначается `hda1`.

В Linux принята следующая схема нумерации разделов: **основные разделы**, которых на диске может быть не более 4-х (см. Структура жёсткого диска [стр. 33]), получают номера от `1` до `4` соответственно. Если основных разделов на диске меньше четырёх, то и номера отсутствующих разделов остаются незанятыми.

Номера, начиная с `5` получают **дополнительные** разделы, вложенные в **расширенный**. Так, номер `5` получает дополнительный раздел в первом расширенном, далее нумерация идет подряд — *вложенные* расширенные разделы не нумеруются.

Планирование диска

На жёстком диске любого компьютера хранятся данные, которые используются совершенно по-разному. Одни составляют операционную систему или нужны ей для работы, другие нужны пользователю, он их создаёт сам или откуда-то получает. Некоторые данные нужны временно, например, только на время работы программы, другие предназначены для «вечного» хранения. Есть такие данные, которые может изменить только человек, и такие, которые система сама создаёт или модифицирует в процессе работы. Наконец, есть такие данные, которые могут храниться на одном компьютере, а использоваться на нескольких (например, по локальной сети), и такие, которые предназначены только для данного компьютера.

Надёжность хранения данных и эффективность доступа к ним возрастает, если *разделять* данные разных типов (различающиеся по характеру использования). Для этого всё доступное пространство на жёстком диске (или дисках) разделяется на независимые области, каждая из которых предназначена для данных определённого типа. Для организации таких областей хорошо подходит технология разделения диска на **разделы**.

Поскольку разделы не зависят друг от друга, изменение содержимого одного раздела никак не сказывается на других. Одна из выгод такого подхода: в случае физического сбоя повреждения данных будут локализованы внутри того раздела, где произошёл сбой, и не затронут других разделов. Разделы открывают также путь для оптимизации скорости доступа: скорость чтения и записи для большинства дисков выше в середине и ниже к концу и началу диска. В самой быстрой области можно расположить раздел с данными, для которых важна скорость доступа.

Разделение диска на разделы *необходимо* в том случае, если на одном физическом устройстве должны быть установлены несколько операционных систем. Каждой операционной системе потребуется выделить не менее одного раздела.

Необходимые разделы

Минимальное количество разделов, которые необходимы Linux для работы — два. Первый — для **корневой файловой системы**, второй — для **области подкачки**.

Область подкачки (swap space) — это пространство на диске, используемое системой для организации **виртуальной памяти**. Если какая-то область оперативной памяти долгое время не используется, её содержимое записывается на диск, в область подкачки — тем самым освобождается место в физической памяти для других процессов. Когда же эта область памяти потребуется вновь, ядро *подкачает* её с диска и разместит в оперативной памяти.

Благодаря этому вполне может сложиться ситуация, когда используется больше оперативной памяти, чем её есть в действительности: если не вся заказанная память используется одновременно, что позволяет системе откачивать некоторые области. В Linux принято размещать область подкачки на отдельном разделе, что позволяет увеличить скорость доступа к данным и уменьшить риск повреждения ценных данных на основных разделах.

Для обеспечения максимальной скорости доступа к данным области подкачки ее раздел рекомендуется размещать в начале либо в середине диска. Данные, находящиеся в swap, являются временными и не представляют ценности после перезагрузки компьютера. Поэтому размещение swap-раздела, как и других разделов с неуникальными данными (например, `/tmp` и `/usr` — о них расказано ниже), в начале диска предпочтительнее: снижается риск потери важных данных.

Начало диска более подвержено повреждениям в том числе и из-за человеческого фактора. Печально известным примером опечатки, которая ведет к уничтожению данных в начале диска, может послужить `dd of=/dev/hda` вместо `dd of=/dev/hda3`. Однако если в начале диска хранятся неуникальные данные, то этот процесс, будучи вовремя остановленным, не успеет добраться до действительно ценной информации в разделах `/etc`, `/home` или `/var`, в то время как восстановление таблицы разделов — задача несложная.

С корневой файловой системы начинает расти всё дерево файлов Linux. **Точкой монтирования** для корневой файловой системы служит «/» — корневой каталог. Можно поместить все данные (включая пользовательские файлы) на один раздел — это как раз тот случай, когда для Linux потребуется всего два раздела. Для повышения эффективности и надёжности некоторые ветви дерева файлов можно выносить на другие разделы. Поскольку для файловой системы Linux не важно, каким образом части дерева каталогов расположены на разделах дисков, у вас есть возможность использовать каждый из имеющихся разделов диска под любой каталог файловой системы. В этом случае раздел с корневым каталогом будет служить точкой монтирования для остальных файловых систем.

Смонтированной файловой системе важно указать правильные **параметры монтирования**. Задавая разделам с разными типами данных походящие параметры, можно добиться значительного повышения производительности и безопаснсти. Ниже перечислены наиболее часто используемые общие параметры монтирования.

`noatime`

При каждом доступе к файлу, в том числе при чтении, обновляется время последнего доступа к нему. При использлвании этого параметра это обновление производиться не будет, что может быть полезно для ускорения работы (особенно актуально для серверов).

`nodev`

Этот параметр не позволяет создавать на файловой системе файлы-устройства. Если точно известно, что на данной файовой системе файлы-устройства не нужны, можно использовать этот параметр для повышения безопасности.

`nosuid`

Параметр запрещает исполнение SUID-программ.

`noexec`

Запрещает запуск исполняемых программ из файлов на данной файловой системе.

`ro`

Обеспечивает доступ к файловой системе только для чтения.

Дополнительно выделяемые разделы

`/home`

Домашние каталоги пользователей. Здесь хранятся персональные каталоги всех пользователей машины. Размер каталога зависит от количества работающих пользователей и от их потребностей. Рекомендуемые файловые системы — Ext3 или XFS. Параметры — `noexec` (в случае невозможности применения — `nosuid`).

`/usr`

Статические данные: большая часть пакетов устанавливает свои исполняемые файлы и данные в каталог `/usr`. Преимуществом размещения этого каталога в отдельном разделе является то, что при нормальной работе (кроме установки/удаления пакетов) не требуется в него записывать никаких данных, поэтому этот раздел можно монтировать в режиме «только чтение», в том числе по локальной сети. В этом случае несколько машин могут пользоваться одним сетевым разделом `/usr`. Размер этого раздела зависит от количества пакетов, которые будут установлены, он колеблется в пределах от 500 Мб для маленькой установки до нескольких гигабайт для полной установки. Вариант на 2— 3 Гб (в зависимости от размера диска) скорее всего подойдёт.

`/var`

Переменные данные, которые создаются системой в процессе работы. Запись в этот каталог осуществляется весьма часто, а количество данных в нём имеет тенденцию расти (здесь расположены все **системные журналы**). Требования к объёму очень сильно зависят от профиля машины. На пользовательских домашних станциях будет вполне достаточно нескольких сотен мегабайт. Для серверов эта цифра будет существенно больше, поэтому для повышения производительности и надежности хранения информации переменные данные разных типов рекомендуеся располагать на разных разделах. Файловая система этого раздела должна поддерживать журналирование (Ext3). При монтировании желательно задать параметры `noexec` и `nosuid`.

`/var/log`

При установке на сервер лучшим решением будет вынести системные журналы на отдельный раздел. При сбоях или внешних атаках размер журналов может резко увеличиваться, что, в случае их хранения вместе с другими переменными системными данными, чревато сбоем в работе системы. Также, если сервер используется для выполнения узкого круга задач (например, как web-сервер), рекомендуется выносить на отдельный раздел журналы основной системной службы (например, `/var/log/apache`). Оптимальная файловая система — Ext3, параметры — `noatime`, `noexec`, `nosuid`.

`/var/tmp`

Может быть полезным создать отдельную файловую систему для временных данных, которые нежелательно потерять в случае программного или аппаратного сбоя. Этот раздел должен обеспечивать высокую надежность хранения данных, поэтому оптимально создать в нем файловую систему с поддержкой журнализации (Ext3), указав параметры `noexec` и `nodev`.

`/var/spool/mail`

Если на сервере хранится почта пользователей, каталог с ней необходимо выделить в отдельный раздел. Также обязательно устана021ливать ограничения на использование дискового пространства для отдельных пользователей, чтобы избежать неожиданного переполнения раздела и проблем с работоспособностью сервера.

`/var/www`

Раздел для сайтов пользователей.

`/tmp`

Этот каталог предназначен для **временных файлов**: в таких файлах программы хранят промежуточные данные, необходимые для работы. После завершения работы программы временные файлы теряют смысл и должны быть удалены. Обычно каталог `/tmp` очищается при каждой загрузке системы. Поскольку запись в этот каталог осуществляется очень часто, а требования к надёжности очень низкие, то есть большой смысл выделить `/tmp` в отдельный раздел. В противном случае он окажется частью раздела «/», требования к которому по записи и надёжности прямо противоположные (см. ниже). Размер раздела `/tmp` в обычном случае должен быть примерно равен размеру `swap`. В последнее время раздел `/tmp` зачастую размещают в виртуальной файловой системе `tmpfs` непосредственно в оперативной памяти.

`/`

Корневой раздел — это самый важный раздел. Он не только содержит наиболее важные данные и программы системы, но будет также служить точкой монтирования для других разделов. Если `/usr`, `/var` и `/home` вынесены на отдельные разделы, то потребность в объёме корневого раздела небольшая, обычно достаточно 500 Мб. Требования: корневой раздел должен быть доступен в процессе загрузки, в процессе работы доступ на запись в этот раздел требуется нечасто, но весьма важна надёжность.

`/boot`

Небольшой раздел (5— 10 Мб), на котром хранятся исполняемые и failsafe ядра и данные используемого загрузчика. Обычно располагается в самом начале жесткого диска и всегда является первичным разделом (в отличие от логических томов в случае использования LVM). Оптимальная файловая система — Ext2, поскольку запись в раздел производится редко, а его объем мал.

Выделение вышеперечисленных разделов направлено прежде всего на повышение эффективности работы сервера. Для домашних рабочих станций чаще всего вполне достаточно, помимо необходимых разделов, выделить всго один — для хранения пользовательских данных (`/home`). Увеличения гибкости управления разделами, особенно при большом их количестве, можно добиться использованием технологии LVM, которая позволяет создавать, удалять и изменять размер логических устройств без риска потери данных.

Структура жёсткого диска

Сектора

Любой жёсткий диск можно представить как огромный «чистый лист», на который можно записывать данные и откуда потом их можно считать. Чтобы ориентироваться на диске, всё его пространство разбивают на небольшие «клеточки» — **сектора**. Сектор — это минимальная единица хранения данных на диске, обычно его размер составляет 512 байт. Все сектора на диске нумеруются: каждый из n секторов получает номер от 0 до n— 1. Благодаря этому любая информация, записанная на диск, получает точный адрес — номера соответствующих секторов. Так что диск ещё можно представить как очень длинную строчку (ленточку) из секторов. Можете посчитать, сколько секторов на вашем диске размером в N гигабайт.

Разделы

Представлять жёсткий диск как единый «лист» не всегда бывает удобно: иногда полезно «разрезать» его на несколько независимых листов, на каждом из которых можно писать и стирать что угодно, не опасаясь повредить написанное на других листах. Логичнее всего записывать раздельно данные большей и меньшей важности или просто относящиеся к разным вещам.

Конечно, над жёстким диском следует производить не физическое, а логическое разрезание, для этого вводится понятие **раздел** (partition). Вся последовательность (очень длинная ленточка) секторов разрезается на несколько частей, каждая часть становится отдельным разделом. Фактически, нам не придётся ничего разрезать (да и вряд ли бы это удалось), достаточно объявить, после каких секторов на диске находятся границы разделов.

Таблица разделов

Технически разбиение диска на разделы организовано следующим образом: заранее определённая часть диска отводится под **таблицу разделов**, в которой и написано, как разбит диск. Стандартная таблица разделов для диска IBM-совместимого компьютера — HDPT (**H**ard **D**isk **P**artition **T**able) — располагается в конце самого первого сектора диска, после **предзагрузчика** (**M**aster **B**oot **R**ecord, MBR) и состоит из четырёх записей вида «*тип начало конец*», по одной на каждый раздел. *Начало* и *конец* — это номера тех секторов диска, где начинается и заканчивается раздел. С помощью такой таблицы диск можно поделить на четыре или меньше разделов: если раздела нет, *тип* устанавливается в 0.

Однако четырёх разделов редко когда бывает достаточно. Куда же помещать дополнительные поля таблицы разбиения? Создатели IBM PC предложили универсальный способ: один из четырёх основных разделов объявляется **расширенным** (extended partition); он, как правило, является последним и занимает *всё* оставшееся пространство диска.

Расширенный раздел можно разбить на подразделы тем же способом, что и весь диск: в самом начале — на этот раз не диска, а самого *раздела* — заводится **таблица разделов**, с записями для четырёх разделов, которые снова можно использовать, причём один из подразделов может быть, опять-таки, расширенным, со своими подразделами и т. д.

Разделы, упомянутые в таблице разделов *диска*, принято называть **основными** (primary partition), а все подразделы расширенных разделов — **дополнительными** (secondary partition). Так что основных разделов может быть не более четырёх, а дополнительных — сколько угодно.

Чтобы не усложнять эту схему, при разметке диска соблюдают два правила: во-первых, расширенных разделов в таблице разбиения *диска* может быть не более одного, а во-вторых, таблица разбиения *расширенного раздела* может содержать либо одну запись — описание дополнительного раздела, либо две — описание дополнительного раздела и описание вложенного расширенного раздела.

Тип раздела

В таблице разделов для каждого раздела указывается **тип**, который определяет **файловую систему**, которая будет содержаться в этом разделе. Каждая операционная система распознаёт определённые типы и не распознаёт другие, и, соответственно, откажется работать с разделом неизвестного типа.

Внимание

Следует всегда следить за тем, чтобы тип раздела, установленный в таблице разделов, правильно указывал тип файловой системы, фактически содержащейся внутри раздела. На сведения, указанные в таблице разделов, может полагаться не только ядро операционной системы, но и любые утилиты, чьё поведение в случае неверно указанного типа может быть непредсказуемым и повредить данные на диске.

Подробнее о файловых системах см. раздел Типы файловых систем `[стр. 23]`.

Логические тома (LVM)

Работая с разделами, нужно учитывать, что производимые над ними действия связаны непосредственно с разметкой жёсткого диска. С одной стороны, разбиение на разделы — это наиболее традиционный для PC способ логической организации дискового пространства. Однако если в процессе работы появится потребность изменить логику разбиения диска или размеры областей (т. е. когда возникает задача **масштабирования**), работа с разделами не очень эффективна.

Например, при необходимости создать новый раздел или увеличить размер существующего, можно столкнуться с рядом трудностей, связанных с ограничением количества дополнительных разделов или перераспределением данных. Избежать их очень просто: нужно лишь отказаться от «привязки» данных к определённой области жёсткого диска. В Linux эта возможность реализуется при помощи **менеджера логических томов** (LVM — **L**ogical **V**olume **M**anager). LVM организует дополнительный уровень абстракции между *разделами* с одной стороны и хранящимися на них *данными* с другой, выстраивая собственную иерархическую структуру.

Дисковые разделы (в терминологии LVM — **физические тома** объединяются в **группу томов**, внутри которой создаются **логические тома**. Таким образом, группа томов выстраивает соответствие между физическим и логическим пространством диска.

Технологически это организуется следующим образом. Физические тома разбиваются на отдельные блоки — **физические экстенты**, которые объединяются в **группу томов**. Логические тома разбиваются на блоки такого же размера — **логические экстенты**. В разных группах томов размер экстента может быть различным.

Отношения между логическими и физическими томами представлены в виде **отображения** логических экстентов в физические. Возможны два способа отображения — **линейное** и **расслоенное** (striped). В первом случае логические экстенты располагаются последовательно соответственно физическим, во втором поочередно распределяются между несколькими физическими томами.

В свою очередь, между логическим томом и группой томов возникают отношения, аналогичные таковым между разделом и жёстким диском, с отличием в уровне абстракции и, соответственно, колоссальной разнице в гибкости манипуляции. Поскольку раздел — конкретная область физического диска между двумя определёнными секторами, а том — логическая категория, принимаемая для удобства использования дискового пространства, производить манипуляции со вторым значительно проще. Можно свободно перераспределять логические тома внутри группы, изменять их размер, увеличивать размер группы томов за счёт внесения в неё нового раздела (только при линейном отображении) и многое другое.

Дисковые массивы (RAID)

Иногда обычной производительности жёсткого диска может не хватать. В случаях, когда во главу угла ставится скорость работы с данными (скорость записи и чтения) или надёжность их хранения, используется технология **RAID** (Redundant array of independent disks — избыточный массив независимых дисков). Технология RAID позволяет объединять несколько физических дисковых устройств (жёстких дисков или разделов на них) в **дисковый массив**. Диски, входящие в массив, управляются централизованно и представлены в системе как одно логическое устройство, подходящее для организации на нем единой файловой системы.

Существует два способа реализации RAID: аппаратный и программный. Аппаратный дисковый массив состоит из нескольких жёстких дисков, управляемых при помощи специальной платы контроллера RAID-массива. **Программный RAID** в Linux-системах (Linux Software RAID) реализуется при помощи специального драйвера (**M**ultiple **D**evice driver — драйвер MD-устройства). В программный массив организуются дисковые разделы, которые могут занимать как весь диск, так и его часть, а управление осуществляется посредством специальных утилит (`mdadm`).

Программные RAID-массивы, как правило, менее надежны, чем аппаратные, но обеспечивают более высокую скорость работы с данными (производительность процессора и системной шины обычно намного выше, чем у любого дискового контроллера). Также их преимущество по сравнению с аппаратными массивами: независимость от форматов данных на диске и как следствие — большая совместимость с различными типами и размерами дисков и их разделов. Использование программного RAID также позволяет сэкономить на покупке дополнительного оборудования. Однако обратной стороной медали станет увеличение нагрузки на процессор и системную шину, это следует иметь в виду, принимая решение об использовании программного RAID.

Уровни RAID

Существует несколько разновидностей RAID-массивов, так называемых **уровней**. В Linux поддерживаются следующие уровни программных RAID-массивов.

RAID0

Для создания массива этого уровня понадобится как минимум два диска одинакового размера. Запись осуществляется по принципу **чередования**: данные делятся на **чанки** (chunk) — порции данных одинакового размера, и поочерёдно распределяются по всем дискам, входящим в массив. Поскольку запись ведётся на все диски, при отказе одного из них будут утрачены *все* хранившиеся на массиве данные. Это цена выбора в пользу увеличения скорости работы с данными:

запись и чтение на разных дисках происходит параллельно и, соответственно, быстрее.

RAID1

Массивы этого уровня построены по принципу **зеркалирования**, при котором все данные, записанные на одном диске, дублируются на другом. Для создания такого массива потребуется два или более дисков одинакового размера. **Избыточность** обеспечивает отказоустойчивость массива: в случае выхода из строя одного из дисков, данные на другом остаются неповреждёнными. Расплата за надёжность — фактическое сокращение дискового пространства вдвое. Скорость чтения и записи остается на уровне обычного жесткого диска.

RAID4

В массивах RAID4 реализован принцип **чётности**, объединяющий технологии чередования и зеркалирования. Один из трёх (или из большего числа) дисков задействуется для хранения информации о чётности в виде суперблоков с контрольными суммами блоков данных, последовательно распределённых на остальных дисках (как в RAID0). Достоинства этого уровня — отказоустойчивость уровня RAID1 при меньшей избыточности (из скольких бы дисков не состоял массив, под контрольную информацию задействуется лишь один из них). При отказе одного из дисков утраченные данные можно будет восстановить из контрольных суперблоков, причем, если в составе массива есть резервный диск, реконстукция данных начнется автоматически. Очевидным недостатком, однако, является снижение скорости записи, поскольку информацию о чётности приходится высчитывать при каждой новой записи на диск.

RAID5

Этот уровень аналогичен RAID4, за тем исключением, что суперблоки с информацией о чётности располагаются не на отдельном диске, а равномерно распределяются по всем дискам массива вместе с блоками данных. Как результат — повышение скорости работы с данными и высокая отказоустойчивость.

