bhv

Евгений Андреев

Администрирование Astra Linux

- Переход с Windows на Linux
- Установка системы
- Начальные навыки администрирования
- Настройка и защита сети
- Установка программ
- Настройка Samba, Apache, SSH, VNC и других служб
- Мандатный контроль целостности, мандатное управление доступом
- Системный киоск, учет сменных накопителей
- Служба каталогов Astra Linux Directory

СИСАДМИН
СИСТЕМНЫЙ АДМИНИСТРАТОР

Евгений Андреев

Администрирование Astra Linux

Санкт-Петербург
«БХВ-Петербург»
2024

УДК 004.43
ББК 32.973.26-018.2
А65

Андреев Е. Д.

А65 Администрирование Astra Linux. — СПб.: БХВ-Петербург, 2024. — 400 с.: ил. — (Системный администратор)

ISBN 978-5-9775-1993-9

Рассмотрена установка Astra Linux, настройка после установки, процесс загрузки системы, в том числе система инициализации systemd, основы командной строки, настройка сети, беспроводного и проводного соединения с Интернетом, установка ПО, настройка хранилища, работа с файловой системой, настройка Samba (интеграция с Windows-сетью), Apache (веб-сервер), SSH, VNC (графический удаленный доступ) и других необходимых сетевых служб. Сделан акцент на специфические особенности Astra Linux, а именно: мандатный контроль целостности, мандатное управление доступом, системный киоск, служба каталогов ALD и учет сменных накопителей. Книга написана с учетом запросов системных администраторов, которым поставлена задача перейти с Microsoft Windows на популярный отечественный дистрибутив Astra Linux. При этом специально для читателей, никогда ранее не использовавших Linux, рассмотрены основы работы с Linux.

Для системных администраторов

УДК 004.43
ББК 32.973.26-018.2

Группа подготовки издания:

Руководитель проекта	*Евгений Рыбаков*
Зав. редакцией	*Людмила Гауль*
Редактор	*Григорий Добин*
Компьютерная верстка	*Ольги Сергиенко*
Дизайн серии	*Марины Дамбиевой*
Оформление обложки	*Зои Канторович*

Подписано в печать 10.06.24.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 32,25.
Тираж 1200 экз. Заказ № 9776.
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Отпечатано с готового оригинал-макета
ООО "Принт-М", 142300, М.О., г. Чехов, ул. Полиграфистов, д. 1

ISBN 978-5-9775-1993-9

Оглавление

ЧАСТЬ I

Введение

ASTRA LINUX®

ГЛАВА 1

Linux: основные сведения

- Что такое Linux?
- Архитектура Linux
- Дистрибутивы Linux
- Версии Astra Linux

1.1. Что такое Linux?

В настоящее время мало̀ кто не слышал об операционной системе Linux. Если еще лет 20 назад Linux была для простого пользователя компьютера такой себе диковинкой (хотя появилась в 1992 году, т. е. более 30 лет назад), то сегодня ноутбуком с ОС Linux «на борту» уже никого не удивишь.

Причина тому понятна. Во-первых, за это время Linux стала гораздо более дружественной к пользователю. Ее современные интерфейсы не менее удобны, чем интерфейс Windows, более удобными стали и встроенные в нее пользовательские приложения. И если раньше Linux в основном устанавливалась на серверы, где до сих пор славится своей стабильной работой, то сейчас она уверенно занимает места и на рабочих станциях. И, конечно же, не следует упускать из внимания и тот момент, что Linux — это система с открытым исходным кодом, и многие ее дистрибутивы совершенно бесплатны.

Многие производители ноутбуков раньше продавали свои устройства или с Windows или с FreeDOS (Free Disk Operating System) — простейшей операционной системой, позволяющей работать только с файлами на дисках. Никакого подключения к Интернету, никакого графического интерфейса... Она служит сугубо для предпродажной проверки того, что устройство работоспособно и сможет загрузить операционную систему, выбранную для него пользователем. То есть выбор у пользователя был таким: или переплатить и купить ноутбук с лицензионной Windows, или же купить его дешевле, но без операционной системы. Что делать потом? Пользователь, желающий сэкономить свои кровные (вполне нормальное желание), ранее в такой ситуации скачивал с торрентов пиратскую версию Windows и устанавливал ее на компьютер. Моральные аспекты этого подхода мы

опустим. А вот присутствие на компьютере пиратской версии Windows создавало огромнейшие дыры в безопасности. Бардак, да и только...

Но в последнее время производители стали поставлять на рынок ноутбуки с одним из дистрибутивов Linux (на выбор самого вендора). В результате пользователь получает уже готовое к использованию устройство и пытается разобраться с новой для него операционной системой и привыкнуть к ней. Заменять ее на пиратскую Windows ему уже не хочется — во всяком случае с самого начала. Он может попытаться поработать в Linux, понять, подходит она ему или нет, а уже потом принять решение: оставить Linux или снести ее и установить Windows. Но основной бонус этого подхода в том, что ноутбук с Linux можно использовать сразу же после покупки: смотреть видео, слушать музыку, бороздить просторы Интернета и даже работать с документами MS Office (с помощью офисного пакета LibreOffice).

Так что в последнее время пользователей Linux стало больше, чем вы можете себе представить. И это уже не говоря о бюджетных организациях, которые массово отказываются от Windows в пользу открытой Linux. По большому счету многим кабинетным сотрудникам, постоянно работающим в основном с текстовыми документами, Windows и не нужен. Зачем платить за то, что можно получить бесплатно, при этом не нарушив закон? Выбор очевиден, и он приводит нас к Linux.

1.2. Архитектура Linux

Операционная система Linux состоит из четырех основных компонентов: ядро, библиотеки, оболочка и приложения. Рассмотрим их подробнее.

1.2.1. Ядро

Ядро, как уже было отмечено, является основным компонентом Linux. Именно ядро контролирует работу аппаратных компонентов компьютера (устройств), управляет ресурсами компьютера (в частности, выделением и освобождением памяти), процессами (позволяет добиться их псевдопараллельного выполнения на одноядерном процессоре) и т. д.

Основными компонентами ядра являются:

- планировщик процессов, отвечающий за распределение процессорного времени между всеми процессами.

  Как можно добиться параллельного выполнения двух и более процессов на одноядерном процессоре? Планировщик приостанавливает выполнение одного процесса на какой-то квант времени и позволяет выполняться второму процессу. Затем приостанавливается второй процесс и дает возможность выполняться первому, и так далее, пока один из процессов не прекратит свою работу. Конечно, все это представлено очень упрощенно, поскольку нельзя просто приостановить выполнение инструкций одного процесса и продолжить выполнять инструкции второго процесса. Нужно загрузить контекст второго процесса, т. е. записать в регистры процессора информацию, с которой работал второй процесс, и вы-

полнить много чего еще. Но в общем картина складывается так: один процесс мы приостанавливаем и разрешаем выполняться второму;

- модуль управления памятью, позволяющий распределять память между всеми процессами. Он также выделяет память в области подкачки, если физическая память заканчивается;
- модуль межпроцессного взаимодействия, позволяющий запущенным процессам обмениваться информацией друг с другом;
- виртуальная файловая система, предоставляющая единый интерфейс для работы с файлами и каталогами для всех типов файловых систем.

  Это означает, что вы будете использовать одни и те же системные вызовы, невзирая на используемую файловую систему. Вам не понадобятся отдельные системные вызовы для каждой файловой системы;
- сетевая подсистема, предоставляющая доступ к ресурсам компьютерной сети.

Расширить функциональность ядра можно с помощью модулей. Например, когда вам потребуется добавить поддержку того или иного устройства, вам понадобится драйвер этого устройства. Как правило, в Linux драйверы ядра реализованы в виде модулей ядра. Добавить драйвер можно путем перекомпиляции ядра, включив в состав ядра нужный драйвер (если нужна высокая производительность) или же командой `modprobe`, что позволяет обойтись без перекомпиляции ядра. Обычно пользователи используют второй вариант (`modprobe`), поскольку он самый удобный.

1.2.2. Библиотеки

Как правило, никто не пишет программы, используя непосредственно инструкции самого процессора, — это сложно, долго, приходится писать несколько версий программы под каждую архитектуру процессора и т. д. Ядро предоставляет программам системные вызовы, которые программисты могут использовать, чтобы упростить себе жизнь и ускорить разработку ПО. Например, вы можете использовать системный вызов `open()`, чтобы открыть какой-то файл.

Множество самых разных системных вызовов, предоставляемых ядром, можно разбить на группы:

- системные вызовы для управления файлами;
- системные вызовы для управления процессами и потоками;
- системные вызовы для синхронизации процессов;
- системные вызовы управления памятью;
- сетевые системные вызовы;
- системные вызовы управления устройствами;
- прочие системные вызовы.

Впрочем, использовать системные вызовы не всегда удобно, особенно, когда не очень хочется «изобретать велосипед» заново. В Linux есть набор библиотек, предоставляющий множество полезных функций. Так, библиотека libcrypt предостав-

ляет функции шифрования, libm — множество математических функций, и т. д. Особняком стоит библиотека libc, содержащая фундаментальную систему функций. Практически каждая серьезная программа для Linux не обходится без использования libc.

1.2.3. Оболочка

Оболочка (shell) — это средство для взаимодействия с пользователем и запуска программ. В состав дистрибутива Linux входит несколько оболочек, и пользователь может выбрать ту, которая ему больше нравится. Как правило, по умолчанию используется оболочка bash.

С помощью оболочки можно писать сценарии автоматизации, позволяющие автоматизировать множество самых разных действий по системному администрированию, и не только. Даже если вы пока не особо владеете языком командной оболочки, в Интернете вы сможете найти множество готовых сценариев.

Конечно же, в современных дистрибутивах, кроме текстовой оболочки, есть еще и графическая. Графическая оболочка, используемая по умолчанию в Astra Linux, называется Fly. Она предоставляет удобный графический интерфейс, задача которого сводится к тому же — к запуску программ.

1.2.4. Приложения

Приложения — это программы, с которыми взаимодействуют пользователи. В состав Linux входят тысячи программ: от самых простых команд вроде `ls` (вывод каталога) или `cp` (копирование файлов) до графических комбайнов, таких как офисный пакет LibreOffice.

Приложения запускаются оболочкой (текстовой или графической). Далее посредством системных вызовов и библиотечных функций приложения обращаются к ядру с просьбой выполнить какую-то функцию. Ядро выполняет ее и возвращает ответ. Если приложение использовало системный вызов, то ответ придет в само приложение, если же библиотечную функцию, то результат будет возвращен в эту функцию, а она уже обработает его и что-то передаст/не передаст (зависит от реализации этой функции) в приложение.

1.3. Дистрибутивы Linux

Прежде всего, нужно пояснить, что такое дистрибутивы. Дистрибутив — это набор, компонентов операционной системы, включающий в себя ядро и программы. С 1992 года было создано огромное количество самых разных дистрибутивов Linux. Некоторые дожили до наших дней, некоторые — нет.

О дистрибутивах Linux можно было бы рассказать еще очень много. Однако важно запомнить следующее:

- основные дистрибутивы — это Red Hat (сейчас существует в виде RHEL, Red Hat Enterprise Linux) и Debian, а все остальные — лишь производные от них.

Так, дистрибутивы Astra Linux и Ubuntu изначально были основаны на Debian. К числу современных RH-подобных дистрибутивов относятся CentOS, Fedora и openSUSE, к числу современных Debian-подобных — Ubuntu, а также его клоны и всевозможные варианты (Kubuntu, Xubuntu, Mint и т. д.);

- номер версии дистрибутива не совпадает с номером ядра — это принципиально разные вещи;
- самыми популярными дистрибутивами на сегодняшний день считаются Ubuntu и Fedora — для настольного применения, а также CentOS и Debian — для серверного.

Все современные дистрибутивы состоят из ядра, установщика (инсталлятора) и набора программ. Набор программ от дистрибутива к дистрибутиву может различаться, и наиболее заметное визуально различие — это графическая среда: GNOME, KDE или Fly (как в Astra Linux). Это то, что бросается в глаза сразу. А вот что касается «подкапотной» начинки, то она практически везде одинаковая: то же ядро (может различаться номер версии), та же система инициализации systemd, тот же загрузчик GRUB2, та же оболочка bash. Разница еще может быть в типе пакетов ПО: RPM или DEB. Дистрибутивы, основанные на Debian, используют формат пакетов DEB. Дистрибутивы, основанные на RedHat, — RPM. Но конечному пользователю, по сути, все равно, какой формат будет у пакетов с ПО. Главное, чтобы в состав дистрибутива входили необходимые программы.

Говорить о выборе дистрибутива уже поздно. Если вы читаете эту книгу, то выбор сделан, и это Astra Linux. Далее мы поговорим о редакциях и версиях этого дистрибутива.

1.4. Версии Astra Linux

Первая версия Astra Linux появилась 30 июня 2010 года. Дистрибутив был основан на Debian 5.0 и практически полностью совместим с ним.

Существуют две редакции дистрибутива: Common Edition и Special Edition. Первая — это дистрибутив общего назначения, не сертифицированная для специального применения версия, которую можно, например, с успехом использовать на домашнем компьютере. Вторая — это сертифицированная операционная система, содержащая встроенные средства защиты. На ее базе можно построить безопасную ИТ-инфтраструктуру любого масштаба. Она идеально подойдет в качестве рабочей станции или сервера в корпоративной среде.

Начиная с версии 1.7, в Astra Linux появились режимы защищенности:

- базовый уровень защищенности «Орел» — базовые средства защищенности (например, системный киоск, описанный в *главе 25*);
- усиленный уровень защищенности «Воронеж» — более защищенный уровень, где по умолчанию включен мандатный контроль целостности (МКЦ), запрет установки бита выполнения, блокировка интепретаторов и т. д.;

- максимальный уровень защищенности «Смоленск» — вдобавок к предыдущему уровню поддерживается мандатное управление доступом (МРД).

Более подробно об уровнях защищенности Astra Linux рассказано в *разд. 2.10*.

Номер версии дистрибутива Astra Linux различается в зависимости от архитектуры процессора и редакции. В табл. 1.1 и 1.2 приведены номера версий Special Edition/Common Edition для архитектуры процессора x86-64. Так и есть — на текущий момент самая актуальная версия ядра находится в Special Edition. Пользователям Common Edition придется довольствоваться старой версией ядра 5.15 (табл. 1.2).

Таблица 1.1. *Версии Special Edition*

Версия дистрибутива	Версия ядра	Дата выпуска
1.7	5.4	27.06.2021
1.7.1	5.10	29.12.2021
1.7.2	5.15	24.08.2022
1.7.3	5.15.0-33	15.11.2022
1.7.4	5.15.0-70	26.04.2023
1.7.5	6.1	16.10.2023
1.8	в разработке	в разработке

Таблица 1.2. *Версии Common Edition*

Версия дистрибутива	Версия ядра	Дата выпуска
2.12.40	5.4	29.12.2022
2.12.43	5.10	07.09.2021
2.12.44	5.10	01.02.2022
2.12.45	5.15	08.08.2022
2.12.46	5.15	17.04.2023

По первым цифрам версии дистрибутива можно понять архитектуру процессора, для которой предназначен дистрибутив. Так, 1 и 2 соответствуют архитектуре x86-64 — наиболее распространенной архитектуре CPU, 3 соответствует IBM System Z, 4 — ARM, 6 — MIPS, 8 — Эльбрус. Разумеется, эти архитектуры менее распространены, и вряд ли вы будете с ними сталкиваться, однако если вам на глаза попадется дистрибутив Astra Linux версии 4.*, то это не более новая версия, а просто версия для архитектуры ARM.

ASTRA LINUX®

ГЛАВА 2

Установка дистрибутива

- Получение дистрибутива
- Подготовка к установке
- Выбор типа установки
- Выбор способа переключения раскладки клавиатуры
- Установка имени компьютера
- Создание локального пользователя
- Установка часового пояса
- Разметка жесткого диска
- Выбор компонентов системы
- Выбор уровня защищенности
- Установка загрузчика GRUB и завершение установки

2.1. Получение дистрибутива

Загружать дистрибутивы (любые) рекомендуется только с их официальных сайтов — во избежание ситуаций, когда в них мог быть встроен сторонний код. Для свободной загрузки дистрибутива Astra Linux доступна только редакция общего назначения — Common Edition.

> ***РЕДАКЦИЯ SPECIAL EDITION***
>
> Редакция Special Edition — платная и доступна только по запросу на сайте **https://astralinux.ru/**. Категорически не рекомендуется загружать Special Edition со сторонних сайтов, если вы действительно заботитесь о безопасности данных.

Дистрибутив редакции Astra Linux Common Edition можно скачать по адресу: **https://dl.astralinux.ru/astra/stable/2.12_x86-64/iso/**. В табличке с файлами, доступными для закачки, вы найдете насколько файлов с расширением iso, — вам нужно скачать любой ISO-файл размером более 5 Гбайт. Ссудя по размеру и дате — всё это файлы с одним и тем же содержимым.