Массивы всех уровней помимо блоков данных и суперблоков с контрольными суммами могут также содержать специальный **суперблок** (persistent superblock), который располагается в начале всех дисков массива и содержит информацию о конфигурации MD-устройства. Наличие отдельного суперблока позволяет ядру операционной системы получать информацию о конфигурации устройства RAID прямо с дисков, а не из конфигурационного файла, что может быть полезным, если файл по каким-то причинам перестанет быть доступным. Кроме того, наличие отдельного суперблока — необходимое условие автоопределения RAID-устройств при загрузке системы.

Более подробная информация о RAID

Более подробную информацию можно найти в документации и статьях, посвящённых RAID:

- `mdadm(8)`
- `http://opennet.ru/docs/HOWTO/Software-RAID-HOWTO.html`

(русский перевод: `http://www.opennet.ru/docs/HOWTO-RU/Software-RAID-HOWTO.html`)

- `http://freesource.info/wiki/HCL/XranenieDannyx/SoftwareRAID`
- `http://ferra.ru/online/storage/26107/`
- `http://citforum.ru/operating_systems/linux/raid_linux/`
- `http://nber.org/sys-admin/linux-nas-raid.html`
- `http://pythian.com/blogs/411/aligning-asm-disks-on-linux`
- `http://linux-ata.org/faq-sata-raid.html`

Разбиение диска средствами программы установки

Для администратора Linux важным моментом как при установке системы, так и для поддержания ее работоспособности, является планирование и организация дискового пространства. Программа-установщик, кроме стандартных средств, поддерживает технологии, повшающие гибкость работы с жестким диском. Они могут пригодиться в случае, если обычной установки Linux недстаточно, и требуются более сложные решения. Ниже описаны возможности, предоставляемые программой установки, а также изложены принципы их использования.

Рекомендации по разбиению диска

Доступное Linux дисковое пространство, как правило, разбивается на несколько логических областей, или **томов** в терминологии программы установки. **Том** — это дополнительный уровень между разделом и файловой системой, который создается для унифицированного представления в операционной системе различных типов устройств (аналогичен логическому тому LVM).

Разбивка на тома может быть организована с помощью разных технологий: самое простое — создавать тома, привязанные непосредственно к физическим дискам или областям дисков: т. е. занимать под том целиком жёсткий диск или раздел жёсткого диска. При использовании одного из стандартных профилей разбиения диска применяется именно эта схема: создаётся несколько разделов на свободном месте жёстких дисков.

Программа установки позволяет создавать на диске и более сложную разметку с использованием технологий LVM и Linux Software RAID. Технология LVM предоставляет возможность более гибко распределять логические тома по физическим устройствам. Интерфейс управления логическими томами доступен при выборе пункта `LVM` в дереве устройств.

Помимо этого, программа установки позволяет устанавливать ALT Linux на поддерживаемые аппаратные и программные RAID-массивы (в том числе создавать программные RAID уровней 0, 1, 4/5). Интерфейс для создания RAID доступен при выборе в дереве устройств пункта `RAID`.

Перед размещением данных на логическом томе в нём должна быть создана **файловая система** (т. е. произведено форматирование раздела). Далее каждому тому (точнее, файловой системе в нём) должна быть назначена **точка монтирования**, т. е. тот фрагмент единой файловой системы Linux, который следует разместить на этом томе.

Работа с диском

В дереве устройств представлены доступные жёсткие диски и разделы на них (в том числе здесь могут оказаться съёмные USB-носители, подключённые к компьютеру в момент установки), а также в дерево включены отдельные ветки для управления/отображения устройств LVM и RAID. Узнать, каким устройствам вашего компьютера соответствуют названия в списке, можно в разделе Именование дисков и разделов в Linux [стр. 25].

Если на жестком диске присутствует таблица разделов, в ветке дерева, начинающейся от этого диска, будет отображено текущее расположение разделов, кроме случаев, когда раздел входит в состав устройств LVM или RAID — такие разделы в составе диска не о отображаются. Для каждого раздела указаны его размер и тип файловой системы (в колонке «Файловая система»). Возможно удалить существующую таблицу разделов диска. Для этого понадобится поочередно удалить с него все разделы, после чего, выбрав диск в дереве устройств, нажать «Удалить таблицу разделов».

Для каждого вновь создаваемого раздела предлагается выполнить стандартную последовательность операций: от создания раздела до назначения точки монтирования.

- Создать раздел
- Создать том
- Создать файловую систему
- Назначить точку монтирования

Для **создания нового раздела** выберите свободное место на диске (выбрав в списке значок диска или свободную область `<unused>` на нем) и нажмите «Создать раздел». Если свободного места нет и оно не было освобождено заранее, это нужно сделать сейчас, удалив один или несколько из существующих разделов или, если есть возможность, уменьшив их размер.

При создании раздела прежде всего нужно указать его размер и определить его расположение на диске. Для этого используются регуляторы «Размер» и «Смещение». Можно **изменить размер** уже созданного раздела, для этого выберите раздел и нажмите кнопку «Увеличить» или «Уменьшить». При увеличении раздела пределом служит свободное место на диске, а при уменьшении — объём, фактически *занятый данными* на этом разделе.

Том с файловой системой, как правило, создается в разделе диска, однако может быть создан непосредственно на жестком диске, в случае если на нем ещё нет таблицы разделов (эту возмжность следует использовать с осторожностью, поскольку есть риск в будущем принять такой диск за неформатированный и потерять данные на нем).

Для **создания тома** выделите нужный диск или раздел и нажмите «Создать том». Поскольку единственным параметром тома является тип создаваемой в нем файловой системы, вам будет сразу предложено выбрать в появившемся списке ее тип и перейти к следующей операции — назначению точки монтирования.

Можно отложить операцию **создания файловой системы** (сняв метку с пункта перехода к следующей операции), например, для того, чтобы изменить размер только что созданного тома. Вместе с размером тома изменится и размер раздела, в котором создан изменяемый том. Изменять размер только что созданного тома с файловой системой нельзя. Для изменения размера такого тома файловую систему с него необходимо предварительно удалить. Для создания файловой системы нажмите «Создать файловую систему».

Для тома с файловой системой могут быть доступны дополнительные настройки: проверка тома на наличие ошибок (сбойных участков) и присвоение ему метки тома. Для файловой системы Ext2/3 можно выбрать, использовать ли функцию поддержки журналирования.

Созданной файловой системе возможно сразу присвоить точку монтирования. причем наиболее подходящий вариант будет предложен по умолчанию. Есть возможность выбрать из списка наиболее часто используемых вариантов или вписать нужный самостоятельно. Выбор точки монтирования для файловой системы на уже существующем разделе осуществляется нажатием кнопки «Изменить точку монтирования».

Работа с LVM

Для создания группы томов и логических томов LVM необходимым условием является наличие на диске как минимум одного пустого раздела, т.е. такого, на котором нет тома с файловой системой. Необходимо создать такой раздел, не забыв правильно указать его тип — Linux LVM.

Выбрав в списке устройств LVM, нажмите кнопку «Создать группу томов». Сразу появится окно создания группы томов, в котором нужно определить основные параметры — дать новой группе имя и выбрать размер экстента.

Внутри группы томов создаются логические тома. Их может быть сколь угодно много в зависимости от требований пользователя. Как и при работе с разделами, можно сразу перейти к созданию логического тома, отметив пункт «Создать том». Если вы не хотите создавать том, например, если вы решили изменить размер группы томов, снимите выделение с этого пункта. Будет создана пустая группа томов, к созданию тома внутри неё можно вернуться, выделив ее и нажав «Создать том».

Каждому созданному в группе томов логическому тому нужно дать название и указать его размер. Имя тома может быть любым и, например, указывать на тип хранящихся на томе данных. Отметьте разделы для размещения каждого тома и способ распределения данных по разделам. Выбор линейного или распределенного отображения логических экстентов в физические осуществляется при помощи движка «Число расслоений (stripes)». Перемещая его, можно изменять числовое значение от единицы (обозначающей линейное отображение) до числа, соответствующего количеству физических томов, по которым будут распределяться данные создаваемого логического тома. В новом томе нужно создать файловую систему.

В результате в дереве устройств LVM появится созданная группа томов с вложенными логическими томами. Одновременно с этим разделы, вошедшие в группу томов, перестанут отображаться среди разделов диска.

Над логическими томами LVM можно производить те же операции, что и над разделами с томами: изменять их размер или удалять. Для удаления группы томов необходмо сначала удалить все входящие в нее логические тома, в противном случае в качестве отказа выполнить операцию появится сообщение «Device or resourse busy».

Работа с RAID

Для RAID, так же как для LVM, необходим пустой раздел. Будьте внимательны: для того, чтобы при старте системы RAID-массив определялся корректно, необходимо указать тип раздела Linux RAID. Создав раздел нужного типа, в таблице устройств нужно выбрать RAID и нажать «Создать RAID».

В открывшемся списке выберите уровень RAID-массива, который вы хотите создать. Ниже перечислены уровни RAID, которые позволяет создавать программа установки.

`RAID 0`

Для массива этого уровня нужно определить два параметра: определить размер чанка (минимум 4 кб, 32 кб по умолчанию) и выбрать, нужно ли создавать в нем отдельный суперблок.

`RAID 1`

Для этого уровня кроме вышеперечисленных параметров можно определить количество резервных дисков. Есть пункт «Деградированный массив», выбрав который, можно создавать массив с неполным набором дисков. Это может быть полезно, если вы решили создать массив, но еще не установили второй диск.

`RAID 4/5`

Для создания массивов этих уровней определяются те же параметры, что и для устройств уровня RAID1: размер чанка, наличие отдельного суперблока и поддержка возможности создания неполного (деградированного) массива. Также можно выбрать, какие диски или разделы войдут в массив, а какие будут использоваться в качестве резервных. Поскольку в массивах RAID4/5 используется четность, помимо перечисленных выше параметров можно выбрать алгоритм проверки четности, выбрав нужное значение из выпадающего меню рядом с соответствующим пунктом («Алгоритм RAID5».

После создания массива в нем создается *один том* с файловой системой. Эта операция аналогична созданию тома в разделе диска. Том занимает весь объем массива, в него входят все разделы или диски, входящие в массив. Размер тома не может быть изменен, пользователю доступны операции удаления устройства целиком или содержащейся на нем файловой системы, изменение точки монтирования. Для их выполнения служат соответствующие кнопки, отображающиеся на экране при выборе устройства RAID.

Глава 2

Загрузка системы

Настройка загрузки

Загрузка системы

Linux, установленный на жёстком диске, загружается при включении компьютера при помощи специальной программы — **загрузчика**. Программа-загрузчик исполняется при загрузке системы с жёсткого диска и загружает ядро ОС Linux, расположенное также на жёстком диске.

Загрузчики Linux можно также использовать для загрузки нескольких операционных систем, поскольку они позволяют выбирать при включении компьютера, какую систему нужно загрузить в этот раз. Если есть выбор из нескольких вариантов загрузки, то после некоторого времени ожидания будет загружена та система, которая выбрана по умолчанию: это не обязательно должен быть Linux, а может быть другая операционная система или специальный режим загрузки (например, восстановительный).

Например, при стандартной установке в начальном меню загрузчика ALT Linux доступны три альтернативы: «ALT Linux», «ALT Linux — Безопасные настройки» (загрузка с минимальным количеством драйверов, что может оказаться необходимым в случае неполадок), «Спасательная система». Если у вас есть установочный CD ALT Linux, вы также можете загрузиться с него: помимо установки новой системы можно загрузить уже установленный на жёстком диске Linux, который по тем или иным причинам невозможно загрузить прямо с жёсткого диска.

В нижней части экрана начального меню загрузчика располагается строка «Параметры». В этой строке можно указать параметры, которые будут переданы ядру Linux при загрузке.

Загрузка нескольких операционных систем

Прежде всего следует отметить, что ОС Linux может быть загружена с любого жёсткого диска системы и любого типа раздела — и **основного** (primary), и **дополнительного** (secondary), с различных типов файловых систем (например Ext2, Ext3, ReiserFS). При этом раздел, содержащий корневую файловую систему, не обязательно должен быть активным (иметь статус «`A`» в таблице разделов). Более того, вы можете использовать любой загрузчик, при условии, что он в состоянии передать управление на загрузочный сектор любого раздела (при этом несущественно, с какой операционной системой поставляется данный загрузчик). При наличии какого-либо стороннего загрузчика, загрузчик Linux следует устанавливать не в MBR первого жёсткого диска системы, а в загрузочный сектор корневого раздела Linux, на который впоследствии необходимо передать управление со стороны стороннего загрузчика. Подавляющее большинство UNIX-подобных систем не чувствительны к месту их размещения — главное, чтобы был способ передать управление на их программу начальной загрузки[1].

При использовании поставляемого с дистрибутивом загрузчика LILO передача управления на загрузочный сектор любого раздела, физически доступного в момент загрузки, не вызывает проблем. В то же время специфика архитектуры некоторых нестандартных операционных систем накладывает ряд ограничений на размещение этих систем на диске. Возможна, что такая система может загружаться только с активного *основного* (primary) раздела на первом жёстком диске системы, в противном случае возможны самые неожиданные проблемы с загрузкой. В такой ситуации лучше полностью сохранить статус загрузочного раздела этой операционной системы.

Настройка загрузчика

Первое решение, которое нужно принять — где расположить загрузчик. Программа установки предлагает на выбор несколько позиций, где может быть размещён загрузчик. Общее правило: если устройство указано как «полный» жёсткий диск (без указания но-

[1] Для очень старых BIOS действует правило 1024-го циллиндра: загрузка невозможна, если раздел, с которого загружается система, будет расположен на диске далее 1024-го циллиндра. В случае Linux — это раздел, содержащий корневую файловую систему или `/boot`, если он выделен на отдельный раздел.

мера раздела — например, `/dev/hda`), то загрузчик будет поставлен в MBR указанного диска; если устройство указано как раздел диска (в конце номер раздела), то загрузчик будет установлен в загрузочный сектор соответствующего раздела. Можно переместить загрузчик и после установки, исправив соответствующим образом конфигурационные файлы и дав команду `lilo` (см. ниже).

Если для загрузки всех операционных систем предполагается использовать загрузчик Linux (LILO), то в качестве загрузочного устройства необходимо выбрать первый диск системы; обычно это `/dev/hda` или `/dev/sda`. При таком выборе загрузчик первым получит управление от BIOS. Чтобы загрузчик Linux мог загружать другие операционные системы, ему нужно сообщить об их существовании. Программа установки ALT Linux умеет делать это автоматически. Однако если вам нужна более тонкая настройка, или что-то изменилось уже после установки ALT Linux, то можно отредактировать конфигурационные файлы загрузчика самостоятельно.

Это делается следующим образом: в файле `/etc/lilo.conf` для каждой операционной системы, которую потребуется загружать, нужно добавить новый раздел по аналогии со следующей записью:

```
other=/dev/hda1
label=other
table=/dev/hda
```

Данная запись сообщает LILO о том, что на раздел `/dev/hda1` установлена неизвестная ОС; в меню её надо отобразить под именем «other»; если пользователь выберет этот пункт меню — передать управление на загрузочный сектор `/dev/hda1`.

После сохранения данного файла конфигурации необходимо дать команду `lilo`, чтобы изменения вступили в силу.

Наоборот, если общим для всех ОС будет загрузчик другой операционной системы, то LILO необходимо установить на *корневой раздел* Linux (точка монтирования — `/`). После этого необходимо сообщить общему загрузчику всех ОС о том, как передавать управление на раздел Linux. Как это сделать — смотрите в документации к используемому вами программному обеспечению.

Восстановление загрузчика

Загрузка Linux может быть нарушена, если загрузчик Linux окажется по каким-то причинам повреждён или заменён другой программой. Последнее может произойти, например, в процессе установки другой ОС, если загрузчик был установлен в загрузочный сектор диска (MBR), содержание которого будет перезаписано и заменено загрузчиком другой ОС. В этой и подобных ситуациях необходимо восстановить загрузчик Linux, и возможно, изменить его размещение на диске.

Восстановить загрузчик в той же конфигурации, в которой он был до повреждения, несложно, для этого достаточно:

- любым способом загрузиться в Linux;
- смонтировать тот раздел жёсткого диска, на котором находится корневая файловая система Linux (выполнить `mount раздел /mnt`, где *раздел* — это имя соответствующего файла устройства, например, `/dev/hda1`);
- объявить раздел со смонтированной корневой файловой системой корневым (выполнить `chroot /mnt`);
- выполнить команду `lilo`.

В случае, если потребуется изменить конфигурацию загрузчика, например, переместить его на другой диск или раздел, перед выполнением `lilo` нужно будет соответствующим образом исправить конфигурационный файл `/etc/lilo.conf`.

Стартовые сценарии

Стартовые сценарии ALT Linux устроены примерно так же, как в RedHat Linux, по простой «.d»-схеме. Суть её в следующем. Любое атомарное действие по изменению способностей системы — запуск и останов определённой службы, автоматическое изменение или сохранение настроек и т. п. — оформляется в виде командного сценария в каталоге `/etc/rc.d/init.d`. К такому сценарию предъявляются определённые требования: как минимум, он должен распознавать в качестве первого параметра слова `start` и `stop`. Будучи запущен с параметром `start`, этот сценарий выполняет действия по запуску службы или активации настроек, а с параметром `stop` он останавливает службу или записывает активные настройки в файл.

Когда система загружается, процесс init — выполняет сначала сценарий `/etc/rc.d/rc.sysinit`, а затем — `/etc/rc.d/rcномер`, где *номер* соответствует **уровню выполнения** (профилю использования системы, обычно он равен 3, «многопользовательский с сетью» или 5, «многопользовательский с сетью и графической средой»). Всё это сводится к выполнению в **лексикографическом** (грубо говоря, алфавитном) порядке *всех* сценариев из каталога `/etc/rc.d/rcномер.d/`, начинающихся на букву «`K`» (Kill) с параметром `stop` и *всех* сценариев, начинающихся на «`S`» (Start) с параметром `start`. Принято после буквы «`K`» или «`S`» вставлять двузначное число, а затем — имя службы, а сами файлы в `rcномер.d/` делать символьными ссылками на сценарии в `/etc/rc.d/init.d/` (точнее, на файлы в `../init.d/`, так достигается совместимость с системами, в которых `init.d` и `rc*.d` лежат не в `/etc /rc.d`, а в `/etc` или вообще где угодно). Останов системы рассматривается как переход на уровень выполнения «`0`», а перезагрузка — на уровень выполнения «`6`», так что имена файлов в каталогах `rc0.d` и `rc6.d` начинаются сплошь на «`K`», кроме `S00killall`, `S01reboot` или `S0 1halt`. Таким образом достигается строгий порядок выполнения действий без создания единого стартового сценария (как это делалось, например, в ранних версиях BSD).

В ALT Linux эти рекомендации носят *обязательный* характер, потому что размещение сценариев в каталогах `rc*.d` происходит не вручную, а автоматически. Для этого в начало каждого сценария вносятся комментарии особого вида: «`# description:`», после которого следует краткое описание, и «`chkconfig:`», после которого следует три числа. Первое — набор цифр (без пробелов), соответствующих уровням выполнения, на которых сценарий *будет* использоваться (т. е. создаваться ссылка, начинающаяся на «`S`»). Второе — двузначное число, которое будет добавляться в имя ссылки после «`S`», а третье — двузначное число, которое будет добавляться в имя ссылки после «`K`».

```
#! /bin/sh
#
# crond          Start/Stop the cron clock daemon.
#
# chkconfig: 2345 40 60
# description: cron is a standard UNIX program that runs
user-specified \
#              programs at periodic scheduled times. vixie cron
adds a \
#              number of features to the basic UNIX cron,
including better \
#              security and more powerful configuration options.
# processname: crond
```

```
# config: /etc/crontab
# pidfile: /var/run/crond.pid
```

Команда `chkconfig имя_службы on` автоматически создаёт ссылки типа «`S`» для всех уровней выполнения, указанных в первом числе поля «`chkconfig:`», а для остальных уровней выполнения — ссылки типа «`K`». Команда `chkconfig имя_службы off` заводит во всех каталогах `rc*.d` ссылки типа «`K`». Стоит заметить, что другие поля похожего формата остались в стартовых сценариях от других, более сложных систем автоматической загрузки и в ALT Linux не используются. В приведённом примере сценарий `netfs` (монтирование сетевых файловых систем) запускается на уровнях 3, 4, и 5, причём при запуске служба имеет номер 25, а при останове — 75.

```
$ grep chkconfig: /etc/rc.d/init.d/netfs
# chkconfig: 345 25 75
$ chkconfig &#8212;list netfs
netfs           0:выкл  1:выкл  2:выкл  3:вкл   4:вкл   5:вкл
6:выкл
$ ls -l /etc/rc.d/rc*.d/*netfs*
lrwxrwxrwx  1 root root 15 Авг 15 19:06 /etc/rc.d/rc0.d/K75netfs
-> ../init.d/netfs
lrwxrwxrwx  1 root root 15 Авг 15 19:06 /etc/rc.d/rc1.d/K75netfs
-> ../init.d/netfs
lrwxrwxrwx  1 root root 15 Авг 15 19:06 /etc/rc.d/rc2.d/K75netfs
-> ../init.d/netfs
lrwxrwxrwx  1 root root 15 Авг 15 19:06 /etc/rc.d/rc3.d/S25netfs
-> ../init.d/netfs
lrwxrwxrwx  1 root root 15 Авг 15 19:06 /etc/rc.d/rc4.d/S25netfs
-> ../init.d/netfs
lrwxrwxrwx  1 root root 15 Авг 15 19:06 /etc/rc.d/rc5.d/S25netfs
-> ../init.d/netfs
lrwxrwxrwx  1 root root 15 Авг 15 19:06 /etc/rc.d/rc6.d/K75netfs
-> ../init.d/netfs
```

Кроме того, правила оформления стартовых сценариев ALT Linux рекомендуют, чтобы они распознавали ещё один параметр — `status`, позволяющий определить состояние службы (по крайней мере, узнать, запущена ли она или нет). Корректный стартовый сценарий всё равно должен уметь это определять, иначе попытка запуска «`K`»-сценария при *остановленной* службе будет выдавать сообщение об ошибке, даже если спецификой службы это предусмотрено. Например, супердемон xinetd при запуске проверяет, зарегистрирована ли в нём хотя бы одна сетевая служба, а не найдя таковых, завершает работу. Для отражения того, что запуск службы прошёл успешно, в ALT Linux заведён специальный каталог, `/var/lock/subsys`, в котором стартовые сценарии создают т. н. lock-файлы. Если lock-файл службы есть в этом каталоге, значит, она запущена, и «`K`»-сценарий действительно должен её останавливать. Поэтому рекомендуется запускать и останавливать системные службы не напрямую, запуском демонов, и даже не путём запуска сценария из `init.d`, а с помощью утилиты `service`. Впрочем, вызов `service имя_службы команда` эквивалентен `/etc/rc.d/init.d/имя_службы команда`, просто командная строка выходит короче.

При написании стартового сценария рекомендуется пользоваться файлом `/etc/init.d/functions`, в котором определены основные функции для работы с `/var/lock/subsys` и унифицированным выводом сообщений на экран. Обратите внимание, что по соображениям совместимости обычно используется имя каталога `/etc/init.d`, а не `/etc/rc.d/init.d`, при этом первое имя — всего лишь символьная ссылка на второе.

Глава 3

Система управления пакетами APT

Введение: пакеты, зависимости и репозитории

В современных системах на базе Linux огромное число общих ресурсов, которыми пользуются сразу несколько программ: разделяемых библиотек, содержащих стандартные функции, исполняемых файлов, сценариев и стандартных утилит и т. д. Удаление или изменение версии одного из составляющих систему компонентов может повлечь неработоспособность других, связанных с ним компонентов, или даже вывести из строя всю систему. В контексте системного администрирования проблемы такого рода называют нарушением **целостности системы**. Задача администратора — обеспечить наличие в системе согласованных версий всех необходимых программных компонентов (обеспечение целостности системы).

Для установки, удаления и обновления программ и поддержания целостности системы в Linux в первую очередь стали использоваться **менеджеры пакетов** (такие, как rpm в дистрибутивах RedHat или dpkg в Debian GNU/Linux). С точки зрения менеджера пакетов программное обеспечение представляет собой набор компонентов — программных **пакетов**. Такие компоненты содержат в себе набор исполняемых программ и вспомогательных файлов, необходимых для корректной работы программного обеспечения. Менеджеры пакетов облегчают установку программ: они позволяют проверить наличие необходимых для работы устанавливаемой программы компонент подходящей версии непосредственно в момент установки, а также производят необходимые процедуры для регистрации программы во всех операционных средах пользователя: сразу после установки программа может быть доступна пользователю из командной строки и — если это педусмотрено — появляется в меню всех графических оболочек.

Внимание

Благодаря менеджерам пакетов, пользователю Linux обычно не требуется непосредственно обращаться к установочным процедурам отдельных программ или непосредственно работать с каталогами, в которых установлены исполняемые файлы и компоненты программ (обычно это `/usr/bin`, `/usr/share/имя_пакета`) — всю работу делает менеджер пакетов. Поэтому установку, обновление и удаление программ в Linux обычно называют **управлением пакетами**.

Часто компоненты, используемые различными программами, выделяют в отдельные пакеты и помечают, что для работы ПО, предоставляемого пакетом A, необходимо установить пакет B. В таком случае говорят, что пакет A **зависит** от пакета B или что между пакетами A и B существует **зависимость**.

Отслеживание зависимостей между такими пакетами представляет собой серьёзную задачу для любого дистрибутива — некоторые компоненты могут быть взаимозаменяемыми: может обнаружиться несколько пакетов, предлагающих затребованный ресурс.

Задача контроля целостности и непротиворечивости установленного в системе ПО ещё сложнее. Представим, что некие программы A и B требуют наличия в системе компоненты C версии 1.0. Обновление версии пакета A, требующее обновления компоненты C до новой, использующей новый интерфейс доступа, версии (скажем, до версии 2.0), влечёт за собой обязательное обновление и программы B.

Однако менеджеры пакетов оказались неспособны предотвратить все возможные коллизии при установке или удалении программ, а тем более эффективно устранить нарушения целостности системы. Особенно сильно этот недостаток сказывается при обновлении систем из централизованного репозитория пакетов, в котором последние могут непрерывно обновляться, дробиться на более мелкие и т. п. Этот недостаток и стимулировал создание систем управления программными пакетами и поддержания целостности системы.

Для автоматизации этого процесса и применяется Усовершенствованная система управления программными пакетами APT (от англ. Advanced Packaging Tool). Такая автоматизация достигается созданием одного или нескольких внешних **репозиториев**, в которых хранятся пакеты программ и относительно которых производится сверка пакетов, установленных в системе. Репозитории могут содержать как официальную версию дистрибутива, обновляемую его разработчиками по мере выхода новых версий программ, так и локальные наработки, например, пакеты, разработанные внутри компании.

Таким образом, в распоряжении APT находятся две базы данных: одна описывает установленные в системе пакеты, вторая — внешний репозиторий. APT отслеживает целостность установленной системы и, в случае обнаружения противоречий в зависимостях пакетов, руководствуется сведениями о внешнем репозитории для разрешения конфликтов и поиска корректного пути их устранения.

Первоначально APT был разработан для управления установкой и удалением программ в дистрибутиве Debian GNU/Linux. При разработке ставилась задача создать систему управления пакетами с простым пользовательским интерфейсом, позволяющую производить установку, обновление и повседневные «хозяйственные» работы с установленными на машине программами без необходимости изучения тонкостей используемого в дистрибутиве менеджера программных пакетов.

Эти привлекательные возможности долгое время были доступны только пользователям Debian, поскольку в APT поддерживался только один менеджер пакетов, а именно применяемый в Debian менеджер пакетов `dpkg`, несовместимый с используемым в ALT Linux RPM. Эта несовместимость заключается прежде всего в различии используемых форматов данных (хотя существуют программы-конвертеры), но имеются и другие различия, обсуждение которых выходит за рамки изложения.

APT, однако, изначально проектировался как не зависящий от конкретного метода работы с установленными в системе пакетами, и эта особенность позволила разработчикам из бразильской компании Conectiva реализовать в нём поддержку менеджера пакетов RPM. Таким образом, пользователи основанных на RPM дистрибутивов (дистрибутивы ALT Linux входят в их число) получили возможность использовать этот мощный инструмент.

Система APT состоит из нескольких утилит. Чаще всего используется утилита управления пакетами `apt-get`: она автоматически определяет зависимости между пакетами и строго следит за их соблюдением при выполнении любой из следующих операций: установка, удаление или обновление пакетов.

Источники программ (репозитории)

Репозитории, с которыми работает APT, отличаются от обычного набора пакетов наличием метаинформации — индексов пакетов, содержащихся в репозитории, и сведений о них. Поэтому, чтобы получить всю информацию о репозитории, APT достаточно получить его индексы.

APT может работать с любым количеством репозиториев одновременно, формируя единую информационную базу обо всех содержащихся в них пакетах. При установке пакетов APT обращает внимание только на название пакета, его версию и зависимости, а расположение в том или ином репозитории не имеет значения. Если потребуется, APT в рамках одной операции установки группы пакетов может пользоваться несколькими репозиториями.

Внимание

Подключая одновременно несколько репозиториев, нужно следить за тем, чтобы они были совместимы друг с другом по пакетной базе, т. е. отражали один определённый этап разработки. Например, совместимыми являются основной репозиторий дистрибутива и репозиторий обновлений по безопасности *к данному дистрибутиву*. В то же время смешение среди источников APT репозиториев, относящихся к разным дистрибутивам, или смешение стабильного репозитория с нестабильной веткой разработки (Sisyphus) чревато различными неожиданными трудностями при обновлении пакетов.

APT позволяет взаимодействовать с репозиторием с помощью различных протоколов доступа. Наиболее популярные — HTTP и FTP, однако существуют и некоторые дополнительные методы.

Для того, чтобы APT мог использовать тот или иной репозиторий, информацию о нем необходимо поместить в файл `/etc/apt/sources.list`[1]. Описания репозиториев заносятся в этот файл в следующем виде:

```
rpm [подпись] метод:путь база название
rpm-src [подпись] метод:путь база название
```

[1] Если быть точным, этот файл может называться и иначе, другое имя файла со списком источников программ можно указать в конфигурационном файле `/etc/apt/apt.conf`. Подробнее о формате файла `apt.conf` можно узнать из руководства apt.conf(5).

`rpm` или `rpm-src`

Тип репозитория (скомпилированные программы или исходные тексты).

`[подпись]`

Необязательная строка-указатель на электронную подпись разработчиков. Наличие этого поля подразумевает, что каждый пакет из данного репозитория должен быть подписан соответствующей электронной подписью. Подписи описываются в файле `/etc/apt/vendor.list`.

`метод`

Способ доступа к репозиторию: ftp, http, file, rsh, ssh, cdrom, copy.

`путь`

Путь к репозиторию в терминах выбранного метода.

`база`

Относительный путь к базе данных репозитория.

`название`

Название репозитория.

Для добавления в `sources.list` репозитория на компакт-диске в APT даже предусмотрена специальная утилита — `apt-cdrom`. Чтобы добавить запись о репозитории на компакт-диске, достаточно вставить диск в привод и выполнить команду `apt-cdrom add`. После этого в `sources.list` появится запись о подключённом диске примерно такого вида:

```
rpm cdrom:[Junior Disk 1]/ ALTLinux main
rpm-src         cdrom:[Junior Disk 1]/ ALTLinux main
```

После того как отредактирован список репозиториев в `sources.list`, необходимо обновить локальную базу данных APT о доступных пакетах. Это делается командой `apt-get update`.

Если в `sources.list` присутствует репозиторий, содержимое которого может изменяться (как происходит с любым постоянно разрабатываемым репозиторием, в частности, обновлений по безопасности (updates), backports[1]) или Sisyphus[2], то прежде чем работать с APT,

[1] `http://backports.altlinux.ru`
[2] `http://sisyphus.ru`

необходимо синхронизировать локальную базу данных с удалённым сервером командой `apt-get update`. Локальная база данных создаётся заново каждый раз, когда в репозитории происходит изменение: добавление, удаление или переименование пакета. Для репозиториев, находящихся на компакт-дисках и подключённых командой `apt-cdrom add`, синхронизация производится единожды в момент подключения.

При выборе пакетов для установки, APT руководствуется *всеми* доступными репозиториями вне зависимости от способа доступа к ним. Так, если в репозитории, доступном по сети Интернет, обнаружена более новая версия программы, чем на компакт-диске, то APT начнёт загружать данный пакет из Интернет. Поэтому если подключение к Интернет отсутствует или ограничено низкой пропускной способностью канала или выской стоимостью, то следует закомментировать те строчки в `/etc/apt/sources.list`, в которых говорится о ресурсах, доступных по Интернет.

Репозитории ALT Linux

Все дистрибутивы ALT Linux выпускаются на основе репозитория Sisyphus команды ALT Linux Team[1]. Следует иметь в виду, что Sisyphus не является самостоятельным дистрибутивом, а отражает текущее состояние разработки и может содержать нестабильные версии пакетов. Периодически на базе этого проекта выпускаются отдельные оттестированные «срезы» — дистрибутивы.

В отличие от Sisyphus, ежедневно обновляемого разработчиками, такие срезы являются «замороженными» — разработка в них не ведётся, и сами срезы сохраняются в целях обеспечения целостности среды дистрибутива, в которой уже не должны обновляться версии пакетов. Единственное исключение делается для обновлений, исправляющих проблемы в безопасности системы, однако такие обновления помещаются в отдельном репозитории для каждого дистрибутива. Срезы Sisyphus и репозитории обновлений также являеются полноценными репозиториями APT.

Пользователи стабильных дистрибутивов не всегда готовы переходить на нестабильную ветку разработки и в то же время заинтересованы в обновлении версий некоторых прикладных программ (в которых появляется новая функциональность и т. п.). Для этих целей

[1] `http://altlinux.ru`

заинтересованными пользователями для каждого дистрибутива ALT Linux создаются и поддерживаются специальные репозитории с обновлёнными версиями программ — backportshttp://backport.altlinux.ru. Репозитории backports являются согласованными по пакетной базе с основным репозиторием соответствующего дистрибутива, и могут быть без опасений подключены параллельно с ним.

Непосредственно после установки дистрибутива ALT Linux в `/etc/apt/sources.list` обычно указывается несколько репозиториев:

- репозиторий обновлений в системе безопасности дистрибутива;
- полный срез репозитория Sisyphus, подмножеством которого является дистрибутив.

Поиск пакетов

Если вы не знаете точного названия пакета, для его поиска можно воспользоваться утилитой `apt-cache`, которая позволяет искать не только по имени пакета, но и по его описанию.

Команда `apt-cache search подстрока` позволяет найти все пакеты, в именах или описании которых присутствует указанная подстрока. Например:

```
$ apt-cache search ^ve-
ve-base - Basic appliance
ve-build-scripts - scripts used for VE building
ve-caching-nameserver - Caching name server
ve-ftp-server - FTP server
ve-imap-server - POP3/IMAP4 server
ve-kerberos-server - Kerberos server
ve-list-server - Mailing list server
ve-ntp-server - NTP server
ve-openvpn-server - virtual package for openvpn server appliance
ve-pptp-server - virtual package for pptp server appliance
ve-print-server - Print server
ve-proxy-server - Proxy server
ve-smtp-server - SMTP server
```

Обратите внимание, что в данном примере в поисковом выражении используется символ «^», указывающий на то, что необходимо найти совпадения только в начале строки (в данном случае — в начале имени пакета).

Для того, чтобы подробнее узнать о каждом из найденных пакетов и прочитать его описание, можно воспользоваться командой `apt-cache show`, которая покажет информацию о пакете из репозитория:

```
$ apt-cache show ve-ftp-server
Package: ve-ftp-server
Section: System/Base
Installed Size: 0
Maintainer: Stanislav Ievlev <inger@altlinux>
Version: 0.1-alt4
Pre-Depends: rpmlib(PayloadFilesHavePrefix) (<= 4.0-1),
rpmlib(CompressedFileNames) (<= 3.0.4-1)
Depends: apt, basesystem, sysklogd, etcnet, glibc-nss,
glibc-locales, netlist, anonftp, alterator-fbi, alterator-vsftpd,
alterator-users, openssh-server, passwd, less
Provides: ve-ftp-server (= 0.1-alt4)
Architecture: noarch
Size: 1989
MD5Sum: e7646f729b2bced59e4e759393cb1432
Filename: ve-ftp-server-0.1-alt4.noarch.rpm
Description: FTP server
 virtual package for ftp server appliance
```

`apt-cache` позволяет осуществлять поиск и по русскому слову, однако в этом случае будут найдены только те пакеты, у которых помимо английского есть ещё и описание на русском языке. К сожалению, русское описание на настоящий момент есть не у всех пакетов, хотя описания наиболее актуальных для пользователя пакетов переведены.

Установка или обновление пакета

Установка пакета с помощью APT выполняется командой

```
# apt-get install имя_пакета
```

`apt-get` позволяет устанавливать в систему пакеты, требующие для работы другие, пока ещё не установленные. В этом случае он определяет, какие пакеты необходимо установить, и устанавливает их, пользуясь всеми доступными репозиториями.

Установка пакета `ve-ftp-server` командой `apt-get install ve-ftp-server` приведёт к следующему диалогу с APT:

```
# apt-get install ve-ftp-server
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
```

```
  alterator-users alterator-vsftpd anonftp klogd netlist pwgen
shadow-groups sysklogd vsftpd xinetd
The following NEW packages will be installed:
  alterator-users alterator-vsftpd anonftp klogd netlist pwgen
shadow-groups sysklogd ve-ftp-server vsftpd xinetd
0 upgraded, 11 newly installed, 0 removed and 0 not upgraded.
Need to get 417kB of archives.
After unpacking 700kB of additional disk space will be used.
Do you want to continue? [Y/n]
. . .
Fetched 417kB in 0s (2176kB/s)
Committing changes...
Preparing...
################################################ [100%]
 1: xinetd
################################################ [  9%]
 2: vsftpd
################################################ [ 18%]
 3: alterator-vsftpd
################################################ [ 27%]
 4: netlist
################################################ [ 36%]
 5: pwgen
################################################ [ 45%]
 6: klogd
################################################ [ 54%]
 7: anonftp
################################################ [ 63%]
 8: shadow-groups
################################################ [ 72%]
 9: alterator-users
################################################ [ 81%]
Done.
```

Команда `apt-get install имя_пакета` используется и для обновления уже установленного пакета или группы пакетов. В этом случае `apt-get` дополнительно проверяет, не обновилась ли версия пакета в репозитории по сравнению с установленным в системе.

При помощи APT можно установить и отдельный бинарный rpm-пакет, не входящий ни в один из репозиториев (например, полученный из Интернет). Для этого достаточно выполнить команду `apt-get install путь_к_файлу.rpm`. При этом APT проведёт стандартную процедуру проверки зависимостей и конфликтов с уже установленными пакетами.

Иногда, в результате операций с пакетами без использования APT, целостность системы нарушается, и `apt-get` отказывается выполнять операции установки, удаления или обновления. В этом случае необходимо повторить операцию, задав опцию `-f`, заставляющую `apt-get` исправить нарушенные зависимости, удалить или заменить конфликтующие пакеты. В этом случае необходимо внимательно следить за сообщениями, выдаваемыми `apt-get`. Любые действия в этом режиме обязательно требуют подтверждения со стороны пользователя.

Удаление установленного пакета

Для удаления пакета используется команда `apt-get remove имя_пакета`. Для того, чтобы не нарушать целостность системы, будут удалены и все пакеты, зависящие от удаляемого: если отсутствует необходимый для работы приложения компонент (например, библиотека), то само приложение становится бесполезным. В случае удаления пакета, который относится к базовым компонентам системы, `apt-get` потребует дополнительного подтверждения производимой операции с целью предотвратить возможную случайную ошибку.