Как уже отмечалось в *главе 1*, обе редакции: Special Edition и Common Edition — поставляются вместе с графическим интерфейсом собственной разработки Fly, поэтому конечный пользователь не заметит разницы между этими двумя редакциями. Что же касается самого интерфейса Fly, то хотя он и выглядит довольно старомодным (как будто бы смотришь на дистрибутив 2010 года), зато удобен в использовании, и к нему можно быстро привыкнуть.

2.2. Подготовка к установке

Установить дистрибутив можно как на физический, так и на виртуальный компьютер. Перед установкой следует подготовить свой компьютер к процессу установки.

- Убедитесь, что на целевом компьютере достаточно свободного пространства. Чтобы полноценно работать с Astra Linux, вам понадобится не менее 30 Гбайт свободного пространства на жестком диске компьютера. Объем оперативной памяти должен составлять не менее 2 Гбайт, но если вы будете пользоваться браузером (а вы будете!), то лучше, чтобы на компьютере было не менее 4 Гбайт ОЗУ.
- Если вы будете устанавливать дистрибутив на компьютер, где уже установлена Windows или другая операционная система, выполните полный бэкап всех важных файлов и желательно всего образа жесткого диска — так процесс восстановления (если что-то пойдет не так) будет более быстрым. Сделать такой бэкап можно с помощью программы Acronis True Image: **https://www.acronis.com/en-us/support/trueimage/2021/**.

  Совет

  Учитывая, что программа Acronis True Image — платная, чтобы не заморачиваться с ее пробными версиями, работающими не более 30 дней и ограниченными в возможностях, можно порекомендовать для этих целей ни в чем ей не уступающую свободно распространяемую программу AOMEI Backupper Standard (**https://www.ubackup.com/download.html**).

- Если вы собираетесь устанавливать дистрибутив в виртуальную машину (для ознакомления с ним или даже для работы — некоторые предпочитают работать именно так), нужно предварительно создать эту виртуальную машину (в *разд. 2.2.2* показано, как это сделать).
- Для установки на физический компьютер надо предварительно создать загрузочную флешку (см. далее).

2.2.1. Подготовка к установке на физический компьютер

Создание загрузочной флешки

Для создания загрузочной флешки рекомендуется использовать утилиту Rufus (рис. 2.1):

1. Загрузите утилиту с официального сайта: **https://rufus.ie/ru/**.
2. Вставьте флешку, на которую вы будете записывать загруженный ISO-образ. Скопируйте с нее все важные файлы (если они там есть), но форматировать флешку не нужно — это будет сделано в процессе записи образа.
3. Запустите утилиту Rufus.
4. Из меню **Устройство** выберите флешку, на которую будет производиться запись. Как правило, приложение автоматически выбирает подключенный к системе USB-накопитель, но лучше убедиться, что выбрано правильное устройство.
5. Нажмите кнопку **Выбрать** для выбора загруженного ISO-образа с дистрибутивом.
6. После выбора ISO-образа все остальные параметры будут установлены автоматически. Нажмите кнопку **Старт** для записи образа на флешку. Флешка будет отформатирована, и на нее записан дистрибутив.

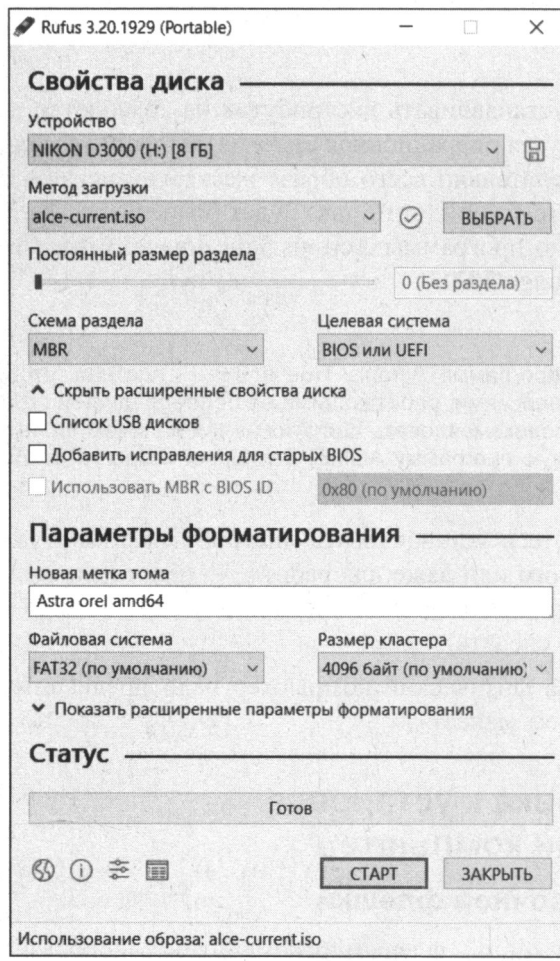

Рис. 2.1. Утилита Rufus

Загрузка дистрибутива с USB-накопителя

Подготовленную флешку самостоятельно подключите к компьютеру, на который будет устанавливаться дистрибутив, перезагрузите компьютер и в настройках BIOS SETUP выберите загрузку с USB-накопителя. Информацию о том, как при загрузке именно вашего компьютера открыть BIOS SETUP, вы найдете в инструкции по его материнской плате или в Интернете. Конкретные действия тут зависят от производителя компьютера, разработчика BIOS SETUP и даже от его версии.

2.2.2. Подготовка к установке на виртуальный компьютер

Подготовка к установке на виртуальный компьютер сводится к созданию виртуальной машины. Процесс создания виртуальной машины здесь будет рассмотрен на примере программы VMware Workstation. Если для виртуализации вы используете другое программное обеспечение, подробные инструкции по работе с ним вы всегда сможете найти в Интернете.

Итак, выполните следующие действия:

1. Запустите VMware Workstation.
2. Выберите команду меню **File | New Virtual Machine**.
3. В открывшемся окне **New Virtual Machine Wizard** нажмите кнопку **Next**.
4. На следующей странице мастера создания виртуальной машины (рис. 2.2) нажмите кнопку **Browse** и выберите образ инсталляционного диска дистрибутива, который вы скачали ранее.

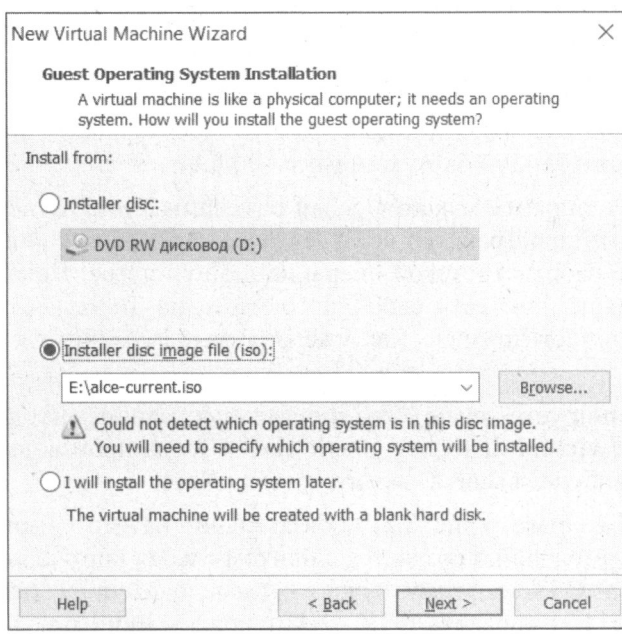

Рис. 2.2. Выбор инсталляционного носителя

Обратите внимание на предупреждение программы, что она не нашла в выбранном вами образе какого-либо дистрибутива ОС. Не придавая значения этому предупреждению, нажмите кнопку **Next**.

5. В открывшемся списке дистрибутивов, известных VMware Workstation, дистрибутива Astra Linux вы не найдете, поэтому выберите **Linux**, а из списка версий — **Ubuntu 64-bit**, как показано на рис. 2.3. Нажмите кнопку **Next**.

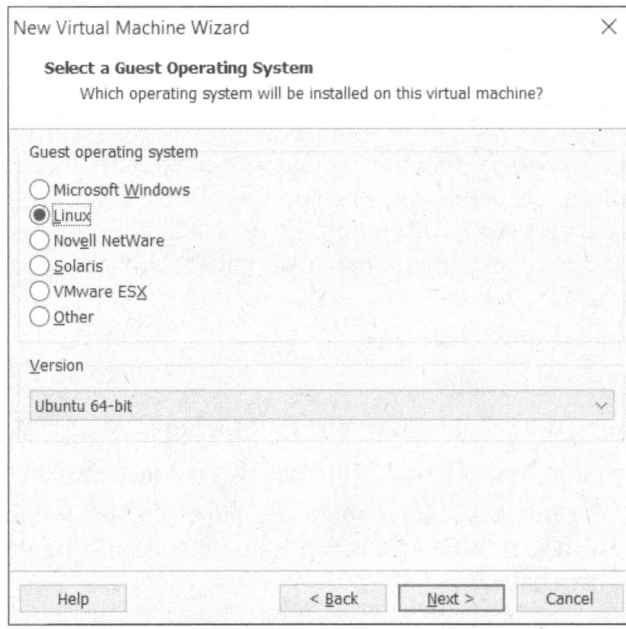

Рис. 2.3. Выбор гостевой операционной системы

6. Следующий шаг очень важен (рис. 2.4) — вам нужно выбрать расположение виртуальной машины. Убедитесь еще раз, что на выбранном диске достаточного свободного места: 20–30 Гбайт, минимум 10 Гбайт.

7. Далее нужно установить максимальный размер накопителя виртуальной машины (рис. 2.5). Это пространство не будет выделено прямо сейчас, а станет выделяться по мере работы гостевой операционной системы. Пока же можно выделить даже больше, чем есть свободного места на диске, — при условии, что в будущем вы очистите диск. Как уже отмечалось, минимум вам понадобится 10 Гбайт.

   Если вы не планируете переносить виртуальную машину на другой компьютер, выберите **Store virtual disks as a single file** — тогда производительность виртуальной машины будет выше. Нажмите кнопку **Next**.

8. На следующем шаге (рис. 2.6) обязательно нажмите кнопку **Customize Hardware**. По умолчанию создается слишком слабая виртуальная машина: одно ядро и 1 ГБайт ОЗУ, поэтому нужно в разделе **Memory** открывшегося окна **Hardware** (рис. 2.7) добавить ОЗУ — установить минимум 2 Гбайт, затем перейти в раздел **Processors** (рис. 2.8) и установить хотя бы 2 ядра.