Если вы попробуете при помощи `apt-get` удалить базовый компонент системы, вы увидите такой запрос на подтверждение операции:

```
# apt-get remove filesystem
Обработка файловых зависимостей... Завершено
Чтение списков пакетов... Завершено
Построение дерева зависимостей... Завершено
Следующие пакеты будут УДАЛЕНЫ:
basesystem filesystem ppp sudo
Внимание: следующие базовые пакеты будут удалены:
В обычных условиях этого не должно было произойти, надеемся, вы
точно
представляете, чего требуете!
basesystem filesystem (по причине basesystem)
0 пакетов будет обновлено, 0 будет добавлено новых, 4 будет
удалено(заменено) и 0 не будет обновлено.
Необходимо получить 0B архивов. После распаковки 588kБ будет
освобождено.
Вы собираетесь совершить потенциально вредоносное действие
```

```
Для продолжения, наберите по-английски 'Yes, I understand this may
be
bad'
(Да, я понимаю, что это может быть плохо).
```

Каждую ситуацию, в которой APT выдаёт такое сообщение, необходимо рассматривать отдельно. Однако, вероятность того, что после выполнения этой команды система окажется неработоспособной, очень велика.

Обновление всех установленных пакетов

Для обновления всех установленных пакетов используется команда `apt-get upgrade`. Она позволяет обновить те и только те установленные пакеты, для которых в репозиториях, перечисленных в `/etc/apt/sources.list`, имеются новые версии; при этом из системы не будут удалены никакие другие пакеты. Этот способ полезен при работе со стабильными пакетами приложений, относительно которых известно, что они при смене версии изменяются несущественно.

Иногда, однако, происходит изменение в именовании пакетов или изменение их зависимостей. Такие ситуации не обрабатываются командой `apt-get upgrade`, в результате чего происходит нарушение целостности системы: появляются неудовлетворённые зависимости. Например, переименование пакета MySQL-shared, содержащего динамически загружаемые библиотеки для работы с СУБД MySQL, в libMySQL (отражающая общую тенденцию к наименованию библиотек в дистрибутиве) не приводит к тому, что установка обновлённой версии libMySQL требует удаления старой версии MySQL-shared. Для разрешения этой проблемы существует режим обновления в масштабе дистрибутива — `apt-get dist-upgrade`.

В случае обновления всего дистрибутива APT проведёт сравнение системы с репозиторием и удалит устаревшие пакеты, установит новые версии присутствующих в системе пакетов, а также отследит ситуации с переименованиями пакетов или изменения зависимостей между старыми и новыми версиями программ. Всё, что потребуется поставить (или удалить) дополнительно к уже имеющемуся в системе, будет указано в отчёте `apt-get`, которым APT предварит само обновление.

При работе с Sisyphus для обновления системы рекомендуется использовать команду `apt-get dist-upgrade`.

Глава 4

Виртуализация

Инфраструктура сервера: виртуальные контейнеры[1]

Внимание

В дистрибутивах ALT Linux администратору сервера рекомендуется размещать системные службы (сервисы) в изолированных окружениях, *виртуальных контейнерах*. В данном документе рассмотрены задачи, решению которых способствует виртуализация сервисов, а также некоторые технологические ограничения, которые следует принимать во внимание, планируя инфраструктуру сервера с использованием виртуальных контейнеров.

Виртуализация и администрирование

Серверы и сервисы

Под **сервером** обычно понимается отдельный (обычно мощный) компьютер, специализированный под выполнение задач по обслуживанию абонентов (предоставление **сервисов**). Количество таких сервисов, одновременно выполняемых в рамках одного сервера, ограничено сверху вычислительными возможностями оборудования, пропускной способностью канала для запросов клиентов и т. п. На практике количество сервисов на одном сервере обычно больше *одного*, как из экономических соображений (невыгодно позволять простаивать дорогому серверному оборудованию), так и из технологических (зачастую

[1]Огромное спасибо Петру Савельеву (`<peet@altlinux>`) за терпеливые и подробные консультации по виртуализации и системному администрированию.

сервисы бывает удобно аггрегировать на одном узле, т. к. они могут работать взаимосвязанно, например, с одними и теми же данными).

С развитием технологий постоянно нарастающие мощности компьютерного оборудования намного опережают рост запросов к ресурсам со стороны программ. В результате всё выше поднимается верхняя граница по числу возможных сервисов на одном сервере, и все более невыгодно становится держать сервер только для одного сервиса — ресурсы простаивают.

Виртуализация в операционных системах

Виртуализация в операционных системах понимается широко и разнообразно. Если пробовать выделить общее, то речь всегда идёт о запуске в той или иной степени полноценной операционной системы (далее ОС), работающей *в рамках другой ОС* (или в **гипервизоре**: специализированной ОС, которая только и умеет, что обеспечивать работу других ОС внутри себя). Такую виртуализованную ОС называют чаще всего **гостевая ОС** (а также виртуальное окружение и др.), а ОС, внутри которой выполняется гостевая, — **хост-системой** (также аппаратный узел, Hardware Node). Виртуальное окружение, внутри которого выполняется программа, обычно неотличимо для нее от невиртуализованной ОС. Здесь мы не будем подробно рассматривать разные технологии виртуализации, для этого лучше обращаться к специальным источникам[1].

В контексте системного администрирования наиболее существенны два свойства любой технологии виртуализации. Первое состоит в том, что виртуализация вводит *промежуточный уровень* между оборудованием (аппаратным обеспечением) и программным обеспечением. Использование этого уровня даёт администратору возможность гибко управлять распределением аппаратных ресурсов между параллельно выполняющимися программами. Второе — помещение программ в виртуальное окружение довольно надёжно *изолирует* их от всех прочих программ, исполняющихся на этом же физическом оборудовании вне данного окружения. Тем самым радикально снижается вероятность побочного влияния работы одних программ на работу других, в том числе снижается вероятность атак, построенных на подобных побочных взаимодействиях.

[1] См. например хорошую вводную статью (на английском языке) `http://www-128.ibm.com/developerworks/linux/library/l-linuxvirt/?ca=dgr-lnxw01Virtual-Linux`.

Виртуализация сервисов

Традиционный подход к размещению сервисов состоит в том, что на *одном* физическом сервере работает *одна* операционная система, в рамках которой выполняются несколько **системных служб**, предоставляющих те или иные сервисы.

У нескольких задач, выполняющихся в рамках одной ОС (иначе говоря, в **разделяемой среде**), всегда есть возможности для взаимодействия, поскольку ОС именно для организации такого взаимодействия и предназначена. Кроме того, в разделяемой среде задачи совместно пользуются общими интерфейсами (файловая система, структуры ядра...) и в конечном итоге общими аппаратными ресурсами, что открывает дорогу побочным взаимодействиям. В случае независимых сервисов подобное взаимодействие равнозначно помехам и нарушению работы.

В UNIX-системах, при разработке которых многозадачность являлась одним из важнейших приоритетов, уже присутствует комплекс средств для решения задач изоляции и контроля. К подобным средствам относятся, например, системный вызов `chroot()`, обеспечивающий изоляцию на уровне файловой системы, во FreeBSD существует также технология `jail`, дополнительно к этому ограничивающая доступ к структурам ядра. В последних версиях ядра Linux реализована поддержка аппаратной паравиртуализации (`kvm`, применимо в том случае, если ее поддерживает процессор). Однако стандартные средства UNIX-систем не решают всех проблем совместного размещения сервисов.

Внимание

Проще говоря, в разделяемой среде работает логика: «кто успел, тот и съел». Поэтому в ситуации, когда критически важна работа нескольких сервисов, в особенности требовательных к аппаратным ресурсам, работа в рамках одной разделяемой среды (ОС) может быть нежелательна.

Путь разрешения обозначенных проблем — использовать средства виртуализации, чтобы исключить возможности побочного взаимодействия и получить дополнительную к средствам ОС точку контроля за потреблением аппаратных ресурсов. Так, в последнее время нормой постепенно становится размещение нескольких сервисов в рамках одного сервера не в одной и той же ОС, а в *отдельных виртуальных* *окружениях*. Далее мы будем называть такое виртуальное окружение с работающей в нём системной службой **виртуальным контейнером**.

С точки зрения задачи виртуализации сервисов не принципиально, какая конкретно технология виртуализации выбрана для создания контейнера. При большой вычислительной нагрузке это вряд ли может быть виртуальная машина, эмулирующая все оборудование компьютера, но вполне может использоваться одна из систем с гипервизором. Для UNIX-систем может быть весьма удобно применять одну из технологий виртуализации на уровне ОС, доводящих средства изоляции и контроля UNIX до уровня, достаточного для организации полноценных виртуальных окружений. К таким технологиям относится OpenVZ (широко применяемый в ALT Linux) и VServer.

Высокая надёжность и гарантированное обслуживание

Во многих областях обслуживания абонентов критически важно обеспечивать постоянную работу сервиса, минимизировать простои из-за неполадок или управляться с большим потоком запросов, чтобы они не приводили к нарушению работы сервиса. Системы, обладающие техническими характеристиками для выполнения этих условий, называют **системами высокой надёжности** (high availability systems). Помещение сервиса в виртуальный контейнер можно использовать для повышения надёжности его работы.

Ограничение ресурсов

Внимание

Аппаратные ресурсы, доступные виртуальным контейнерам, должны быть всегда меньше аппаратных ресурсов всей системы.

Бывают ситуации, особенно при работе под большой нагрузкой, когда затребованные сервисом ресурсы настолько велики, что могут привести к неработоспособности самой операционной системы. Средствами виртуального окружения можно ограничить *верхний предел* ресурсов системы, выделяемых сервису в рамках виртуального контейнера.

Т. е. технология виртуализации применяется в данном случае как средство контроля ресурсов, потребляемых отдельной задачей, и обеспечивает непревышение этих ресурсов (поскольку задача не имеет прямого доступа к оборудованию). Эту задачу большинство средств виртуализации решает весьма надёжно.

Такой подход повышает надёжность системы, поскольку гарантирует операционную систему от неработоспособности по причине превышения ресурсов со стороны сервиса. Однако этот метод, естественно, не может гарантировать от сбоев самого сервиса, вызванных, например, большим потоком запросов.

Помещение сервиса в виртуальный контейнер позволяет минимизировать время простоя даже в ситуации сбоя виртуального контейнера из-за превышения ресурсов, поскольку потребуется только восстановить виртуальный контекст, что в среднем быстрее, чем перезагрузка всего сервера.

Таким образом, даже в случае размещения на сервере только *одного* сервиса, оправдано его помещение в виртуальный контейнер.

Гарантированные ресурсы

Внимание

Аппаратные ресурсы, доступные виртуальному контейнеру, должны быть гарантированно **больше** некоторого минимального значения.

В ряде случаев необходимо предоставить клиентам сервиса некоторые гарантии качества обслуживанияш (QoS, quality of service). Примером такой ситуации может служить телефонный разговор: после соединения клиент получает гарантию, что во время разговора не закончатся ресурсы канала связи и разговор не прервётся. Другой пример, уже из области системного администрирования: весьма желательно гарантировать, что даже при максимальной нагрузке на сервере (в том числе в ситуации DoS-атак), у сервиса SSH будет достаточно ресурсов, чтобы обеспечить удалённый вход на сервер системного администратора, который мог бы принять срочные меры.

Чтобы обеспечить качество обслуживания для сервиса, администратору необходимо гарантировать достаточный объем аппаратных ресурсов для этого сервиса, т. е. исключить ситуации сбоя сервиса из-за физической нехватки ресурсов в системе.

И здесь можно использовать виртуализацию сервисов. Если сервис работает в виртуальном контейнере, то гарантия минимального уровня ресурсов сводится к тому, что процессы, выполняемые вне данного контейнера, в сумме не затребуют больше ресурсов, чем разность между объемом ресурсов системы и ресурсами данного контейнера.

Так, если в системе параллельно выполняется несколько сервисов, каждый из которых помещён в отдельном виртуальном контейнере, то задача администратора — ограничить ресурсы каждого контейнера таким образом, чтобы сумма выделенных им ресурсов не превышала всех ресурсов системы, и при этом выделенные каждому сервису ресурсы были достаточны для его нормальной работы.

В случае, если сервисы не помещены в виртуальные контейнеры, добиться гарантированной нижней границы доступных ресурсов можно только в том случае, если на одном физическом сервере выполняется *только один сервис*. В противном случае работает правило «кто первый встал, того и тапки», т. е. любой из сервисов может в какой-то момент занять большой объём свободных ресурсов, так что оставшегося не хватит другому сервису для нормальной работы.

Иначе говоря, в разделяемой среде *невозможно* гарантировать отдельному сервису нижнюю границу доступных ресурсов, и, следовательно, невозможно дать гарантии клиентам по качеству обслуживания.

Изоляция ergo защищённость

default deny

Сервис, принимающий запросы от внешних клиентов, является одним из классических источников для разного рода уязвимостей в безопасности системы. Зачастую несанкционированный доступ к ОС осуществляется именно через эксплуатацию уязвимостей некоторого сервиса. Если подвергшийся атаке сервис помещён в виртуальный контейнер, последствия взлома могут быть значительно менее разрушительными, чем в случае, когда сервис выполняется непосредственно в операционной системе сервера. В последнем случае ОС сервера может быть взломана через один из сервисов, а под удар подставляются все выполняющиеся сервисы.

В случае виртуализованного сервиса, взломщик может получить доступ только в рамках виртуального контейнера, так что даже если на сервере параллельно выполняются другие сервисы (в других виртуальных контейнерах), то их работе данный инцидент угрожать не будет. Попытки взломщика исчерпать все физические ресурсы приведут только к исчерпанию ресурсов данного контейнера. Таким образом, взломщик не сможет затруднить удалённый доступ администратора на сервер, исчерпав все доступные ресурсы. Кроме того, изнутри контейнера у взломщика нет возможности перезаписать загрузчик сервера и получить какой-либо прямой контроль над оборудованием.

При подобном инциденте администратор может просто целиком уничтожить скомпрометированный виртуальный контекст и восстановить его и данные из резервной копии, не перезагружая сервер и не останавливая работу прочих выполняющихся на нём сервисов.

Программные и аппаратные средства ограничения ресурсов

Из предыдущих разделов становится понятно, что при использовании виртуальных контейнеров на сервере работоспособность как виртуализованных сервисов, так и основной ОС сервера весьма существенно зависит от надёжности работы механизмов ограничения и изоляции, заложенных в технологии виртуализации.

Такие механизмы ограничения могут быть двух родов: аппаратные и программные. Если используется платформа, поддерживающая виртуализацию на аппаратном уровне, то алгоритмы изоляции контекстов реализованы непосредственно в оборудовании, подвести здесь может только ошибка в микросхеме. Во всех остальных технологиях виртуализации механизмы ограничения реализованы так или иначе программно, и, как показывает практика, в программных реализациях ошибки встречаются.

Производители программного обеспечения, выполняющего те или иные серверные функции (т. е. предоставляющие некоторый сервис), могут встраивать в свои продукты собственные средства контроля ресурсов, предоставляющие примерно ту же функциональность, что и ограничение средствами виртуального контейнера. Однако как в любой программной реализации в этих механизмах возможны ошибки, дающие возможность при определённых условиях обходить установленные ограничения.

Однако вероятность ошибок в механизмах ограничения тем выше, чем больше *разных* механизмов использовано в системе. Поэтому в случае размещения на сервере нескольких виртуальных контейнеров с сервисами правильнее использовать один общий механизм контроля, предоставляемый средством виртуализации. Кроме того, если используется стандартное для дистрибутива средство виртуализации (OpenVZ в случае ALT Linux), то возможные ошибки, критичные для безопасности, будут быстрее обнаружены и исправлены.

Защита интерфейса управления

При организации сервера с использованием виртуализованных сервисов критически важной точкой для безопасности системы становится интерфейс управления самими виртуальными контейнерами. Доступ к этому интерфейсу должен быть наиболее обстоятельно защищён.

Возможные варианты защиты доступа к интерфейсу управления:

- запретить удалённый доступ по сети, разрешить только доступ с локальной консоли (виртуальная консоль, последовательный порт);
- физически изолировать сетевое подключение для интерфейса управления (отдельный сетевой адаптер);
- логически изолировать сетевое подключение для интерфейса управления (отдельный VLAN-интерфейс);
- использовать защищённые сетевые соединения для удалённого доступа к интерфейсу управления (защищённый туннель ipsec, openvpn, pptp или ssh-туннели).

Консолидация серверов

Внимание

Консолидация сервисов на одном сервере ведет к экономии на затратах на обслуживание оборудования.

Выше уже говорилось о том, что современное оборудование позволяет одновременно выполнять значительно больше одного сервиса на одном сервере. При использовании некоторых технологий виртуализации вычислительных ресурсов сегодня хватает на организацию множества виртуальных контекстов на сервере, даже с учётом дополнительных вычислительных расходов на виртуализацию. Технологии виртуализации на уровне операционной системы (для UNIX-систем) позволяют одновременно эксплуатировать сотни виртуальных контекстов на одном достаточно мощном сервере.

Типичная сфера применения подобных решений — хостинг-провайдеры: при использовании виртуальных контекстов каждому клиенту можно выдать независимое окружение с правами суперпользователя, предоставив администрирование внутри контекста на его полное усмотрение.

Выгоды от малой и до массовой консолидации сервисов на одном сервере в первую очередь лежат в сфере снижения затрат на приобретение, размещение и обслуживание оборудования. Традиционно подобная консолидация сдерживалась соображениями безопасности и качества обслуживания, которые также обеспечиваются средствами виртуальных контейнеров.

Динамическое перераспределение ресурсов

Внимание

Виртуализация сервисов даёт возможность простого и безущербного внедрения, перемещения и выведения из строя любого из сервисов без ущерба общей инфраструктуре и без простоя.

Еще одна неочевидная, но немаловажная выгода, представляемая помещением сервисов в виртуальные контейнеры происходит от того, виртуальные контексты, с одной стороны, самодостаточны, а с другой — отчуждаемы от оборудования. Благодаря этому, администратор имеет возможность манипулировать виртуальными контейнерами как целостными объектами, в том числе переносить с одного сервера на другой. Так, для большинства технологий виртуализации есть средства для «заморозки» контейнера в текущем состоянии, с возможностью последующей разморозки, в том числе на другом сервере, при том что выполняемые внутри контейнера процессы даже не заметят происшедшей перемены.

Такие возможности позволяют даже в процессе работы мигрировать виртуализованные сервисы, например, по разным узлам сети, гибко перераспределяя вычислительную нагрузку, в том числе среди узлов кластера. Кроме того, можно уничтожить ставший ненужным виртуальный контейнер, при этом не возникнет необходимости продавать высвободившийся сервер: просто остальные контейнеры смогут разделить высвобившиеся мощности.

Краткое введение в OpenVZ[1]

Различные технологии виртуализации позволяют запускать на одном сервере несколько в той или иной степени независимых экземпляров операционной системы (или даже разных операционных систем). Системы виртуализации различаются как степенью изоляции выполняющихся операционных систем, так и другими особенностями и ограничениями. В дистрибутивах ALT Linux в качестве технологии виртуализации используется OpenVZ[2].

OpenVZ является реализацией виртуализации на уровне операционной системы (подобно, например, технологии `jail`, доступной в ОС `FreeBSD`). Все запущенные с её помощью экземпляры операционной системы (далее **виртуальные контейнеры**) используют один общий экземпляр ядра Linux. Это способствует повышению производительности по сравнению с технологиями, основанными на полной виртуализации оборудования, а также даёт администратору основной системы дополнительные возможности по контролю и управлению виртуальными контейнерами.

Минусом этой технологии является невозможность использования в разных виртуальных контейнерах операционных систем, не основанных на ядре Linux, а также разных версий ядра.

Система OpenVZ использует модифицированное ядро, обеспечивающее виртуализацию, изоляцию, управление ресурсами, сохранение состояний виртуальных серверов.

Виртуализация и изоляция

Каждый виртуальный сервер выглядит с точки зрения его пользователя или администратора практически неотличимо от обычного физического сервера. Каждый виртуальный сервер имеет свои собственные:

Файлы

Системные библиотеки, приложения, виртуализованные файловые системы `/proc` и `/sys`, виртуализованные блокировки и т. п.

Пользователи и группы

Свои собственные пользователи и группы, включая root.

[1] Текст отчасти основан на `http://ru.wikipedia.org/wiki/OpenVZ`
[2] `http://openvz.org`

`Дерево процессов`

Виртуальный контейнер имеет доступ только к своим собственным процессам (начиная с `init`). Идентификаторы процессов (`PID`) также виртуализованы, поэтому `PID` программы `init` равняется 1, как ему и следует быть.

`Сеть`

Виртуальное сетевое устройство (`venet`) обеспечивает наличие виртуальной сети в пределах физического сервера. Каждый виртуальный контейнер может иметь свой набор правил фильтрации пакетов.