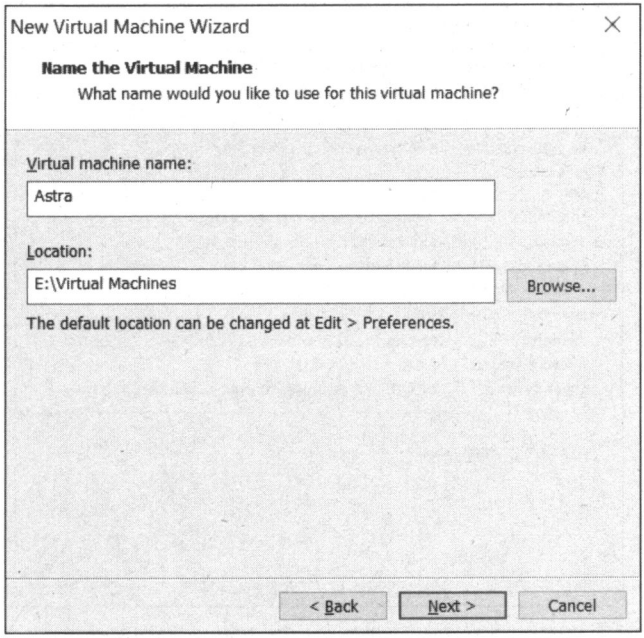

Рис. 2.4. Выбор расположения виртуальной машины

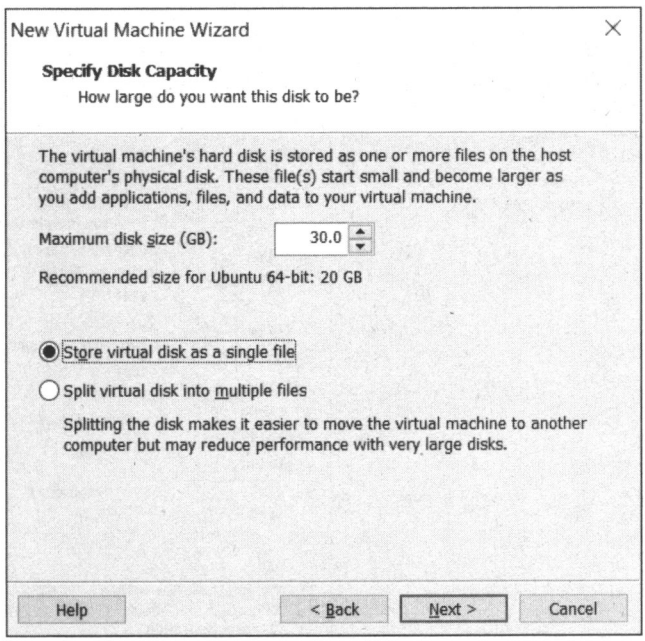

Рис. 2.5. Установка размера накопителя

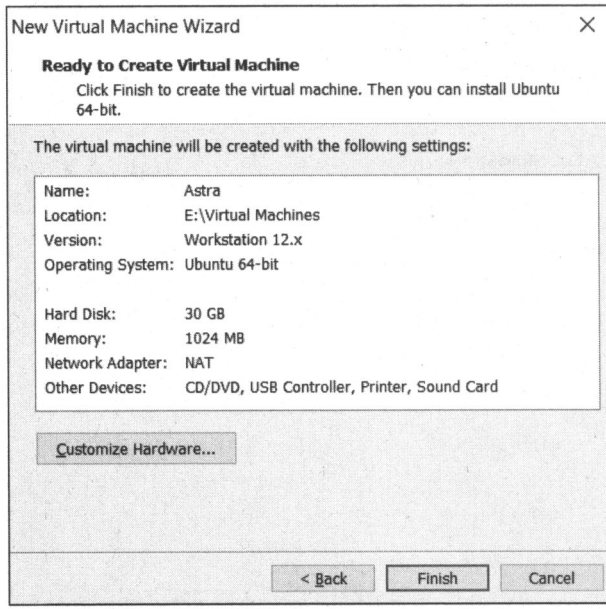

Рис. 2.6. Нажмите кнопку **Customize Hardware**

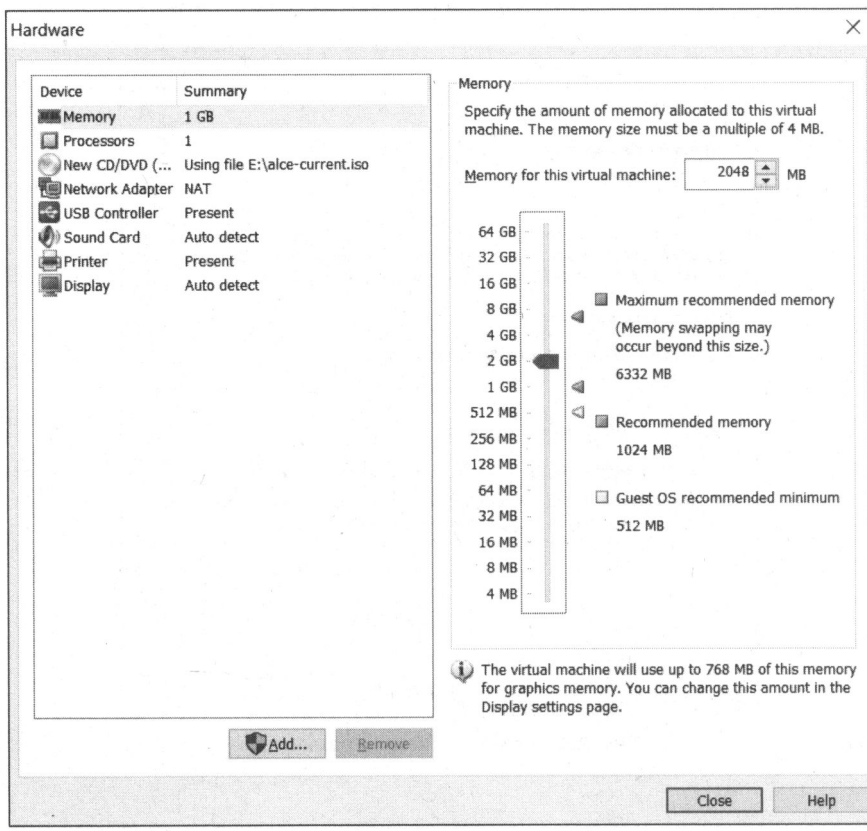

Рис. 2.7. Увеличиваем объем ОЗУ

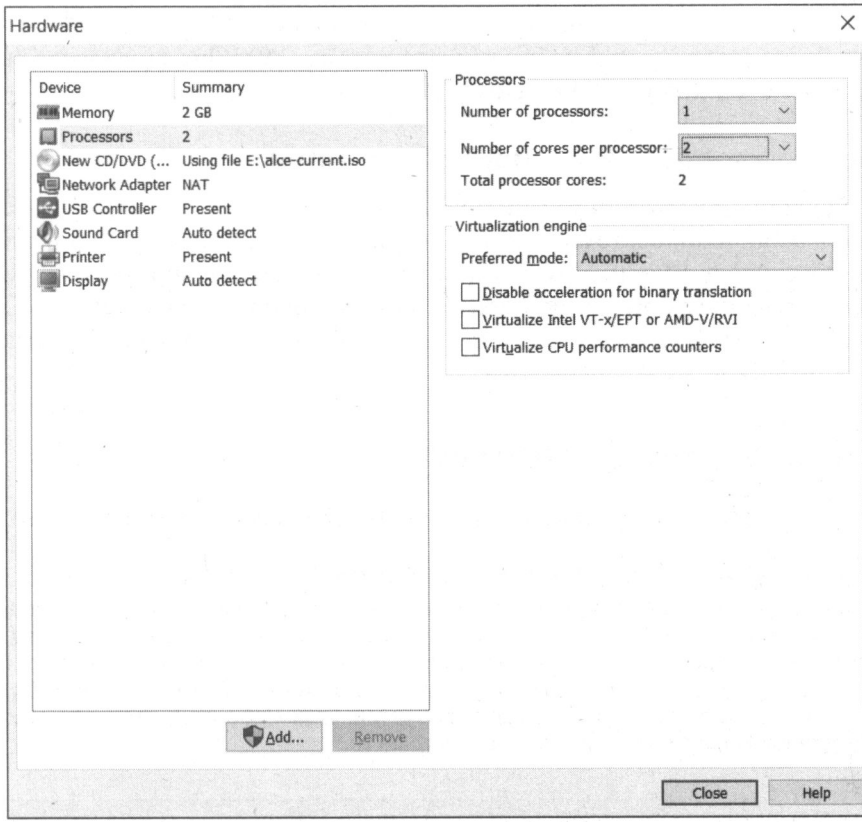

Рис. 2.8. Увеличиваем количество ядер

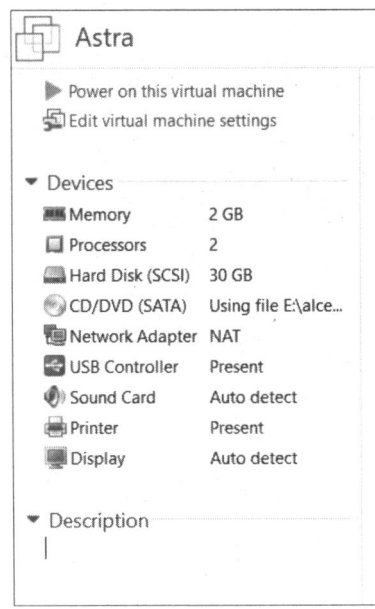

Рис. 2.9. Нажмите кнопку **Power on this virtual machine**

9. Выполнив указанные установки, нажмите кнопку **Close**, а затем — кнопку **Finish**. Виртуальная машина будет создана (рис. 2.9). Для ее запуска нажмите кнопку **Power on this virtual machine**.

Установка гостевой операционной системы

По нажатию кнопки **Power on this virtual machine** (см. рис. 2.9) начнется установка гостевой операционной системы в виртуальную машину, созданную на вашем компьютере. Обратите внимание, что процесс установки ОС на виртуальный компьютер практически ничем не отличается от ее установки на физический. Поэтому далее показан сам этот процесс без уточнения, на какой компьютер (физический или виртуальный) ОС устанавливается.

2.3. Выбор типа установки

После загрузки с инсталляционного носителя вы увидите меню загрузчика (рис. 2.10).

> ***О полиграфическом отображении окон Astra Linux***
>
> К сожалению, некоторые окна Astra Linux предполагают ввод команд и вывод результатов их выполнения белым текстом на обширном черном поле. Полиграфическое отображение скриншотов с таких исходных окон оставляет желать лучшего. Поэтому мы решили здесь и далее размещать иногда в книге инвертированные скриншоты с окон Astra Linux, подрезая их, если требуется, таким образом, чтобы на них оставались только информационно значащие надписи.

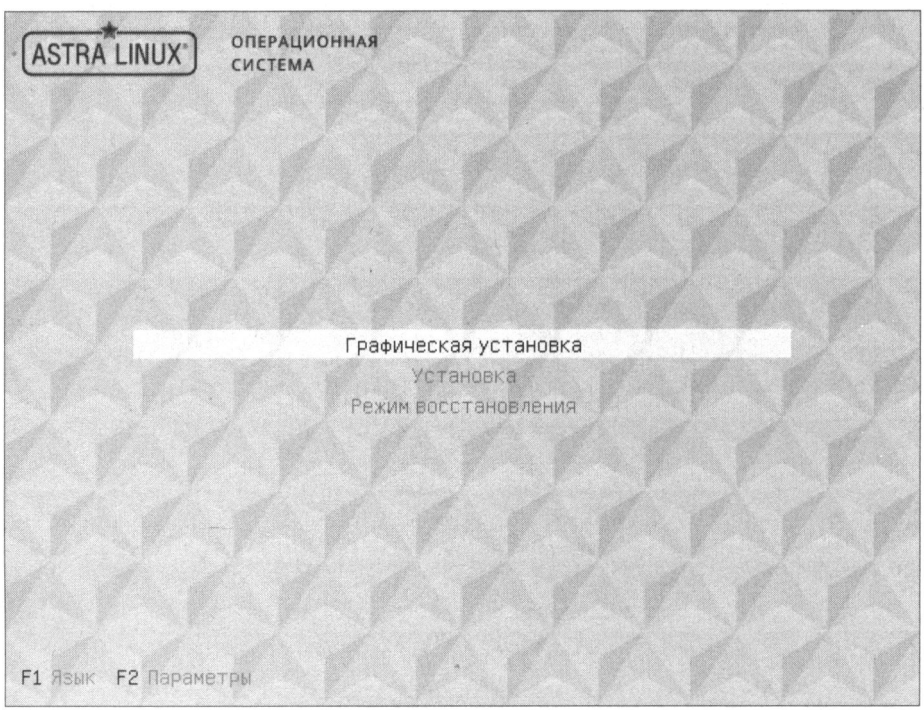

Рис. 2.10. Загрузчик инсталляционного диска

- **Графическая установка** — установка в графическом режиме, ее и нужно выбрать в большинстве случаев;
- **Установка** — установка в текстовом режиме. Этот способ может пригодиться при установке дистрибутива на сервер, где графический интерфейс не нужен;
- **Режим восстановления** — поможет восстановить дистрибутив в случае сбоя. В этом режиме вам будет предоставлена командная оболочка, используя которую вы можете попытаться восстановить «упавшую» систему.

САМОСТОЯТЕЛЬНОЕ УПРАЖНЕНИЕ

Выберите для эксперимента **Режим восстановления** (рис. 2.11). Как можно видеть, программе установки не удалось найти ни одного раздела. Воспользуйтесь предложением режима восстановления и откройте его командную оболочку (рис. 2.12). Введите команду `fdisk -l`, чтобы узнать, какие разделы созданы на подключенных к компьютеру накопителях (если они есть). Подумайте, на какой раздел накопителя вы будете производить установку и какой раздел придется удалить, чтобы образовалось свободное пространство 10–30 Гбайт, достаточное для установки дистрибутива. Разобравшись с работой режима восстановления, выйдите из него, выбрав опцию **<Вернуться>**.

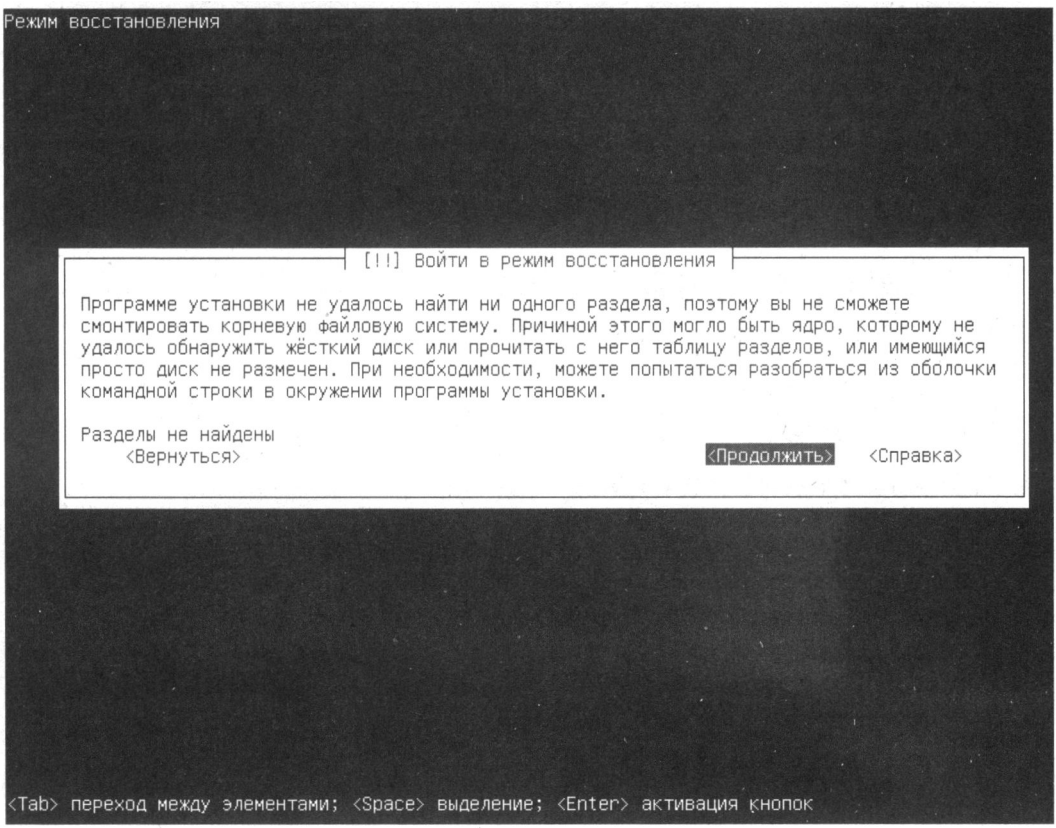

Рис. 2.11. Режим восстановления не нашел разделы и предлагает запустить командную оболочку

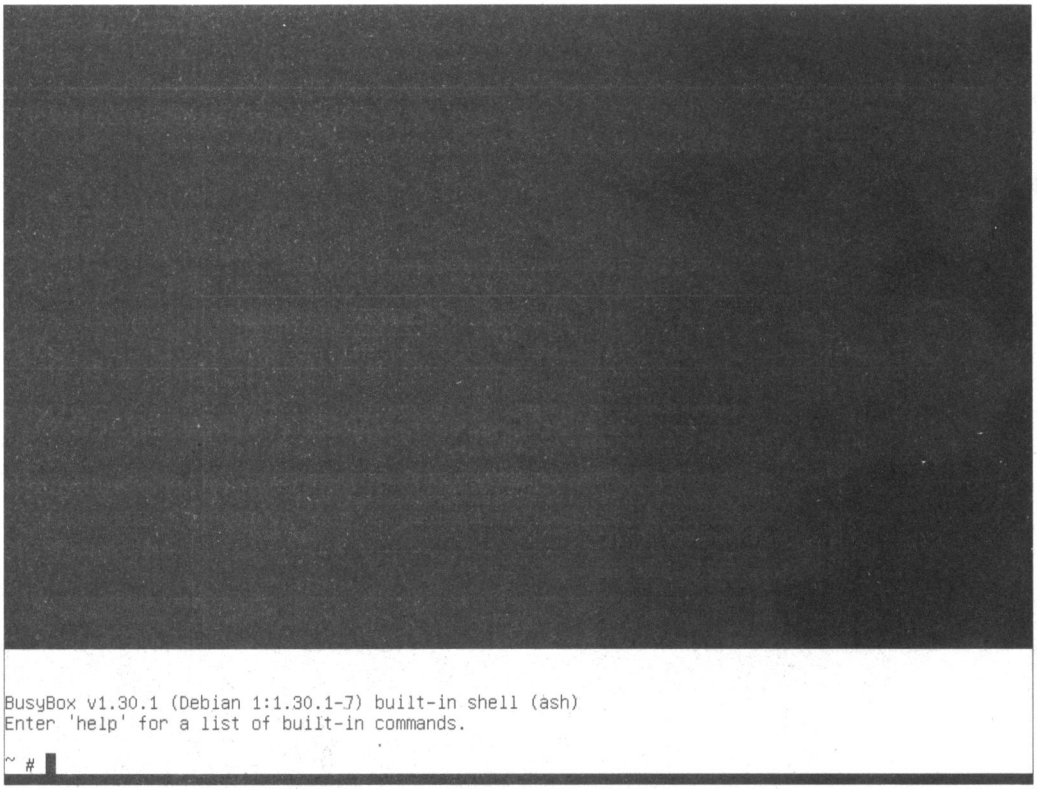

Рис. 2.12. Командная оболочка режима восстановления: здесь можно вводить команды

Вернувшись в меню загрузчика, выберите **Графическая установка** и нажмите клавишу <Enter>. На следующем экране вам нужно прочитать условия лицензионного соглашения и нажать кнопку **Продолжить**.

2.4. Выбор способа переключения раскладки клавиатуры

По умолчанию установщик предлагает комбинацию клавиш <Alt>+<Shift> для переключения языков ввода, но вы можете выбрать любую другую (рис. 2.13). Выберите способ переключения, который вам больше нравится, и нажмите кнопку **Продолжить**.

> ***Примечание***
>
> Все иллюстрации в книге соответствуют версии Astra Linux Special Edition 1.7.5. Установка Common Edition аналогична, разве что вы не сможете выбрать уровень защищенности, вам будет показано немного иначе выглядящее окно выбора дополнительных настроек и не будет запрашиваться настройка пароля для загрузчика GRUB2.

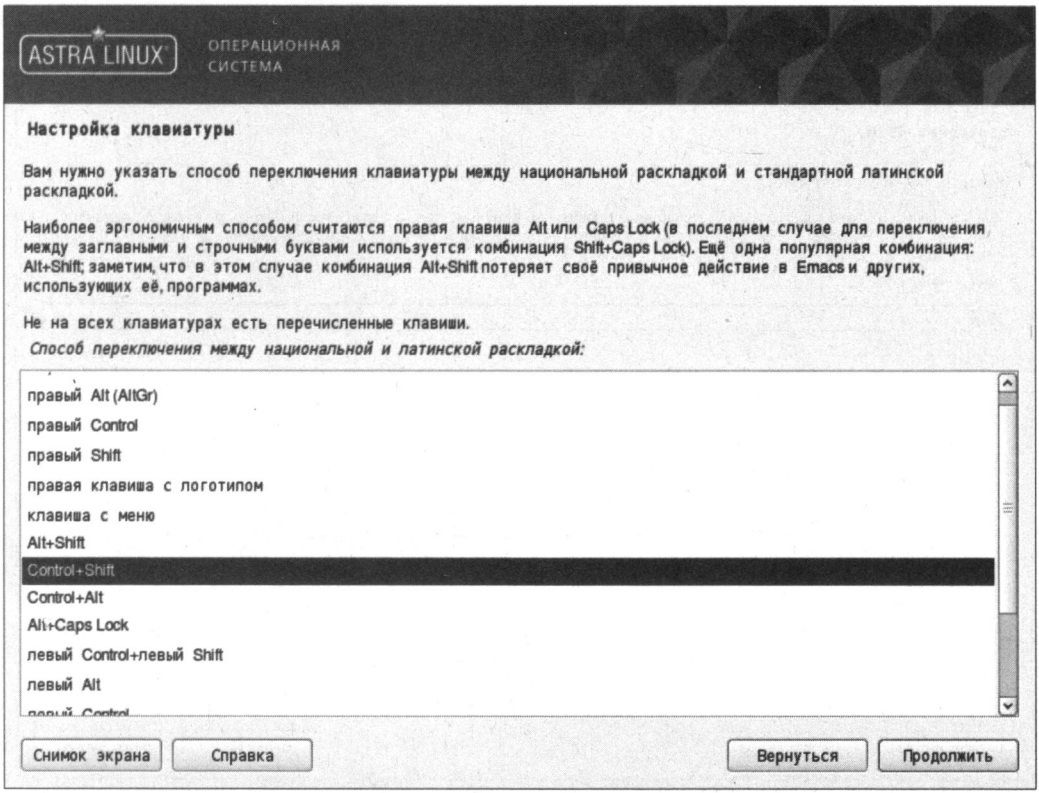

Рис. 2.13. Выбор комбинации клавиш для переключения языков ввода

2.5. Установка имени компьютера

Теперь нужно задать имя компьютера (рис. 2.14), а после этого — имя домена (иллюстрация не приводится). Если вы устанавливаете дистрибутив на домашний компьютер или в виртуальный для ознакомления, можете использовать любое имя. В корпоративной сети, как правило, имя должно быть зарегистрировано в локальной системе DNS, поэтому имя компьютера в этом случае лучше уточнить у администратора сети.