`Устройства`

При необходимости администратор физичесого сервера может давать виртуальным контейнерам доступ к реальным устройствам, например, сетевым адаптерам, портам, разделам диска и т. д.

`Объекты IPC`

Разделяемая память, семафоры, сообщения.

Управление ресурсами

Ядро OpenVZ обеспечивает ограничение ресурсов каждого виртуального контейнера так, чтобы исключить для него возможность помешать работе других виртуальных контейнеров.

Основными компонентами управления ресурсами в OpenVZ являются: двухуровневая дисковая квота, специальный планировщик процессора, ограничения (user beancounters) и дополнительные полномочия (capabilities). Все эти ресурсы могут быть изменены во время работы виртуального сервера без перезагрузки.

Использование диска

Администратор физического сервера может установить для виртуальных серверов дисковые квоты, в терминах дискового пространства и количества индексных узлов (i-nodes), число которых примерно равно количеству файлов. Это верхний уровень дисковой квоты.

В дополнение к этому, администратор виртуального контейнера может использовать обычные средства управления дисковыми квотами внутри своего контейнера.

Работая внутри контейнера, можно определить объем доступного ему дискового пространства, выполнив следующие команды:

```
# echo "simfs / simfs rw 0 0" >>/etc/mtab
$ df -h
```

Одной из не всегда очевидных проблем при планировании ресурсов является дисковый ввод/вывод. При создании нагруженного сервера с обширным вводом/выводом в различных контейнерах следует помимо общей практики использования RAID1+ для повышения надёжности множества сервисов на одной системе по возможности применять разнесение потоков данных по различным физическим дискам[1].

Например, практически возможно использование одного физического сервера как терминального (LTSP) и для сборки программных пакетов (hasher) при условии, что терминал-сервер и пакетные базы размещены на программном RAID1, а сборка ведётся на отдельном SCSI-диске; иначе даже существенный объём RAM (например, 4Gb) не поможет избежать заметного взаимного влияния контейнеров при дисковых операциях.

Планировщик процессора

Планировщик процессора в OpenVZ также двухуровневый. На первом уровне планировщик на основании значения параметра cpuunits решает, какому виртуальному серверу дать квант процессорного времени. На втором уровне стандартный планировщик Linux решает, какому процессу выбранного виртуального сервера дать квант времени, основываясь на стандартных приоритетах процесса в Linux и т. п.

Администратор физического сервера может устанавливать количество процессорного времени, выделяемого каждому виртуальному серверу, как на основании системы весов (`cpuunits`), так и задавая явные пределы (`cpulimit`).

Ограничения

OpenVZ предоставляет для каждого виртуального контейнера набор счётчиков, ограничений и гарантий. Имеется набор из примерно 20 параметров, которые выбраны таким образом, чтобы никакой виртуальный контейнер не мог злоупотребить каким-либо ресурсом, и

[1]Подробную информацию на эту тему можно найти в Multi-Disk HOWTO (его можно найти в Интернет или в пакете `howto-html-en`).

таким образом помешать работе других виртуальных контейнеров на этом же физическом сервере. Смысл порогового и максимального значений может различаться в зависимости от параметра.

По умолчанию для виртуального контейнера устанавливаются достаточно низкие значения ограничителей, поэтому в ряде случаев может оказаться, что выделяемых ресурсов недостаточно для нормального функционирования процесса в контейнере. В случае, если работа процесса завершается ошибкой выделения памяти, ошибкой создания процесса или открытия файла, следует снять показания счётчиков виртуального контейнера:

```
# cat /proc/user_beancounters >~/user_beancounters.`date
"+%Y%m%d-%H%M"`
```

Затем повторить операцию, которая привела к ошибке, и снова снять показания счётчиков. Затем их нужно сравнить и выяснить, какой из счётчиков failcnt увеличил значение — соответствующий лимит нужно будет увеличить. Краткое пояснение смысла разных лимитов приведено ниже, более подробную информацию можно найти в `man vzctl`. Нужно также принимать во внимание, что некоторые лимиты установлены в блоках (обычно 512 байт), некоторые — в страницах памяти (4 Кб на x86 архитектуре), некоторые в байтах и других натуральных величинах.

Память

`kmemsize`

Количество памяти, используемое ядром. Этот параметр связан со значением параметра `numproc` Каждый процесс потребляет некоторое количество памяти ядра: 30— 50 килобайт, как минимум 16. Важно иметь достсточно большой зазор между пороговым и максимальным значениями, чтобы не вынуждать ядро завершать процессы, выполняющиеся в виртуальном окружении.

`lockedpages`

Количество страниц памяти, заблокированной при помощи `mlock`.

`privvmpages`

Объем памяти, размещаемой процессами. Память, разделяемая между разными процессами, не включается в это значение.

`shmsize`

Объем памяти, доступный для межпроцессного взаимодействия через разделяемую память. Установка порогового значения, отличающегося от максимального, не имеет смысла.

`physpages`

Объём физической памяти, используемое виртуальным контейнером. Используется только для учёта, пороговое значение должно быть выставленно в 0.

`vmguarpages`

Гарантированный размер доступной памяти в страницах. Имеет смысл изменять только пороговое значение.

`oomguarpages`

Пока использование памяти в страницах не превышает заданного порогового значения, процессы не будут принудительно завершены из-за недостка памяти даже при её реальной нехватке.

Процессы и файлы

`numproc`

Количество процессов и обеспечиваемых ядром потоков. Установка порогового значения, отличающегося от максимального, не имеет смысла.

`numflock`

Максимальное количество файловых блокировок. Рекомендуется оставлять зазор между пороговым и максимальным значениями.

`numpty`

Количество псевдотерминалов. Не может превышать 255. Установка порогового значения, отличающегося от максимального, не имеет смысла.

`numsiginfo`

Количество структур `siginfo`. Установка порогового значения, отличающегося от максимального, не имеет смысла.

`dcachesize`

Максимальный размер кешей файловой системы (в байтах). Максимальное значение должно быть больше порогового.

`numfile`

Максимальное количество открытых файлов. Установка порогового значения, отличающегося от максимального, не имеет смысла.

Сеть

`numtcpsock`

Максимальное количество сокетов протокола TCP. Установка порогового значения, отличающегося от максимального, не имеет смысла.

`tcpsndbuf`

Максимальный размер буферов для данных, отправляемых по протоколу TCP. Пороговое значение не должно быть меньше 64 Кб, а разница между ним и максимальным значением не должна быть меньше значения `numtcpsock`, умноженного на 2,5.

`tcprcvbuf`

Максимальный размер буферов для данных, получаемых через протокол TCP. Пороговое значение не должно быть меньше 64 Кб, а разница между ним и максимальным значением не должна быть меньше значения `numtcpsock`, умноженного на 2,5.

`othersockbuf`

Максимальный размер прочих (не TCP) буферов отправки. Увеличение лимита ускоряет связь через локальные сокеты.

`dgramrcvbuf`

Максимальный размер прочих (не TCP) буферов получения. Установка порогового значения, отличающегося от максимального, не имеет смысла.

`numothersock`

Максимальное количество не-TCP сокетов (локальных, UDP и других). Установка порогового значения, отличающегося от максимального, не имеет смысла.

`numiptent`

Макисмальное количество записей в таблицах фильтрации сетевых пакетов (`iptables`).

Дополнительная информация

- Официальный сайт проекта `http://openvz.org`
- Вики разработчиков и пользователей OpenVZ `http://wiki.openvz.org`
- Форум проекта `http://forum.openvz.org`
- Рекомендации по настройке OpenVZ в ALT Linux `http://www.freesource.info/wiki/AltLinux/Dokumentacija/OpenVZ`

Развёртывание виртуальных серверов

Web-интерфейс управления виртуальными серверами доступен по адресу `https://server-name:8080/index.scm/ovz-ve`. Вы можете просмотреть список доступных виртуальных серверов, запустить, остановить или перезапустить выбранные. Для запущенных виртуальных серверов можно изменить настройки виртуального контейнера (ссылка «Настроить контейнер») или настроить сервисы, выполняющиеся внутри контейнера (ссылка «Настроить службы»). Также имеется возможность создания нового виртуального контейнера.

Виртуальный контейнер создаётся на основе шаблона, который, в свою очередь, создаётся на основе профиля.

Профиль представляет собой описание конфигурации и составляющих частей виртуального сервера. Главной составляющей профиля является список пакетов, которые будут установлены в шаблон, создаваемый на основе этого профиля. Для создания работоспособного контейнера в список пакетов обязательно нужно включать пакеты `basesystem` и `sysklogd`. Чтобы в созданном контейнере была возможность управлять параметрами системы через web-инетрфейс, аналогично поставляемым ALT стандартным контейнерам, в профиль следует включить пакет `alterator-fbi`.

Интерфейс управления профилями позволяет просмотреть, изменить или удалить существующие профили, а также создавать новые.

Шаблон представляет собой образ файловой системы виртуального сервера. Он создаётся на основе профиля и используется для создания экземпляров виртуальных серверов.

Интерфейс управления шаблонами позволяет удалять существующие шаблоны и создавать новые. Для создания шаблона надо выбрать для него имя и профиль, на основе которого он будет создан.

Одновременное создание двух шаблонов невозможно.

Обновление виртуальных серверов

Обновление виртуальных серверов рекомендуется производить только из официального источника обновлений дистрибутива.

Обновление виртуальных серверов может производиться при помощи `apt`. В каждом виртуальном сервере доступны web-интерфейсы для управления источниками обновлений («Система->Источники обновлений») и обновления («Система->Обновления»).

В случае, когда на сервере или в сети работает значительное число виртуальных серверов, рекомендуется создать специальный контейнер, содержащий `squid`, настроенный на кэширование обновлений и настроить контейнеры на обновление через него или же обеспечить зеркалирование обновлений с сайта ALT Linux и производить обновление из локального зеркала.

Глава 5

Настройка системных служб

Удалённый доступ и удалённое администрирование (OpenSSH)

Одна из самых распространённых задач, возникающих при работе в сети, — удалённый доступ к другим компьютерам, в том числе их удалённое администрирование. Технология SSH (Secure Shell) призвана предоставить возможность безопасного способа управления удалённой системой. Посторенная по клиент-серверной модели, она позволяет при помощи ssh клиента подключаться к ssh серверу, предоставляя командную оболочку для работы с удалённой системой.

SSH реализует соединение с удалённым компьютером, защищающее от:

- прослушивания данных, передаваемых по этому соединению;
- манипулирования данными на пути от клиента к серверу;
- подмены клиента или сервера путём манипулирования IP-адресами, DNS или маршрутизацией.

В дополнение к отличным характеристикам в области обеспечения безопасного клиент-серверного соединения, SSH обладает следующими возможностями:

- Сжатие передаваемых данных.
- Туннелирование каналов внутри установленного соединения, в том числе соединений с X-сервером, что позволяет запускать графические приложения на удалённой машине.

- Широкая распространённость: существуют реализации SSH для самых различных аппаратных платформ и операционных систем.

OpenSSH — это включённая в дистрибутивы ALT Linux реализация SSH. Она представлена следующими пакетами:

- openssh-clients — клиентская часть (утилита ssh, sftp и др.).
- openssh-server — серверная часть (сервер sshd, sftp-server).
- openssh — утилиты, необходимые для работы как клиентской, так и серверной части.

Пакет openssh-server в ALT Linux 4.0 Server устанавливается по умолчанию, позволяя непосредственно после установки системы воспользоваться возможностью удалённого администрирования.

В простейшем виде сеанс работы будет выглядеть следующим образом. Запускается ssh клиент (ssh) с указанием имени пользователя на удалённой системе (если оно отличается от локального) и имени машины (либо IP-адреса). На удалённой машине, естественно, должна быть настроена и работать серверная часть — sshd.

```
$ ssh user@hostname
```

После этого sshd предоставляет командную оболочку для работы на удалённой системе. Ваш компьютер становится удаленным терминалом другого компьютера.

Аутентификация

Прежде чем начать удалённый сеанс работы, необходимо пройти процедуру **аутентификации**, аналогичную той, что происходит при непосредственной работе с физическим терминалом Linux-системы. Утилита sshd поддерживает различные способы аутентификации, наиболее часто используются два из них: аутентификация **по паролю** и аутентификация **по ключу**.

Параметры, относящиеся к способу аутентификации, а так же все прочие настройки sshd указываются в конфигурационном файле /etc/openssh/sshd_config. Для вступления в силу изменений, вносимых с конфигурационный файл, каждый раз после правки файла необходимо выполнять команду

```
# service sshd reload
```

Аутентификация по паролю

Аутентификации по паролю включается добавлением строки `PasswordAuthentication yes` в файл `etc/openssh/sshd_config`. Хотя пароль в этом случае и не передается по сети в открытом виде, всё же существует опасность подбора пароля. Поэтому не следует выбирать слишком простые пароли.

Аутентификация по ключу

Аутентификация по ключу включается строкой
`PubkeyAuthentication yes`
в файле `/etc/openssh/sshd_config`. При использовании этой схемы аутентификации будет использоваться асимметричное шифрование, следовательно, необходимо сгенерировать пару ключей: секретный и публичный[1].

Для генерации пары ключей на компьютере пользователя необходимо выполнить `$ ssh-keygen` с необходимыми параметрами, например:
`$ ssh-keygen -t dsa`

На вопрос о файле для сохранения ключа нажать *Enter* (принять вариант по умолчанию). Будет задан вопрос, следует ли создать пароль к ключу. Поскольку ключи рекомендуется создавать с паролем, нужно указать свой пароль.

После генерации ключей образуются 2 файла: `~/.ssh/id_dsa` — секретный ключ, и `~/.ssh/id_dsa.pub` — публичный. Его предстоит поместить на удалённый сервер.

Следующим шагом содержимое файла `id_dsa.pub` неоходимо добавить в файл `~/.ssh/authorized_keys2` на удалённой машине. Если под этой учётной записью будет входить один пользователь, файл `id_dsa.pub` можно просто скопировать, назвав его `authorized_keys2`. Также можно воспользоваться утилиой `ssh-copy-id`:
`$ ssh-copy-id username@hostname`

Обратите внимание, что для выполнения этой операции должна работать аутентификация по паролю. В дальнейшем её можно будет отключить.

[1] Более подробно о технологии асимметричного шифрования можно узнать на `http://www.intuit.ru/department/security/networksec/7/` и `http://www.mgpu.ru/crypto/2.1.htm`.

После указанных действий можно использовать аутентификацию по ключу. В процессе такой аутентификации запрашивается не пароль удаленного пользователя, а пароль на ваш ключ, что делает ssh сессию ещё более безопасной.

Безопасность

При настройке `sshd` следует весьма щепетильно отнестись к вопросам безопасности. SSH — очень важная часть системы. Она является незаменимым помощником любого администатора Linux, предоставляя ему доступ к командной оболочке удалённой системы. Но как раз по этой же причине данная служба может стать и источником бед.

Далее приводятся лишь некоторые рекомендации, призванные обезопасить вашу систему от насанкционированного доступа.

По возможности следует использовать способ аутентификации по ключу как наиболее безопасный. Если вы все же по каким-либо причинам используется аутентификация по паролю, то следует тщательнейшим образом отнестись к выбору паролей. В вопросах выбора паролей надо руководствовться теми же соображениями что и при выборе паролей при регистрации пользователей.

В качестве дополнительных мер, можно явно указать список пользователей, которым разрешена либо запрещена удалённая работа. За это отвечают параметры `AllowUsers` и `DenyUsers` в `/etc/openssh/sshd_config`.

Наконец, можно осложнить жизнь потенциальному злоумышленнику, отключив возможность удалённой регистрации в системе для суперпользователя (`root`). Для этого в `/etc/openssh/sshd_config` нужно добавить строку `PermitRootLogin no`.

Как и в случае с любыми потенциально опасными службами, следует постоянно следить за обновленями безопасности, связанными с данными пакетами. Администратор системы, заботящийся о безопасности, должнен непременно устанавливать обновления штатным способом:

```
# apt-get update
# apt-get upgrade
```

За более детальной информацией по настройке sshd следует обращаться к руководству `sshd_config(5)`. Настройка клиентской части подробно описана в `ssh_config(5)`.

Служба DNS (Bind)

Если локальная сеть не подключена к сети Интернет, вполне возможно, что внутренний DNS-сервер в ней не нужен. За преобразование доменного имени в IP-адрес и обратно в Linux отвечает несколько разных механизмов, лишь один из которых базируется на службе доменных имён. В самом простом случае имена всех компьютеров вместе с их адресами можно записать в файл `/etc/hosts`. Порядок просмотра различных пространств имён указывается в файле `/etc /nsswitch.conf`. Строка «`hosts: files nisplus nis dns`» этого файла предписывает приложениям, пользующимся стандартной функцией `gethostbyname()` *сначала* заглянуть в `/etc/hosts`, затем попытаться получить таблицы hosts службы nisplus или nis, а только затем обращаться к DNS-серверу. Это вполне разумное правило: учитывая время выполнения запроса в каждом случае.

Если задачу преобразования имён в адреса взял на себя провайдер, собственный DNS-сервер можно тоже не заводить. В этом случае на всех компьютерах в качестве сервера имён указывается сервер провайдера (поле «`nameserver`» в файле `/etc/resolv.conf`), к которому и идут все запросы. Даже если внутренняя сеть организована согласно RFC1918 (т. н. **интранет**) и адреса компьютеров в ней не могут «светиться» в сети Интернет, DNS-запросы во внешний мир работать будут. Между собой компьютерам предлагается общаться с помощью `/etc/hosts` или IP-адресов. Кстати, некоторые службы и системные утилиты, работающие с доменными именами, запускаются в ALT Linux с использованием chroot (в каталоге `/var/resolv`), так что после изменения упомянутых файлов рекомендуется выполнить команду `update_chrooted conf`.

Однако уже здесь очевидны две задачи, для решения которых можно организовать собственную службу доменных имён.

Первая задача — уменьшение времени ответа на DNS-запрос абонентов внутренней сети. Если канал подключения к Интернет обладает большим временем задержки (скажем, четверть секунды), то работа с данными, включающими в себя *много* доменных имён (например, с www-страницами) может стать весьма медленной, хотя общий объём трафика будет невелик. Система доменных имён — распределённая база данных, замечательно поддерживающая механизм **кеширования** запросов. Первое обращение к кеширующему DNS-серверу приводит к выполнению **рекурсивного запроса**: опрашивается сервер более высокого уровня, который, если не знает ответа, передаст запрос дальше.

Результат запроса оседает в кеше, так что все последующие обращения именно к этой записи дальше кеширующего сервера не уйдут. **Время жизни** (Time To Live, TTL) записи в кеше определяется хозяином запрошенного доменного имени. По истечении TTL запись из кеша удаляется.

Вторая задача — именование компьютеров в интранет-сети. Это бывает важно, если среди компьютеров внутренней сети есть свои серверы (например, корпоративный WWW-сервер), к которым другие компьютеры обращаются по доменному имени. Поскольку адреса такой сети не пойдут дальше межсетевого экрана, вы можете использовать имя какого угодно, — в том числе несуществующего — домена и составить табличку в `/etc/hosts`. Однако *раздавать* эту табличку лучше всё-таки средствами DNS.

Обе эти задачи можно решить, воспользовавшись Bind — мощным полнофункциональным DNS-сервером (обратите внимание, пакет и сервис называются «`bind`», а сам демон — «`named`»). Bind — программа весьма сложная (общий объём документации в пакете bind-doc — более полутора мегабайтов, одних RFC поддерживается около сорока), с замысловатой моделью безопасности и долгой историей эксплуатации. Как во всякой сложной программе, в Bind изредка обнаруживаются уязвимости, поэтому в ALT Linux этот сервер (Bind версии 9) запускается с использованием chroot. В `/etc` от Bind осталась только символьная ссылка на главный файл настроек, `named.conf`. Всю свою жизнь named проводит, запертый в `/var/lib/bind` в качестве корневого каталога. Там у него есть собственный `/etc`, содержащий (в варианте ALT Linux) целый набор включаемых друг в друга конфигурационных файлов, `/var` и даже `/dev`. Обратите внимание, что все пути к файлам и каталогам в настройках Bind начинаются именно из этого каталога, и `/zone` соответствует `/var/lib/bind/zone`.