2.6. Создание локального пользователя

Далее надо создать пользователя, от имени которого вы будете входить в систему для повседневной работы. Учтите, что имена root (суперпользователь) и admin (администратор) уже используются, поэтому выберите какое-нибудь другое имя. Создаваемый при установке пользователь считается администратором системы и имеет право выполнять команды с правами root посредством команды `sudo`. Также это имя пользователя будет использоваться для входа в редактор конфигурации GRUB2 при загрузке системы.

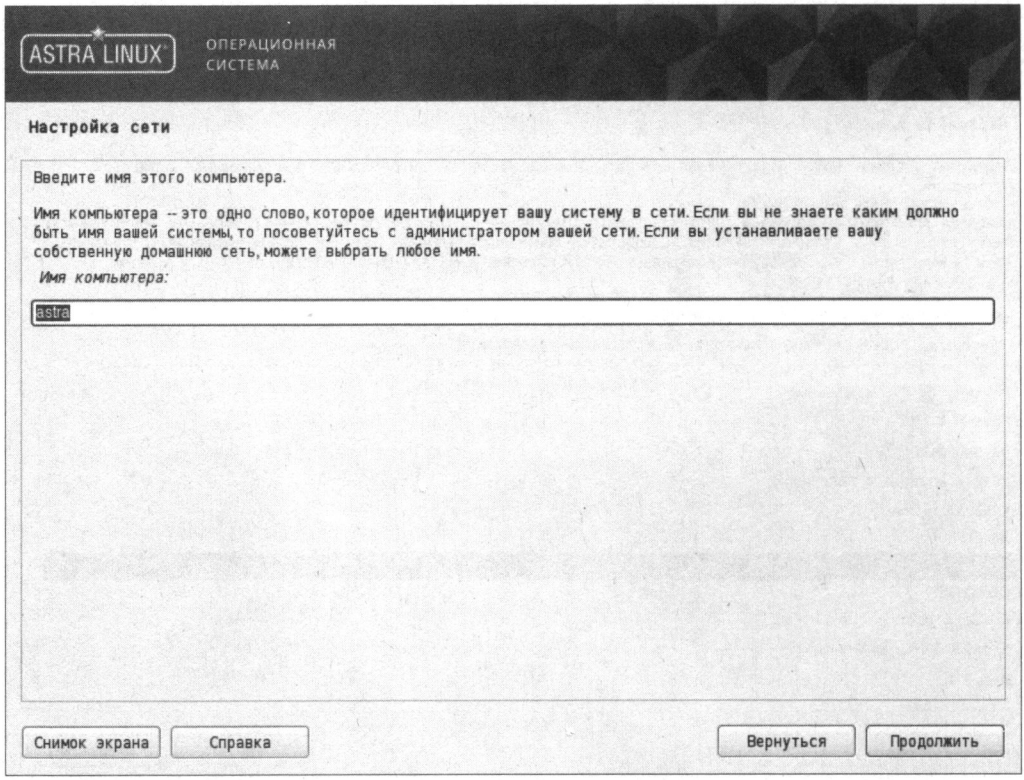

Рис. 2.14. Задание имени компьютера

Итак, выполните следующие действия:

1. Введите имя пользователя и нажмите кнопку **Продолжить** (рис. 2.15).
2. Введите пароль (он должен содержать буквы и цифры — см. далее) и подтвердите правильность ввода путем повторного ввода того же пароля. Нажмите кнопку **Продолжить** (рис. 2.16).

> ***САМОСТОЯТЕЛЬНОЕ УПРАЖНЕНИЕ: ХОРОШИЙ ПАРОЛЬ***
>
> Придумайте себе хорошо запоминающийся, но при этом сложный для подбора пароль. По возможности пароль не должен содержать словарных слов (*словарными* слова называются из-за того, что все они содержатся в орфографических словарях. Пароли из словарных слов легко взламываются простым перебором), должен состоять из символов в обоих регистрах, цифр и различных дополнительных символов: пунктуации, знака подчеркивания и т. п. Можно воспользоваться каким-либо сайтом-генератором паролей (например, **https://passwordsgenerator.net/**) и получить пароль наподобие kMu(e`S:6v+<^D)~@5g_>N, но вряд ли вы его сможете запомнить. Да, он будет очень сложным для подбора, но с первого раза с ним в систему вы не войдете — это точно. Именно поэтому возьмите все-таки какое-нибудь словарное слово, но запишите его при этом с использованием символов в разном регистре, — например, не пишите linux, а пишите: lINuX, добавьте цифры и пару дополнительных символов: lINuX@2024!, и ваш идеальный пароль готов. Его будет относительно сложно подобрать, т. к. он содержит более восьми символов и специальные символы, и в то же время легко запомнить. Не используйте в качестве пароля номер телефона, дату рождения и тому подобную всем известную информацию.

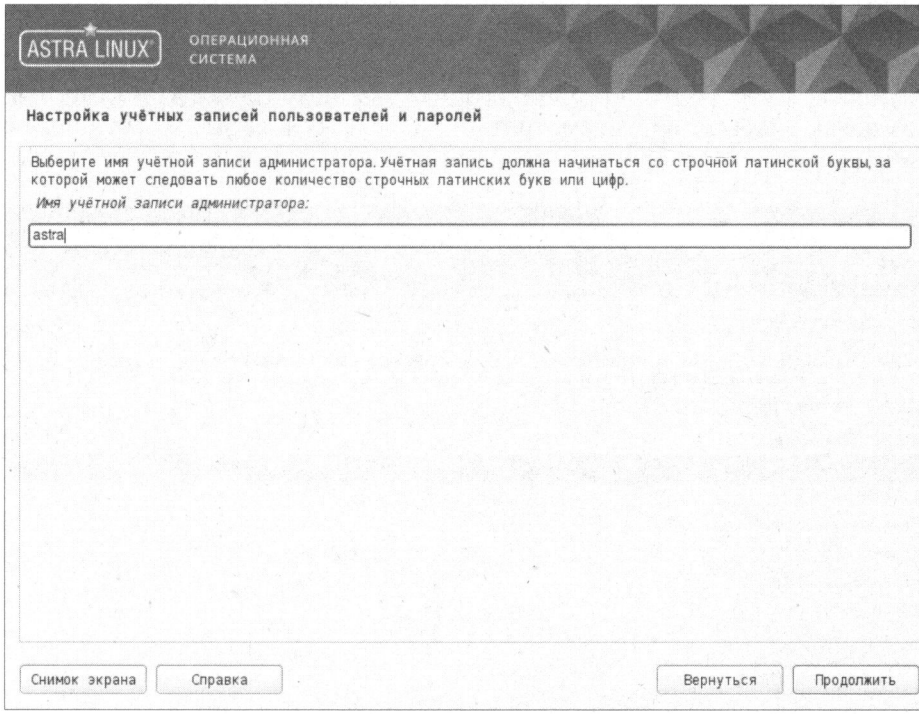

Рис. 2.15. Ввод имени пользователя

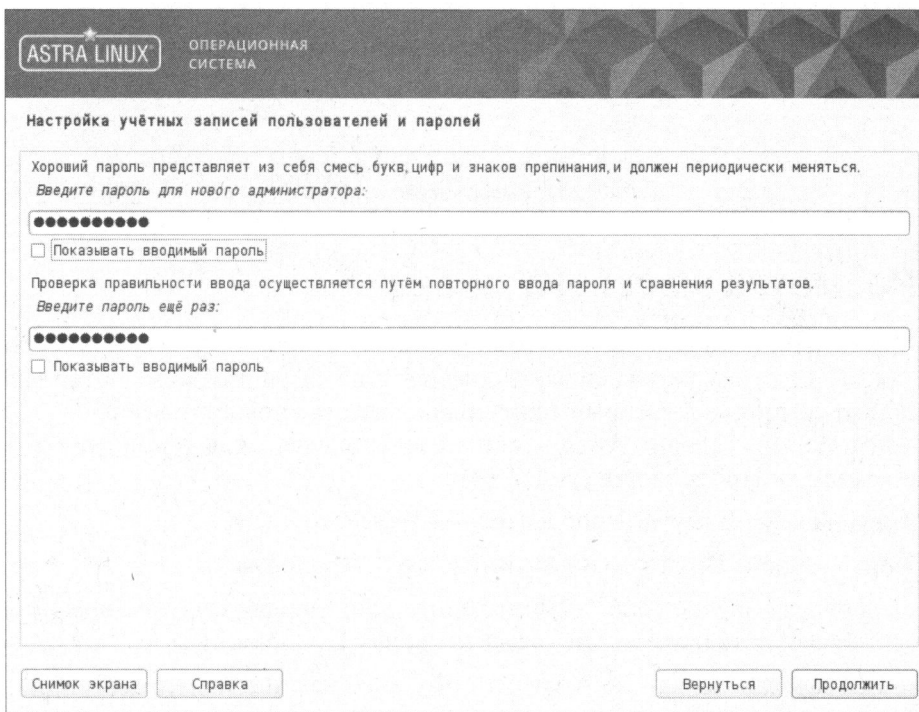

Рис. 2.16. Установка пароля

2.7. Установка часового пояса

На следующем этапе необходимо выбрать часовой пояс (рис. 2.17). Если нужного часового пояса в списке нет, посмотрите на подсказку в верхней части экрана настройки времени.