Для того, чтобы запустить named в кеширующем режиме, достаточно раскомментировать и заполнить раздел настройки «`forwarders`» (вышестоящие серверы) в файле `/var/lib/bind/etc/options.conf`. Обратите внимание на возможные *ограничения* на право обращаться к серверу с обычными и рекурсивными запросами (настройки «`allow-query`» и «`allow-recursion`»). Можно раскомментировать находящиеся в этом файле разумные установки по умолчанию. Эти настройки открывают доступ только абонентам локальных сетей, т. е. сетей, к которым компьютер подключён непосредственно.

```
# grep allow- /var/lib/bind/etc/options.conf
        //      allow-query { localnets; };
```

```
//      allow-recursion { localnets; };
```

Использование Bind для полноценного именования компьютеров в локальной сети требует создания двух **зон** (т. н. **прямой** и **обратной**), содержащих в виде **записей** определённого формата информацию о доменных именах компьютеров и об их роли в этих доменах. Каждая зона должна включать запись типа SOA (State Of Authority, сведения об ответственности). В этой записи определяются основные временные и административные параметры домена, в том числе электронный адрес лица, ответственного за домен (администратора) и серийный номер зоны. Серийный номер — число в диапазоне от 0 до 4294967295 (2**32); каждое изменение, вносимое в зону, должно сопровождаться *увеличением* этого номера. Обнаружив увеличение серийного номера, кеширующие и вторичные серверы признают все закешированные записи из этой зоны устаревшими. Структура номера может быть любой, лишь бы он постоянно увеличивался. Удобно использовать формат *годмесяцчисловерсия*, где все числа, кроме года, двузначные, а *версия* может обнуляться раз в день, соответствовать времени (например, по формуле «`100*(часы*60+минуты)/(60*24)`» или иметь сквозную нумерацию (в этом случае появляется сложность с переходом от версии 99 к версии 100, то есть 0). Даже если серийный номер генерируется автоматически, рекомендуется пользоваться этим форматом, наглядно отражающим время создания зоны. Пример зоны, не содержащей ничего, кроме записи SOA и обязательной записи типа NS, находится в файле `/var/lib/bind/zone/empty`.

Кроме записи типа SOA, в каждой зоне должна быть хотя бы одна запись типа NS (Name Server), указывающая адрес DNS-сервера, авторитетного в этом домене (как минимум — адрес сервера, на котором запущен named). Несколько зон включаются в настройку Bind автоматически (файл `/var/lib/bind/etc/rfc1912.conf`). Они нужны для обслуживания сети, привязанной к сетевой заглушке (127.0.0.1/8). Стоит обратить внимание на то, что имя домена, который обслуживается зоной, задаётся в файле настроек, а в самом файле зоны можно использовать *относительную* адресацию (без «.» в конце имени), так что операция переименования домена делается редактированием одной строки. В ALT Linux рекомендуется добавлять описания зон в конфигурационный файл `/var/lib/bind/etc/local.conf`.

Прямая зона нужна для преобразования доменного имени в IP-адрес — операции, необходимой многим программам постоянно. Большинство записей в прямой зоне — типа A (Address) — предназначены именно для этого. Другие часто встречающиеся типы записей — это CNAME (Canonical Name, **настоящее имя**), позволяющий привязать несколько дополнительных имён к одному, и MX (Mail eXchange, обмен почтой), указывающий, куда пересылать почтовые сообщения, в поле «адресат» которых встречается определённое доменное имя. Вот пример прямой зоны для воображаемого домена internal.domain.net (незначащие поля соответствующих файлов заменены на «. . .»):

```
# cat /var/lib/bind/etc/local.conf
 . . .
zone "internal.domain.net" {
        type master;
        file "internal.domain.net";
};
 . . .
# cat /var/lib/bind/zone/internal.domain.net
$TTL    1D
@       IN      SOA     server root.server (
                                2004082202      ; serial
                                12H             ; refresh
                                1H              ; retry
                                1W              ; expire
                                1H              ; ncache
                        )
        IN      NS      server
                MX 10   server
server          A       10.10.10.1
www             CNAME   server
mail            CNAME   server
jack            A       10.10.10.100
jill            A       10.10.10.101
```

В этом примере совпадение имени файла и имени зоны случайно, но осмысленно — так проще разбираться с содержимым каталога **/var/lib/bind/zone**. Используются правила **умолчания**: если в записи некоторое поле опущено, оно наследуется от предыдущей. Так, вместо «A» можно было бы всюду написать «IN A», а вместо «MX» — «@ IN MX» («@» означает имя домена, указанное в конфигурационном файле). Как видно из пример, всю работу в сети делает компьютер с адресом 10.10.10.1, он же server.internal.domain.net, он же www.internal.domain.net и mail.internal.domain.net. Несмотря на наличие среди CNAME этого сервера имени «mail», MX-запись указывает всё же не на него, а на действительный адрес — так рекомендовано RFC. Распространённая практика заводить запись типа «`A`» непосредственно на имя *зоны* (т. е. присваивать адрес имени internal.domain.net) приводит, помимо незначительных выгод, к мелким, неочевидным и весьма трудно преодолимым трудностям, которые мы здесь описывать не будем, ограничившись советом *не* использовать этот — пусть и законный — трюк.

Для того чтобы преобразовывать IP-адреса в доменные имена, у каждой сети должна быть **обратная** зона. Если такой зоны нет, и в файле `/etc/hosts` тоже ничего не написано, операция не выполнится. Такое преобразование нужно гораздо реже и в основном по соображениям административным: для того, чтобы выяснить принадлежность компьютера (с которого, допустим, пытаются атаковать сервер) по его IP-адресу. Некоторые почтовые серверы проверяют, содержится ли IP-адрес машины, передающей сообщение, в обратной зоне и похоже ли полученное доменное имя на то, что указано в сообщении, и при несовпадении отказываются принимать письмо. К сожалению, поскольку неудобства, связанные с отсутствием обратной зоны, понятны лишь грамотному администратору, а таковых на просторах Интернета не слишком много, обратные зоны во множестве сетей либо отсутствуют, либо дают неверную информацию. В случае внутренней сети обратная зона необязательна, но желательна — для простоты администрирования и удовлетворения потребностей разных программных продуктов, которые ею пользуются.

Обратная зона состоит почти целиком из записей типа «`PTR`» (Pointer, указатель). Чтобы не умножать сущностей, решено было не вводить *новый* способ работы сервера имён и представить **обратное** преобразование IP-адреса как **прямое** преобразование доменного имени специального вида. Например, чтобы выяснить доменное имя компьютера с адресом 1.2.3.4, необходимо запросить информацию о **доменном имени** 4.3.2.1.in-addr.arpa. Таким образом, каждой подсети класса «`C`» (или выше) соответствует определённый домен, в котором можно найти ответ. Вот как выглядит обратная зона для нашего воображаемого домена:

```
# cat /var/lib/bind/etc/local.conf
 . . .
zone "12.11.10.in-addr.arpa" {
        type master;
        file "12.11.10.in-addr.arpa";
};
 . . .
```

```
# cat /var/lib/bind/zone/12.11.10.in-addr.arpa
$TTL    1D
@       IN      SOA     server.internal.domain.net.
root.server.internal.domain.net (
                                2004082201      ; serial
                                12H             ; refresh
                                1H              ; retry
                                1W              ; expire
                                1H              ; ncache
                        )
        IN      NS      server.internal.domain.net.
0               PTR     internal.domain.net.
1               PTR     server.internal.domain.net.
100             PTR     jack.internal.domain.net.
101             PTR     jill.internal.domain.net.
```

Обратите внимание, что относительные адреса, использованные в левой части записей PTR, раскрываются в полные вида *адрес*.12.11.10.in-addr.arpa, а в правой части используются полные (которые вполне могут указывать на имена в *разных* доменах).

Проверить синтаксическую правильность конфигурационного файла и файла зоны можно с помощью утилит `named-checkconf` и `named-checkzone`, входящих в пакет bind. Они же используются при запуске службы командой `service bind start`. Однако при интенсивной эксплуатации сервера, когда остановка службы нежелательна, можно попросить демон перечитать файлы настроек или определённые зоны с помощью утилиты `rndc` — вот тогда не стоит забывать о проверке синтаксиса.

Стоит иметь в виду, что, в отличие от прямых зон, обратные *описывают* административную принадлежность компьютеров, но сами принадлежат хозяину *сети* (как правило, провайдеру). Возникает особого рода затруднение, связанное с работой DNS-сервера уже не во внутренней сети, а в сети Интернет. Дело в том, что подсети класса «`C`» (т. н. сети «`/24`», в которых сетевая маска занимает 24 бита, а адрес компьютера — 8) выдаются только организациям, способным такую подсеть освоить (в сети класса «`C`» 254 абонентских IP-адреса, один адрес сети и один широковещательный адрес). Чаще всего выдаются совсем маленькие подсети — от «`/30`» (на два абонентских адреса) до «`/27`» (на 30 адресов) — или другие диапазоны, сетевая маска которых не выровнена по границе байта. Таких подсетей в обратной зоне получится несколько, а возможности просто разделить её, отдав часть адресов в администрирование хозяевам, нет. Грамотный провайдер в таких случаях пользуется RFC2317 (есть в пакете bind-docs), предписывающем в обратной зоне заводить не записи вида «`PTR`», а ссылки «`CNAME`» на адреса в «классифицированных» обратных зонах специального вида. Обратное преобразование становится двухступенчатым, зато администрирование каждой классифицированной зоны можно отдать хозяину.

DNS-сервер, отвечающий на запросы из сети Интернет, должен быть *зарегистрирован* в родительском домене. В самом деле, единственный способ узнать, кто обслуживает домен internal.domain.net — спросить об этом у DNS-сервера домена domain.net. Так что о выделении поддомена необходимо договариваться с провайдером. Регистрацией в доменах *первого* уровня занимаются выделенные организации. Например, домен в зоне .ru необходимо регистрировать в компании АНО «Региональный Сетевой Информационный Центр»[1]"(RU-CENTER).

Правила требуют, чтобы при регистрации домена было указано *не менее двух* DNS-серверов, которые будут его обслуживать. Из всех зарегистрированных серверов (записей типа «`NS`» в родительской зоне) только одна соответствует т. н. **первичному** (master) серверу, а остальные — т. н. **вторичным** (slave). Для внешнего пользователя вторичный сервер не отличается от первичного: он столь же авторитетен в ответе на запрос об именах его домена. Отличия только в способе администрирования. Все изменения вносятся в зоны первичного сервера, а вторичный только кеширует эти зоны, целиком получая их по специальному межсерверному протоколу. Полученная зона складывается в файл, редактировать который бессмысленно: первичный сервер при изменении зоны рассылает всем своим вторичным указание скачать её заново. Право на скачивание зоны можно ограничить настройкой «`allow-transfer`» (как правило в ней перечисляются адреса вторичных серверов). Вот слегка отредактированная цитата из руководства по Bind, описывающая задание вторичного сервера в файле настроек:

```
// We are a slave server for eng.example.com
zone "eng.example.com" {
     type slave;
     file "slave/eng.example.com";
     // IP address of eng.example.com master server
     masters { 192.168.4.12; };
```

[1] `http://www.nic.ru`

```
};
```

Вторичный сервер хорошо размещать где-нибудь *подальше* от первичного, во всяком случае, в другой сети — так повышается надёжность обработки запроса (если один сервер недоступен, возможно, ответит второй) и возрастает скорость распространения записей по кешам промежуточных серверов.

Проверку работоспособности, доступности и вообще самочувствия DNS-сервера лучше всего делать утилитой `dig` из пакета bind-utils. Это очень мощная утилита, выдающая максимум информации о том, что происходило с запросом (запрашивая *обратное* преобразование, не забудьте добавить ключ `-x`). Более лаконична, но не менее мощна утилита `host` из того же пакета. Поминаемую старыми руководствами по UNIX утилиту `nslookup` использовать не рекомендуется, о чём она сама честно предупреждает.

```
# dig jack.internal.domain.net

; <<>> DiG 9.2.4rc5 <<>> jack.internal.domain.net
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32751
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1,
ADDITIONAL: 1

;; QUESTION SECTION:
;jack.internal.domain.net.      IN      A

;; ANSWER SECTION:
jack.internal.domain.net. 86400 IN      A       10.11.12.100

;; AUTHORITY SECTION:
internal.domain.net.    86400   IN      NS
server.internal.domain.net.

;; ADDITIONAL SECTION:
server.internal.domain.net. 86400 IN    A       10.11.12.1

;; Query time: 37 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Aug 22 16:07:17 2004
;; MSG SIZE  rcvd: 95
```

Наконец, для выяснения административной принадлежности тех или иных доменов и сетей можно воспользоваться утилитой `whois`, которая обращается к специальной сетевой базе данных (не имеющей отношения к DNS).

Служба FTP (vsftpd)

Протокол FTP (file transfer protocol, протокол передачи файлов) широко используется для обмена файлами в Интернете и локальных сетях. Это специализированный протокол, который предназначен только для передачи файлов и хорошо приспособлен для выполнения этой задачи. К сожалению, изначально протокол спроектирован таким образом, что пароли, данные и управляющие команды передаются открытым текстом и их можно легко перехватить. Однако это не составляет проблемы при работе с многочисленными серверами, которые предоставляют только анонимный доступ.

В этом документе изложены рекомендации, которые помогут вам правильно настроить FTP-сервер и свести к минимуму возможность атак на вашу систему через этот вид сервиса.

FTP-сервер vsftpd

В состав дистрибутива ALT Linux входит `vsftpd` (Very Secure FTP Daemon) — полнофункциональный FTP-сервер, позволяющий обслуживать как анонимные запросы, так и запросы от пользователей, зарегистрированных на сервере и имеющих полноценный доступ к его ресурсам. Именно `vsftpd` рекомендован разработчиками дистрибутива для использования в качестве FTP-сервера.

Разумеется, слова «very secure» (очень защищённый) в названии сервера не являются гарантией полной безопасности, однако указывают на приоритеты его разработчиков. Они стремились создать как можно более надёжную, аккуратно спроектированную и написанную программу, максимально устойчивую к разного рода атакам. Каждая строка кода неоднократно подвергалась тщательным проверкам со стороны специалистов по безопасности информационных систем.

Однако преимущества `vsftpd`, которым он обязан своей популярностью, не ограничиваются его надёжностью и защищённостью. Это производительный, хорошо масштабируемый FTP-сервер. Демонстрацией его возможностей может служить серверный пул ftp.redhat.com[1], обрабатывающий до «15000» соединений одновременно.

Наконец, важным достоинством сервера являются простота и гибкость настройки. Все необходимые настройки осуществляются посредством редактирования единственного конфигурационного файла `/etc/vsftpd.conf`, который фактически является символической ссылкой на файл `/etc/vsftpd/conf`.

Организация анонимного доступа на основе vsftpd

Если вам необходимо создать анонимный FTP-сервер, вы можете использовать `vsftpd` в сочетании с пакетом `anonftp`. Установки этих двух пакетов достаточно для того, чтобы получить работоспособный сервер. В целях безопасности сервер по умолчанию сконфигурирован именно для предоставления анонимного доступа. Запрещены любые команды записи, а также доступ локально зарегистрированных пользователей.

Для обеспечения надёжности системы архитектура сервера `vsftpd` позволяет устанавливать соединения от имени специально указанного непривилегированного пользователя, который определяется директивой `nopriv_user` в конфигурационном файле. Для того, чтобы риск был минимальным, этот пользователь должен обладать как можно меньшими привилегиями. С этой целью при установке `vsftpd` в системе автоматически создаётся учётная запись псевдопользователя novsftpd. Это регистрационное имя не должно использоваться кем-либо для входа в систему, поэтому реальный пароль для него не задаётся. Вместо командного интерпретатора указывается `/dev/null`.

При установке пакета `anonftp` автоматически создаётся каталог, который будет корневым при анонимном подключении, — `/var/ftp` с необходимыми правами доступа. Владельцем этого каталога является пользователь root, а не псевдопользователь, от имени которого работает `vsftpd`. Это сделано для обеспечения безопасности FTP-сервера и системы в целом. Группой-владельцем каталога является специальная группа ftpadmin, предназначенная для администраторов FTP-сервера.

Если вы хотите создать в области для анонимного доступа дерево каталогов, начните с каталога `/var/ftp/pub`. Этот каталог традиционно используется для размещения общедоступных файлов. Для него следует установить права доступа «2775». При этом анонимным поль-

[1] `ftp://ftp.redhat.com`

зователям FTP-сервера будет предоставлен доступ на чтение к файлам, находящимся в каталоге. Владельцем каталога сделайте root. В качестве группы, которой принадлежит `/var/ftp/pub`, целесообразно назначить ftpadmin, включив в неё пользователей, которым необходимо изменять содержимое каталогов FTP-сервера (не стоит работать с содержимым от имени root).

Чтобы разрешить анонимным пользователям вашего сервера доступ на запись, создайте каталог `/var/ftp/incoming` с правами доступа «`3773`» (владелец — ftpadmin, группа-владелец — ftpadmin), тем самым предоставив анонимным пользователям право записи в этот каталог, но лишив их возможности просмотра его содержимого. О том, какие изменения в конфигурации сервера должны быть сделаны для того, чтобы разрешить запись, рассказано ниже в этой главе.

Внимание

Наличие каталога, открытого для анонимной записи по протоколу FTP, делает возможным злонамеренное или случайное переполнение диска данными, что может привести к нарушению работы системы в целом. Для предотвращения подобных атак и недоразумений старайтесь размещать каталог, открытый для записи, на отдельном разделе файловой системы.

Доступ к серверу зарегистрированных пользователей

Чтобы предоставить доступ к FTP-серверу для локально зарегистрированных пользователей, необходимо внести изменения в конфигурационный файл `/etc/vsftpd.conf`. Для этого достаточно удалить знак комментария перед директивой «`local_enable=YES`». В такой конфигурации клиенты FTP-сервера получат доступ к любым каталогам файловой системы, для которых такой доступ разрешён исходя из прав соответствующих локальных пользователей. Это могут быть как домашние каталоги пользователей, так и системные каталоги. Если в настройках `vsftpd` разрешена запись (см. ниже), клиенты получат и все права на запись, которыми располагают эти пользователи.

Сервер `vsftpd` позволяет ограничить возможность пользователей, зарегистрированных локально, перемещаться по дереву каталогов. При этом процесс, работающий с клиентом, будет выполняться в изолированной среде (chrooted environment), и пользователь будет иметь доступ лишь к своему домашнему каталогу и его подкаталогам. Чтобы ограничить таким образом доступ к каталогам для отдельных пользователей, удалите знаки комментариев у следующих строк в конфигурационном файле:

```
chroot_list_enable=YES
chroot_list_file=/etc/vsftpd/chroot_list
```

В файле `/etc/vsftpd/chroot_list` перечислите регистрационные имена пользователей, для которых должна использоваться изолированная среда выполнения. Вы можете использовать для этого и другой файл, указав его имя в строке `chroot_list_file` конфигурационного файла.

Чтобы ограничить доступ к дереву каталогов для всех пользователей, зарегистрированных локально, добавьте в конфигурационный файл директиву «`chroot_local_user=YES`».

В этом случае имена пользователей, перечисленные в файле `/etc/vsftpd/chroot_list` (при условии, что у строк, указанных выше, удалены знаки комментария), имеют противоположное действие. Для них не используется изолированная среда выполнения, и перемещение по файловой иерархии не ограничивается домашним каталогом.

Чтобы запретить анонимный доступ к FTP-серверу, поставьте знак комментария в начале строки «`anonymous_enable=YES`» в конфигурационном файле.

Дополнительные сведения о настройке сервера

Сервер `vsftpd` способен осуществлять всю передачу данных в пассивном режиме, что сопряжено со значительно меньшим риском, однако не всегда удобно. Чтобы разрешить использование только пассивного режима, достаточно удалить символ комментария у директивы «`port_enable=NO`» в конфигурационном файле.

Чтобы разрешить запись файлов на сервер, удалите знак комментария у директивы «`write_enable=YES`». Этого достаточно для того, чтобы пользователи, зарегистрированные локально, получили возможность загружать файлы в те каталоги, для которых они располагают правами на запись. Чтобы разрешить запись файлов анонимным пользователям, необходимо, кроме этого, удалить знак комментария у строки «`anon_upload_enable=YES`». Кроме того, специальный непривилегированный пользователь, используемый для работы с анонимными клиентами, должен иметь права на запись в один или несколько каталогов, доступных таким клиентам.

Для получения дополнительной информации о настройке FTP-сервера `vsftpd` и параметрах конфигурационного файла обратитесь к странице руководства `vsftpd.conf`.

Многие параметры использования `vsftpd`, в том числе относящиеся к безопасности, могут быть заданы при помощи `xinetd` (демона Интернет-служб). В частности, этот сервер позволяет ограничить количество одновременно выполняемых процессов как по системе в целом, так и для каждого отдельного пользователя, указать пользователя, от имени которого будет выполняться служба, задать приоритет процесса (nice), указать адреса, с которых разрешено подключение к данной службе, а также время доступа и множество других параметров. Вот пример файла конфигурации `xinetd` для `vsftpd`:

```
# default: off
# description: The vsftpd FTP server.
service ftp
{
        disable = no # включает службу
        socket_type = stream
        protocol = tcp
        wait = no
        user = root
        nice = 10
        rlimit_as = 16M # устанавливает лимит адресного
пространства
        server = /usr/sbin/vsftpd # путь к исполняемому файлу
        only_from = 192.168.0.0 # предоставляем доступ из всей
подсети 192.168.0
        only_from = 207.46.197.100 207.46.197.101 # доступ с
указанных адресов
#       only_from = 0.0.0.0 # неограниченный по адресам доступ
        access_times = 2:00-9:00 12:00-24:00 # время, когда
возможен доступ
}
```

Для получения дополнительной информации по использованию `xinetd` обратитесь к страницам руководства `xinetd` и `xinetd.conf`.

Электронная почта (postfix)

Сервер электронной почты postfix позволяет организовать обмен электронной почтой с Интернет и внутри локальной сети. postfix рекомендуется к установке в любой конфигурации ALT Linux. Это объясняется тем, что в UNIX-подобных системах возможность отправлять почту с помощью вызова команды из командой оболочки практически обязательна. Некоторые программы (например, служба исполнения заданий по расписанию crond) пользуются этим для отправки сообщений пользователям.

Рекомендуется настраивать систему таким образом, чтобы доставкой всей электронной почты, проходящей через машину, занималась служба MTA (Mail Transport Agent), в нашем случае — postfix. Хотя многие почтовые программы способны отправлять сообщения на удалённый SMTP-сервер, имеет смысл поручить и эту задачу системному серверу электронной почты, чтобы не было необходимости следить за отсылкой отправленных писем.

Помимо postfix существуют и другие популярные реализации MTA, например, qmail, exim и даже ветеран Интернета Sendmail, хотя последний сейчас используется всё реже, поскольку неоправданно сложен в настройке. Однако по разным причинам, включая соображения безопасности, в любом дистрибутиве ALT Linux по умолчанию в качестве MTA предлагается postfix.

В данном руководстве мы ограничимся рекомендациями по настройке postfix для нескольких типичных ситуаций. Более полные сведения можно получить из превосходной документации на английском языке, которая входит в состав пакета postfix, документация в формате HTML размещена в отдельном пакете — postfix-html.

Работа в режиме SMTP-сервера

Обратите внимание, что по умолчанию после установки postfix не работает в режиме SMTP-сервера. Для того, чтобы *принимать* сообщения по протоколу SMTP или ESMTP, как извне так и локально, необходимо переключить службу postfix в режим работы `server` командой `control postfix server`. По умолчанию после установки служба postfix находится в режиме `local`, в котором postfix не принимает соединений из сети, ограничиваясь приёмом локальных соединений посредством сокетов семейства UNIX (UNIX-domain socket). Для большинства рабочих станций работа в режиме SMTP-сервера не требуется (и нежелательна с точки зрения безопасности), а в качестве SMTP-сервера должен использоваться какой-то внешний узел сети.

Пакеты postfix

Базовый RPM-пакет для установки сервера postfix в ALT Linux носит, как нетрудно догадаться, имя postfix. Ряд дополнений, расширяющих возможности postfix, выделены в отдельные пакеты, полный список которых можно получить командой `apt-cache search ^postfix -`.

Конфигурационные файлы

Файлы настройки postfix располагаются в каталоге `/etc/postfix`. Основные параметры определяются в файле `main.cf`; в частности, все параметры, о которых говорится далее в этом разделе, устанавливаются именно в этом файле, остальные случаи оговариваются специально.

Для простоты ориентации в настройке postfix в файле `main.cf` указываются только параметры, выставленные администратором системы, плюс некоторые из значений по умолчанию, которые администратору с большой вероятностью нужно будет изменить. Значения по умолчанию для всех остальных параметров перечислены в файле `main.cf.default` (этот файл не следует редактировать, он служит только для справок). Довольно полный список параметров с развёрнутыми комментариями и примерами приведён в файле `main.cf.dist`, а наиболее полную информацию по всем возможным параметрам следует искать в руководстве `postconf(5)`.

Обратите внимание, что если вы изменили конфигурацию при запущенной службе postfix, новые настройки нужно активизировать командой `service postfix reload`.

Доменная информация

Имя хоста и домена, которые считаются локальными при обработке email-адресов, необходимы для функционирования почтового сервера. Если эти имена для postfix должны отличаться от того, что выдаёт команда `hostname`, установите их с помощью параметров `myhostname` и `mydomain`.

Postfix на узлах с удалённым доступом к сети

Существует несколько проблем, возникающих при попытке отправки исходящей почты с машин, которые не являются полноценными узлами интернет, например, в системах с модемным и другими непостоянными соединениями не всегда возможно немедленно отправить сообщения удалённым адресатам по SMTP и их приходится держать в очереди до тех пор, пока соединение не будет установлено. Для этого используется параметр `defer_transports`, например:

```
defer_transports = smtp
```

Доставка активизируется командой `/usr/sbin/sendmail -q`, которая в ALT Linux исполняется автоматически при установке PPP-соединения.

Как любой полноценный MTA, postfix позволяет доставлять сообщения, связываясь напрямую с сервером, обслуживающим получателя письма. Тем не менее, для dialup-машин непосредственная доставка сообщений нежелательна, поскольку время соединения ограничено. К тому же это излюбленная тактика распространителей спама, поэтому многие почтовые серверы сверяют IP-адрес отправителя с базой известных адресов провайдерских пулов, после чего сообщения с таких адресов отвергаются. Поэтому целесообразно доверить доставку исходящей почты SMTP-серверу провайдера. Этим управляет параметр `relayhost`, например:

```
relayhost = [smtp.provider.net]
```

Postfix на клиентской машине локальной сети

Рабочие станции в локальной сети или машины в провайдерской сети, отделённой от Интернета с помощью межсетевого экрана/NAT, должны переправлять исходящую почту на почтовый сервер, обслуживающий данную сеть. Для этого также используется параметр `relayhost`, описанный выше. Если сервер задан IP-адресом, можно отключить использование DNS для ускорения работы:

```
disable_dns_lookups = yes
```

Для того чтобы в доменной части адреса отправителя фигурировал домен сети, а не имя конкретной машины в этой сети, установите параметр `myorigin` равным имени домена:

```
myorigin = $mydomain
```

Почтовый сервер для небольших доменов и сетей

Домены, для которых сервер получает почту, отличные от значения `mydomain` и не сконфигурированные как виртуальные домены postfix (см. ниже), нужно перечислить с помощью параметра `mydestination`, либо в дополнительном файле, на который ссылается этот параметр. Аналогичным образом параметр `mynetworks` описывает блоки IP-адресов, которые считаются внутренними и с которых разрешён приём исходящих сообщений. Не следует записывать в `mynetworks` блоки адресов, не принадлежащих сети, которую обслуживает сервер, поскольку этим могут воспользоваться распространители спама.

Для пользователей, желающих отправлять сообщения через данный сервер, когда они находятся вне пределов локальной сети, следует организовать SMTP-аутентификацию. В postfix для этой задачи следует использовать библиотеку SASL (начиная с версии 2.3 в postfix появилась возможность использовать разные реализации SASL). На текущий момент доступно две реализации: Cyrus (пакет postfix-cyrus) или Dovecot (пакет postfix-dovecot). Cyrus, кроме того, можно использовать в postfix для аутентификации в качестве клиента на удалённых SMTP-серверах.

Для шифрования соединений в этой схеме используется протокол SSL/TLS, для включения поддержки которого в postfix необходимо установить пакет postfix-tls. Используя postfix-tls следует принимать во внимание, что это расширение включает в postfix значительный объём исходных текстов, в меньшей степени проверенного в плане безопасности. Настройка аутентификации посредством TLS/SASL описана в файлах `TSL_README` и `SASL_README` в документации postfix.

Преобразование глобальных почтовых адресов в локальные адреса назначения устанавливается с помощью таблиц типа «`virtual`» (см. virtual(5)):

```
virtual_alias_maps = cdb:/etc/postfix/virtual
```

Пример содержимого `/etc/postfix/virtual`:

```
    domain1.ru # Домен в стиле postfix (текст здесь
игнорируется)
    name1@domain1.ru user1
    name2@domain2.ru user2@otherbox
    @domain2.ru user3
```

После редактирования не забудьте активизировать изменения командой `service postfix reload`.

Если каким-либо пользователям сети почта должна доставляться по SMTP на их рабочие станции (это предполагает, что на их машинах работают MTA), подставляйте в доменной части их адресов имена машин в таблицах «`virtual`» либо «`aliases`» (см. ниже).

Алиасы и преобразования адресов

Имена локальных адресатов либо совпадают с именами учётных записей пользователей системы, либо подставляются из таблицы «**aliases**» (см. aliases(5)):

```
alias_maps = cdb:/etc/postfix/aliases
alias_database = cdb:/etc/postfix/aliases
```

При установке postfix «с нуля» в этой таблице создаётся алиас на имя root: вся корреспонденция, предназначенная администратору и поступающая на другие системные адреса, будет доставляться на имя реального пользователя, который осуществляет функции администратора. Изначально им становится первый зарегистрированный в системе реальный пользователь. Таблица алиасов отличается от остальных таблиц, используемых postfix; имена в поле слева (псевдонимы) отделяются от значений в поле справа (адреса доставки) двоеточиями. Адресаты в списке перечисляются через запятую. В качестве способа доставки можно использовать почтовые адреса, команды (обозначаются символом «|» в начале правой части; сообщение подаётся на стандартный поток ввода команды) и имена файлов:

```
John.Smith: john
chief: chief@bosscomputer
trio: stock, hausen, walkman
robot: | /usr/bin/robot --process-mail
filebox: /dir/file
```

Рабочий образ таблицы строится с помощью команды **newaliases**, а также при актуализации всех изменений посредством команды **service postfix reload**.

При отправке сообщения postfix формирует адрес отправителя автоматически из имени учётной записи пользователя и значения собственного домена (или значения **myorigin**, если этот параметр выставлен). Даже если почтовый клиент выставил заголовок «**From:**», этот адрес попадает в служебные заголовки сообщения и может быть использован получателем, что не всегда желательно. Преобразование адресов отправителей к глобальным адресам можно задать в таблице типа «**canonical**» (см. canonical(5)):

```
sender_canonical_maps = cdb:/etc/postfix/sender_canonical
```

Аналогичная таблица «**recipient_canonical**» и соответствующий параметр **recipient_canonical_maps** могут быть использованы для преобразования адресов назначения. Для актуализации изменений таблиц используйте команду `service postfix reload`.

Защита от нежелательной корреспонденции

Противодействие **спаму** (массовым рассылкам непрошеной корреспонденции) — отдельная большая тема, которую невозможно полностью раскрыть в этом руководстве; здесь даны лишь несколько практических советов применительно к конфигурации postfix. По умолчанию сервер сконфигурирован так, что отвергает попытки переслать сообщения извне на другие удалённые серверы. Со спамом, адресованным локальным получателям, дело обстоит сложнее.

В postfix предусмотрены возможности передачи почтовых сообщений на проверку на спам как внутренним фильтрам, так и внешним программам. Подробности о фильтрации почты см. в `FILTER_README` в документации postfix Режим фильтрации почты включается в ALT Linux командой `control postfix filter`.

Один из возможных способов — опираться на данные служб, организованных по принципу «чёрного списка» IP-адресов (MAPS RBL и ей подобные). Чтобы задействовать эти сервисы, предварительно ознакомившись с условиями их использования, занесите имена доменов, работающих по принципу RBL, в конфигурацию:

```
      smtpd_client_restrictions = permit_mynetworks,
reject_maps_rbl
      maps_rbl_domains = relays.ordb.org,
blackholes.mail-abuse.org
```

В некоторых случаях требуется адресная работа с отдельными нарушителями почтового этикета. Адресная работа заключается в блокировании SMTP-соединений с их адресов, сетей либо доменов. Для этого предусмотрены таблицы типа «`access`» (см. access(5)):

```
      smtpd_client_restrictions = permit_mynetworks,
cdb:/etc/postfix/access
```

Пример таблицы:

```
1.2.3.4 550 No more canned meat, please
1.2.5 REJECT
goodguy.generallybad.com OK
.generallybad.com REJECT
```

Как и с другими таблицами, закончив редактирование, приведите новые настройки в действие командой `service postfix reload`.

Прочие настройки

По умолчанию размер файла почтового ящика при локальной доставке ограничен «51200000» байтами. Это ограничение можно изменить с помощью параметра `mailbox_size_limit`. Установка параметра в «0» снимает ограничение.

Использование Postfix

После того, как postfix настроен и запущен как служба с предсказуемым именем `postfix`, в настройках почтовых клиентов можно указывать имя или адрес машины (например, localhost) в качестве SMTP-сервера. Для получения почты с удалённых серверов можно использовать программу fetchmail, которая работает в связке с postfix, опрашивая внешние почтовые ящики пользователей по протоколам POP3 или IMAP и передавая полученные сообщения системному MTA для локальной доставки. Подробнее о работе с fetchmail см. в документации соответствующего пакета. Файлы журналов postfix находятся в каталоге `/var/log/mail`.

Кеширующий прокси-сервер (Squid)

Централизованный выход в Интернет через один сервер в сети (proxy)

В корпоративных сетях довольно обычна ситуация, когда, с одной стороны, множество пользователей на разных компьютерах пользуются ресурсами сети Интернет, при этом обеспечить надлежащий уровень безопасности на этих компьютерах одновременно довольно сложно. Ещё сложнее заставить пользователей соблюдать некий «корпоративный стандарт», ограничивающий возможности использования Интернет (например, запретить использование определённого типа ресурсов или закрыть доступ к некоторым адресам).

Простейшим решением будет разрешить только определённые методы доступа к Интернет (например, по протоколам HTTP и FTP) и определить права доступа абонентов на *одном* сервере, а самим абонентам разрешить только обращение к этому серверу по специальному прокси-протоколу (поддерживается всеми современными броузерами). Сервер же, после определения прав доступа, будет транслировать (проксировать) приходящие на него HTTP-запросы, направляя их адресату. Допустим, несколько пользователей с нескольких компьютеров внутренней сети просматривают некоторый сайт. С точки зрения этого сайта их активность представляется потоком запросов от одного и того же компьютера.

Обычный доступ

Непосредственно после установки Squid уже поддерживает функции прокси, но принимает соединения только с того компьютера, на который установлен. Для того, чтобы можно было воспользоваться этим сервисом с других компьютеров, необходимо изменить т. н. **таблицы управления доступом** (Access Control Lists, далее ACL), находящиеся, как и большинство настроек Squid, в файле `/etc/squid/squid.conf`. Файл этот снабжён подробными комментариями и примерами по всем настройкам в виде комментариев, например, так:

```
# TAG: некоторая_настройка
```

Таким же образом там описаны все значения, выставленные по умолчанию. В частности, для того, чтобы Squid принимал соединения из всей внутренней сети, необходимо раскомментировать две строки и подставить в них действительный диапазон адресов вашей сети:

```
acl our_networks src 192.168.1.0/24 192.168.2.0/24
http_access allow our_networks
```

Следует иметь в виду, что при обработке запроса на доступ к Squid все строки «`http_access`» файла `squid.conf` просматриваются последовательно сверху вниз до первой строки, соответствующей параметрам запроса. Это означает, что важен *порядок* следования строк «`http_access`».

Анонимизирующий доступ

При использовании прокси-сервера остаётся некоторая угроза безопасности, связанная с тем, что топология внутренней сети и активность её абонентов не маскируются. Например, в протоколе HTTP используются т. н. заголовки (headers), значения которых заполняются броузерами при отправлении запроса. Squid умеет удалять опасные поля заголовка из запроса при помощи настройки «`header_access`» или подменять их значения на другие, заранее заданные, при помощи настройки «`header_replace`». Варианты использования этих настроек приведены в `squid.conf` в районе «`TAG: header_access`» и «`TAG: header_replace`» соответственно. Имейте в виду, что из-за удаления и/или подмены полей заголовков может нарушиться связь с некоторыми системами, использующими значение этих полей для организации взаимодействия и авторизации.

«Прозрачный» доступ

Можно вообще не сообщать пользователям сети, что в ней используется прокси-сервер. Если прокси-сервер — одновременно и маршрутизатор, весь сетевой трафик *в любом случае* его не обойдёт. При этом можно средствами межсетевого экрана все обычные HTTP-запросы к удалённым серверам перенаправлять на вход особым образом настроенного Squid. С внешней стороны это будет выглядеть обычным прокси-сервером, а со стороны пользователя — ничем не будет отличаться от обычной работы в сети. Команда `iptables`, перенаправляющая HTTP-запросы к внешним серверам на порт Squid может, например, выглядеть так:

```
# iptables -t NAT -A PREROUTING -d ! адрес_самого_сервера \
        -i внутренний_сетевой_интерфейс -p tcp -m tcp --dport 80 \
-j REDIRECT --to-ports 3128
```

Или так:

```
# iptables -t nat -A PREROUTING -p tcp -d 0/0 --dport www \
-i внутренний_сетевой_интерфейс -j DNAT \
--to локальный_адрес_на_котором_слушает_прокси:3128
```

Настройка `squid.conf` при этом использует обратное проксирование, описанное ниже. Как сказано в документации по Squid, для организации прозрачного прокси достаточно добавить в `squid.conf` строку `http_port 80 transparent`

Фильтрация доступа

В Squid существует гибкая схема фильтрации внешних ссылок. С её помощью, например, можно закрыть доступ к определённым сайтам и ресурсам на них, избавиться от навязчивой рекламы (banners), ссылок непристойного содержания и т. п. Содержимое фильтруется с помощью всё тех же ACL и настроек «`http_access deny`», примеры которых приведены в `squid.conf`. Обратите внимание, что при задании фильтруемого URL или доменного имени сервера можно использовать регулярные выражения, таким образом в одной строке определяя фильтр для целого класса адресов или доменных имён. Подробнее о регулярных выражениях можно прочитать в руководстве regex(7). Запрет доступа к домену `baddomain.com`, например, можно оформить в виде строк

```
acl Bad dstdomain baddomain.com
http_access deny Bad
```

Заметим также, что «`http_access deny Bad`» следует помещать *перед* строками вида «`http_access allow`», в этом случае доступ к любому сайту домена будет закрыт безусловно.

Авторизация доступа

В Squid есть возможность определять различные ACL разным пользователям и/или категориям пользователей. Если для определения того, какой именно пользователь подключается к серверу, недостаточно IP-адреса его компьютера, следует воспользоваться одной из многих схем **авторизации**, принятых в Squid. Авторизация конфигурируется с помощью тега «`TAG: auth_param`». Посмотреть, какие схемы (программы) авторизации поддерживает Squid, можно в каталоге `/usr/lib/squid`.

Например, настройка авторизации в LDAP может выглядеть так:

```
auth_param basic program /usr/lib/squid/squid_ldap_auth -b
ou=People,dc=office,dc=lan -f (uid=%s) -h ldap.office.lan
auth_param basic children 5
auth_param basic realm Squid proxy-caching web server
auth_param basic credentialsttl 2 hours
```

Информацию по настройке той или иной программы авторизации можно почерпнуть из соответствующих руководств.

Локальное хранение данных, которые уже запрашивались пользователями (cache)

Не менее важное свойство Squid состоит в том, что данные, полученные по запросам из сети Интернета, сохраняются в системе (**кешируются**). При повторных запросах данные будут предоставляться не из Интернет, а из сохранённой копии. Объекты кешируются до тех пор, пока не истечёт их «срок годности» или пока они не будут вытеснены другими, более часто используемыми. Например, если все пользователи внутренней сети одновременно работают с одним и тем же сайтом в сети, общий поток запросов от Squid будет несравненно меньше потока пользовательских запросов, так как большинство ресурсов сайта уже будет находиться в кеше.

Обычное кеширование

Непосредственно после установки сервера он уже выполняет кеширующие функции. Кеширование позволяет не только сэкономить на оплате трафика, но и уменьшить время доступа к ресурсам Интернет. Однако следует понимать, что снижение внешнего трафика возможно только если один и тот же ресурс был запрошен несколькими пользователями на протяжении некоторого промежутка времени. Если пользовательские запросы почти не пересекаются, снижение трафика может быть незначительным.

Кроме того, кеширование неэффективно в ситуации, когда повторный запрос на большинство объектов приходит уже после того, как эти объекты вытесняются из кеша более новыми. Если анализ статистики показывает, что происходит именно это, можно попробовать увеличить объём кешируемой информации. Настройки Squid, отвечающие за размер кеша, приведены в файле `squid.conf` в разделе, начинающемся словами «`OPTIONS WHICH AFFECT THE CACHE SIZE`».

Наконец, кеширование входного потока данных делает невозможной «справедливую» оплату различными пользователями их *собственного* трафика. Например, если один пользователь посещает некие сайты по совету другого, то его трафик будет по большей части не выходить за пределы сервера, на котором ресурсы этих сайтов уже — по милости первого пользователя — закешированы.

Выборочное кеширование

Некоторые данные (например, очень большие файлы, автоматически изменяемые WWW-страницы, звуки и т. п.) кешировать невыгодно: вероятность повторного запроса в течение «срока годности» низкая, а других объектов вытесняется много. С другой стороны, содержимое некоторых сайтов может понадобиться кешировать в обязательном порядке (например, для ускорения доступа). Эти свойства управляются, как обычно, при помощи ACL и настроек «`always` `_direct`» (без кеширования) и «`never_direct`» (обязательное кеширование). Например, чтобы предотвратить кеширование файлов, получаемых по протоколу FTP (это, как правило, разумно), необходимо в соответствующем месте `squid.conf` раскомментировать строки

```
acl FTP proto FTP
always_direct allow FTP
```

Иерархия серверов

Если запрашиваемый ресурс не найден в локальном кеше Squid, он может попытаться запросить его у «вышестоящих» серверов или у «соседей» — вместо того, чтобы обращаться в Интернет. Таким вышестоящим сервером (parent peer) может быть сервер провайдера, а соседом (sibling peer) — сервер абонента, подключённого к тому же провайдеру. Правила передачи объектов кеша и формирования иерархии серверов описаны в документации Squid. Раздел конфигурационного файла, отвечающий за механизм обмена кешем, начинается словами «`OPTIONS WHICH AFFECT THE NEIGHBOR SELECTION ALGORITHM`».

Необходимо знать, что обмен содержимым кеша требует непременной авторизации доступа между серверами (во избежание подлогов и другого вида атак). Все параметры соединения сервера с сервером (настройка «`cache_peer`») записываются в текстовом виде, включая пароли, так что следует строго ограничить доступ к файлу настройки `squid.conf`.

Обратное проксирование и кеширование внутренних серверов (accelerate)

Кеширующие и проксирующие способности Squid можно использовать и «в обратную сторону», пропуская через сервер *входящие* запросы на *внутренние* серверы. Таким образом можно скрыть реальное расположение и структуру серверов, и уменьшить нагрузку на них.

Обычное обратное проксирование (один сервер)

В режиме «`accelerate`» сервер сам принимает извне HTTP-запросы, адресованные, как правило на 80-й порт. Кроме того, необходимо указать имя сервера и порт, на который будут проксироваться запросы. Это можно сделать, например, так

```
http_port 80 defaultsite=internal.www.com
cache_peer 192.168.1.1 parent 80 0 no-query originserver
```

Если необходимо как-то ограничить доступ к внутреннему серверу, это легко сделать, применив соответствующие ACL.

Множественное проксирование

Для обратного проксирования *нескольких* внутренних серверов необходимо, чтобы внешние запросы на сайты с *разными* доменными именами попадали на вход Squid, который бы ставил в соответствие каждому имени действительный адрес сервера во внутренней сети и перенаправлял бы запрос туда. Делается это с помощью механизма **виртуальных хостов**. Вот как, например, можно организовать прокси для двух серверов, www1.foo.bar и www2.foo.bar, адреса которых в DNS указывают на машину со Squid-сервером (файл `/etc/squid/squid.conf`):

```
http_port 80 defaultsite=www1.foo.bar vhost
hosts_file /etc/hosts
```

Настройка «`defaultsite`» нужна серверу для заполнения HTTP-заголовков. Соответствие доменных имён адресам серверов во внутренней сети разумно получать из стандартного файла `/etc/hosts`:

```
10.0.0.1    www1.foo.bar
10.0.0.2    www2.foo.bar
```

Множественное обратное проксирование имеет важный побочный эффект. В сетях, подключённых к Интернету постоянно, нередки случаи, когда недобросовестные администраторы внутренних сайтов регистрируют их в других доменах и используют их в качестве хостинг-платформ, часто с документами незаконного или нецензурного содержания. Допустим, сайт www1.foo.bar также зарегистрирован как www.verycoolwarez.com, и по этому адресу доступны контрафактные версии программ и т. п. В обычном случае для обнаружения этого факта необходимо анализировать трафик, делать внушение администратору www1.foo.bar и т. д. При использовании обратного проксирования такого вообще не может произойти без договорённости с администратором Squid, то есть без занесения в `/etc/hosts`!

Разное

Безопасность

Squid поддерживает проксирование разнообразных протоколов, из которых разрешать надо далеко не всё и далеко не всем. Необходимо, чтобы воспользоваться этим сервисом могли *только* те, кто имеет на это право. Если ваш прокси-сервер — анонимизирующий, им могут воспользоваться, чтобы скрыть обратный адрес при незаконном доступе к данным или даже атаке на другие серверы.

Неаккуратно настроенный прокси-сервер можно использовать для массовой рассылки почты (спама), что осуждается правилами пользования сетью Интернет и может привести к отказу пользователей Сети и администраторов почтовых серверов принимать почту с вашего домена.

Сбор статистики и ограничение полосы доступа

В Squid входит утилита кеш-менеджер, служащая для отображения статистики и загрузки сервера. Кеш-менеджер представляет собой cgi-приложение и должен выполняться под управлением сконфигурированного HTTP-сервера. Все настройки кеш-менеджера также производятся в `/etc/squid/squid.conf`, строки, которые относятся к кеш-менеджеру, обычно включают «`cachemgr`».

При помощи Squid можно ограничить **полосу пропускания** («толщину канала») для пользователей. За это отвечают параметры «`delay_pools`» и «`delay_class`». Эта технология позволит вам снизить загрузку канала, например, большими по объёму медиафайлами, и отдать полосу какому-то более приоритетному трафику.

Кеширование DNS-запросов

Squid содержит встроенный минисервер запросов DNS. Он выступает как посредник между Squid и внешними DNS-серверами. При запуске Squid производит начальное тестирование доступности DNS и интернет в целом. Это можно отключить, используя опцию `-D`. Время кеширования удачного DNS-запроса по умолчанию составляет шесть часов.

Ссылки

`squid.opennet.ru`[1]

Наиболее полный информационный ресурс, посвящённый Squid на русском языке.

`www.squid-cache.org`[2]

Официальный сайт разработчиков Squid. Содержит огромный FAQ, документацию, списки рассылок, и, скорее всего, ответы на все ваши вопросы.

`www.opennet.ru`[3]

Запрос по ключевому слову «`squid`» в поисковой системе этого сайта выдаст вам наиболее популярные типовые решения задач, которые возникают у администраторов.

Подсистема печати (CUPS)

Технологии печати в Linux

Какие виды принтеров бывают?

Инженерная мысль не останавливается ни на секунду; с момента появления первого печатающего устройства опробована и введена в эксплуатацию масса новых технологий. На данный момент наиболее распространены следующие технологии печати:

`Матричные принтеры`

Матричные принтеры постепенно доживают свой век, но, обладая удивительной простотой и надёжностью, продолжают ещё использоваться. С настройкой этого типа принтеров, за редкими исключениями, проблем не должно возникать.

`Лазерные принтеры`

Лазерные принтеры обладают высокой скоростью печати, незаменимы в офисах организаций всех типов. Моделей существует множество, но, к счастью, для большинства можно найти «родной» драйвер или подобрать совместимый.

`Струйные принтеры`

Главный лозунг струйных принтеров «дёшево и в цвете». Так как в этой индустрии не появилось явного лидера, то процветает множество стандартов и существует вероятность иметь очень экзотический с точки зрения Linux принтер.

`GDI- или Windows-принтеры`

«Дёшево любой ценой». Используют компьютер вместо того, чтобы всё делать самим. Проблемы очень вероятны, так как протокол работы зачастую закрыт. К счастью, существует несколько моделей, напоминающих по интерфейсу полноценные принтеры, что позволяет всё-таки производить печать с низким разрешением.

Где находится драйвер принтера?

Если вам доводилось работать в Microsoft Windows, то вы прекрасно знаете, что в комплекте с каждым принтером поставляется дискета или компакт-диск с драйвером для настройки печати. К великому сожалению, в мире UNIX/Linux пока так и не договорились о едином стандарте драйверов. В настоящее время стандартом де-факто является пакет ghostscript фирмы *Aladdin Software*.

Что такое ghostscript?

Ghostscript (далее GS) — это программа перевода из векторного формата в растровый (RIP). На вход программы подаётся документ в формате PostScript или PDF, а на выходе получается документ на языке, понятном конкретной модели принтера или графическое изображение страницы. Большинство программ Linux формируют документ в формате PostScript, поэтому оказывается, что применения GS вполне достаточно для обеспечения печати из приложений.

Что такое PostScript?

PostScript — это самый настоящий язык программирования, который используется для того, чтобы описать содержимое страницы. Например, пишутся такие серии команд: переместиться туда-то, напечатать слово такое-то, сменить шрифт на такой-то. Существуют модели принтеров, непосредственно понимающие PostScript, для всех остальных требуется посредник, и GS успешно с этим справляется. Вы спросите, как он поддерживает столько принтеров? Дело в том, что количество языков принтеров намного меньше количества существующих моделей. Например, известные всем HP LaserJet «4», «4L», «5», «5L», «6», «6L», «1100», «2100» «понимают» один и тот же язык PCL5. Если вы не страшитесь этих слов и вам любопытно посмотреть, какие драйверы языков поддерживает ваш GS — дайте команду `gs --help`. Так, например, ljet4 как раз и обслуживает вышеупомянутую линейку моделей принтеров.

Итак, теперь понятно, как устроена печать в любом дистрибутиве Linux: программа формирует документ на языке PostScript, GS переводит его на язык принтера, установленного в вашей системе, и передаёт эстафету принтеру.

Что такое спулер?

Но это ещё далеко не всё. А что если вам надо послать документ на принтер, установленный на другой машине, или напечатать сразу несколько документов и нет времени ждать, пока каждый из них будет обслужен ghostscript и уступит место следующему? Для этого существует так называемый сервер печати или **спулер** (от англ. spooler), обслуживающий **очереди печати**. Последний термин более точно отражает суть, поэтому им мы и будем пользоваться и писать дальше просто «спулер». В ALT Linux в качестве спулера работает современная система печати CUPS (Common UNIX Printing System), которая является фактическим стандартом среди спулеров.

Как настроить принтер?

В ALT Linux у пользователя есть несколько способов настроить систему печати. В большинстве ситуации проще и надёжнее всего воспользоваться стандартными средствами CUPS. Есть также альтернативные способы добавления и удаления принтеров, например, средствами foomatic. Дальше будет рассказано обо всём этом подробнее.

Настройка системы CUPS

Настройка CUPS через web-интерфейс

Запустите свой любимый броузер и откройте страницу `http://localhost:631` — вы увидите интерфейс настройки CUPS. Выберите раздел «Manage Printers», далее раздел «Add Printer». Затем у вас спросят имя и пароль администратора CUPS; по умолчанию это администратор системы (root).

Появится первый диалог — достаточно указать имя (Name) принтера, описание (Description) носит вспомогательный характер, а местоположение (Location) нужно только в том случае, если у вас настроена сложная система разграничения доступа в CUPS. В следующем диалоге производится выбор устройства, к которому подключён принтер. Два последующих — выбор модели и производителя принтера. Напоследок можно заказать печать пробной страницы.

Может возникнуть ситуация, что сочетания производитель/модель вашего принтера не окажется в списке, предложенном CUPS — это означает, что в базе данных CUPS не нашлось описания для вашей модели принтера. В этот момент отчаиваться не следует.

Описание принтера в CUPS — это файл `модель_принтера.ppd`, PPD-файлы для известных моделей CUPS держит в каталоге `/usr/share/cups/model`, имеющийся там список принтеров далеко не исчерпывающий. Однако познакомить CUPS с новым принтером очень легко, если принтер поддерживает один из стандартных языков управления и будет работать с одним из уже существующих драйверов (что почти всегда так). Для этого достаточно найти PPD-файл для нужной модели принтера и поместить его в зону видимости CUPS. Большую базу описаний принтеров предоставляет foomatic (ниже написано о том, как им воспользоваться), если нужной модели не нашлось и там, стоит поискать PPD-файл на сайте производителя принтера.

Настройка CUPS через foomatic

Сложно дать краткое описание того, что же такое foomatic. Это и база описаний моделей принтеров, и интерфейс настройки одновременно. Более того, foomatic умеет настраивать практически все известные на сегодняшний день спулеры, в частности CUPS.

Процедуру настройки лучше всего рассмотреть на конкретном примере[1]. Пусть мы хотим настроить printer HP LaserJet 1022 (которого не нашлось в базе известных принтеров CUPS, расположенный на Windows-машине *host* в рабочей группе *group* под именем *printer*.

[1]Скорее всего, в вашей системе уже установлены компоненты foomatic; если нет, то потребуются пакеты foomatic, foomatic-db, foomatic-db-engine, foomatic-filters, hplip-foomatic.

Прежде всего, нужно найти описание для данной модели принтера в базе данных foomatic:

```
$ foomatic-ppdfile -A|grep 1022
HP LaserJet 1022 Id='HP-LaserJet_1022' Driver='hpijs'
Ricoh Aficio 1022 PS Id='Ricoh-Aficio_1022_PS' Driver='Postscript'
```

Чтобы не просматривать огромный список принтеров, лучше всего профильтровать вывод `foomatic-ppdfile -A` утилитой `grep`, задав в качестве поискового выражения название/номер модели принтера.

Нас прежде всего интересует идентификатор-описание в поле `Id`. Кроме того, мы получили имя ghostscript-драйвера, с которым будет работать данный принтер.

Теперь нужно сгенерировать ppd-файл:

```
$ foomatic-ppdfile -p 'HP-LaserJet_1022' > HP-LaserJet_1022.ppd
```

Дальше можно пойти двумя путями. Первый способ — поместить этот файл в каталог `/usr/share/cups/model` и перезапустить службу CUPS. С этого момента принтер попал в список известных CUPS — можно возвращаться к web-интерфейсу CUPS, где появится нужная модель, и настраивать принтер привычным способом. Второй способ — использовать возможности foomatic для настройки спулера описан ниже. Копировать ppd-файл вручную при этом не потребуется.

1) Сформируем URI, описывающее расположение принтера. Общий формат: «`replace{протокол}://replace{местоположение}`». Местоположение зависит от протокола. Если принтер подключён локально, то URI — «`file://replace{путь_к_файлу}`», в частности может быть указан файл-устройство (например, `/dev/lp0`. Если удалённый сервер работает под управлением Microsoft Windows (наш случай), то URI — `smb://user:password@group/host/printer`. Некоторые поля в описании могут отсутствовать. Наконец, в случае Netware-сервера URI — `ncp://user:password@server/printer`.

2) Идентификатор принтера и имя подходящего драйвера мы уже выяснили ранее.

3) Мы собрали теперь достаточно информации для того чтобы произвести конфигурацию принтера. Это делается одной командой `foomatic-configure`. Мы укажем желаемый спулер (параметр `-s`), имя принтера (параметр `-n`) идентификатор описания, драйвер и URI.

   ```
   # foomatic-configure -s cups -n Laser_Jet -p HP-LaserJet_1022
   \
   ```

```
-d hpijs -c smb://user:password@group/host/printer
```

Вот и готово. Настройки можете посмотреть непосредственно в конфигурационных файлах спулера или при помощи команды `foomatic -configure -Q`.

```
<queues>
<defaultqueue>Laser_Jet</defaultqueue>
<queue foomatic="0" spooler="cups">
  <name>Laser_Jet</name>
  <printer>HP-LaserJet_1022</printer>
  <driver>hpijs</driver>
  <connect>smb://user:password@group/host/printer</connect>
  <description>HP LaserJet 1022</description>
</queue>
</queues>
```

Как удалить лишние принтеры?

Если вы слишком увлеклись настройками и теперь путаетесь между десятком заведённых принтеров, то самое время удалить лишние.

Удаление можно производить вручную, удалив описания из конфигурационных файлов спулера (`/etc/cups/printers.conf`). Для CUPS можно вновь воспользоваться web-интерфейсом. Если вы использовали foomatic, то полезно знать команду

```
# foomatic-configure -s тип_спулера -n имя_принтера -R
```

где *тип_спулера* — CUPS.

Конфигурационные файлы CUPS

Конфигурационные файлы CUPS находятся в каталоге `/etc/cups`. Файл `/etc/cups/cupsd.conf` содержит описание главных параметров сервера, каждый из которых сопровождается подробным описанием. Приведём некоторые наиболее часто используемые параметры:

`LogLevel`

Уровень подробности протоколирования. По умолчанию значение равно «`info`». Если у вас какие-то проблемы с CUPS, а в протоколе нет ничего информативного, можете поднять уровень до максимального — «`debug2`».

`Port`

TCP-порт, по которому сервер будет ожидать обращений клиентов. По умолчанию это «`631`» (зарезервированное для протокола IPP).

`Location, Order, Allow, Deny`

Серия директив, аналогичных имеющимся в web-сервере Apache. «`Order`» — порядок просмотра значений «`Allow`» и «`Deny`», «`Allow`» — адреса, с которых разрешён доступ, «`Deny`» — адреса, с которых доступ запрещён. Пример:

```
<Location />
Order Deny,Allow
Deny From All
Allow From 127.0.0.1
</Location>
```

Это означает, что сначала сервер будет смотреть, запрещён ли доступ с машины (здесь запрещено для всех), а потом — разрешён ли (здесь разрешён доступ только с самого сервера). Последний параметр понадобится для организации одного CUPS-сервера на всю локальную сеть. Организация доступа может быть устроена сколь угодно сложно благодаря поддержке концепции классов.

Файл `/etc/cups/client.conf` содержит настройки для клиентской части. В нём указываются всего два параметра — местоположение сервера и защищённость соединения. Скорее всего, вам не придётся там что-либо менять.

Файл `/etc/cups/printers.conf` содержит описание принтеров. Формат записи интуитивно понятен и похож на XML. Ниже приведён пример настройки для локального принтера.

```
<DefaultPrinter printer>
DeviceURI parallel:/dev/lp0
State Idle
Accepting Yes
JobSheets none none
QuotaPeriod 0
PageLimit 0
KLimit 0
</Printer>
```

Существенными параметрами являются `Accepting` (принимает ли принтер задания) и `DeviceURI` (специальное описание местоположения принтера). О том как правильно составлять URI, будет рассказано ниже, когда речь пойдёт о foomatic.

Описание каждого *установленного* принтера (соответствующий PPD-файл) находится в `/etc/cups/ppd/имя_принтера.ppd`. Обратите внимание, что *имя_принтера* — это не название модели, а то имя, под которым установлен принтер (которое дал пользователь). PPD-файл представляет собой текстовый конфигурационный файл, в котором описаны свойства принтера. Названия некоторых параметров говорят сами за себя и/или снабжены комментариями, так что настройку параметров печати можно производить и непосредственно редактируя ppd-файл.

При каждом изменении в конфигурационных файлах CUPS следует перезапускать системную службу cups командой `service cups restart`.