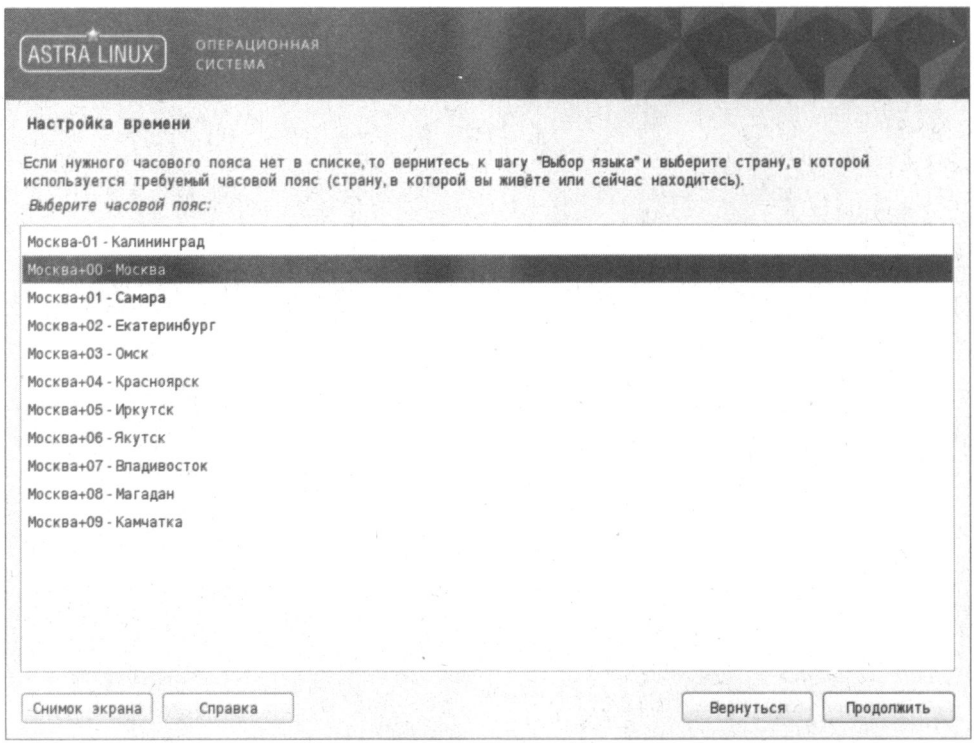

Рис. 2.17. Выбор часового пояса

2.8. Разметка жесткого диска

Самый ответственный шаг при установке любого дистрибутива — это разметка жесткого диска. Если вы устанавливаете дистрибутив на новый компьютер (или на виртуальный компьютер, который тоже можно назвать «новым»), все очень и очень просто — выберите вариант **Авто – использовать весь диск и настроить LVM** и нажмите кнопку **Продолжить** (рис. 2.18).

Рассмотрим на всякий случай процесс ручной разметки диска:

1. Выберите вариант **Вручную** и нажмите кнопку **Продолжить**.
2. На следующем экране (рис. 2.19) выберите ваш накопитель, таблицу разделов которого вы будете изменять, и нажмите кнопку **Продолжить.**
3. Если был выбран новый, не размеченный ранее накопитель, тогда инсталлятор предложит вам создать новую пустую таблицу разделов (рис. 2.20), — выберите **Да** и нажмите кнопку **Продолжить**.

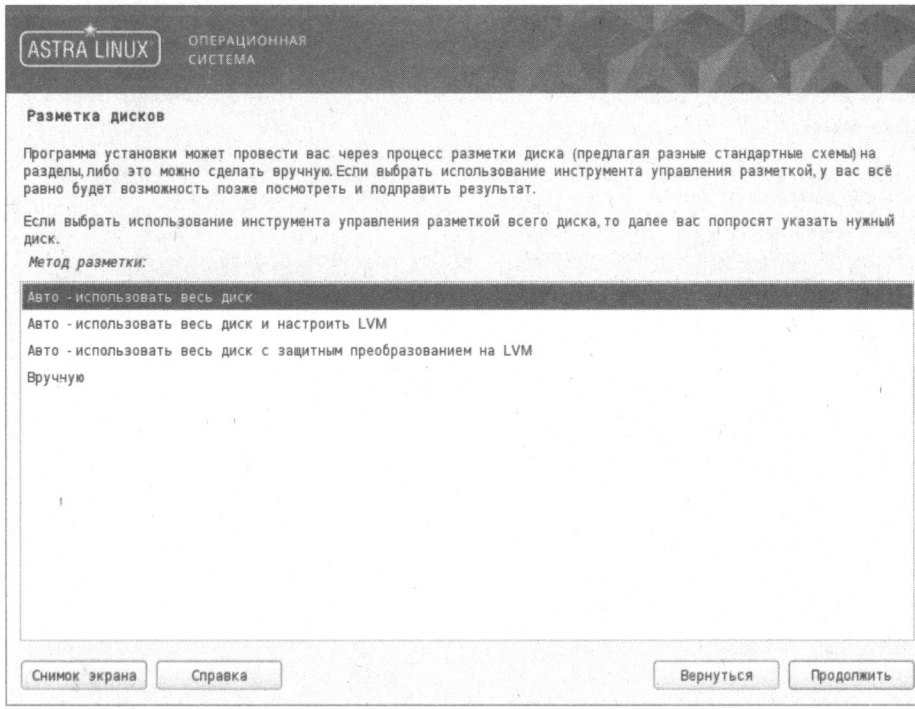

Рис. 2.18. Разметка жесткого диска: начальный экран

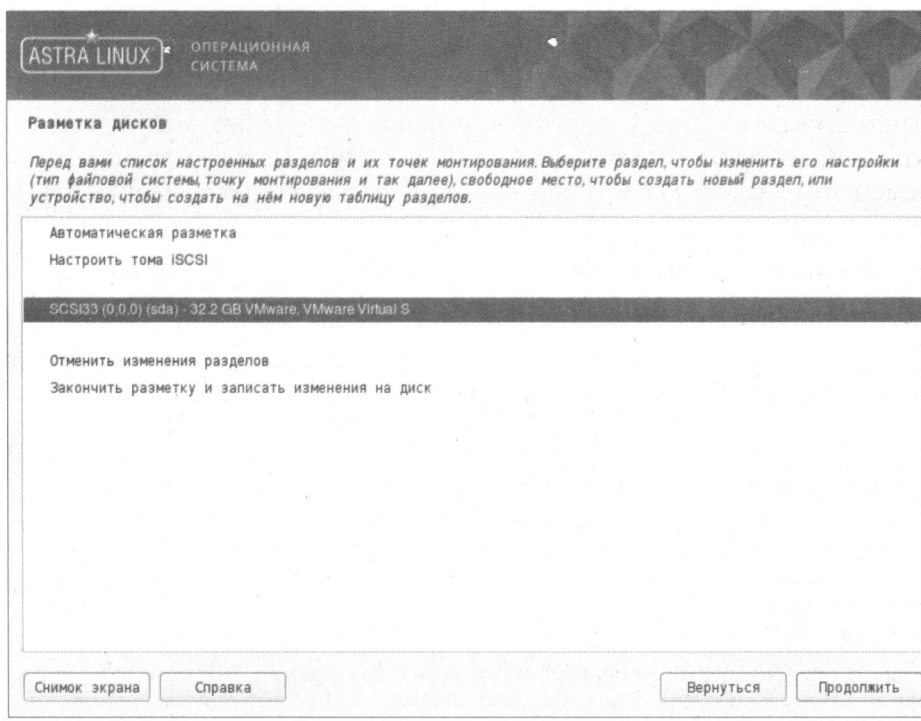

Рис. 2.19. Выберите накопитель

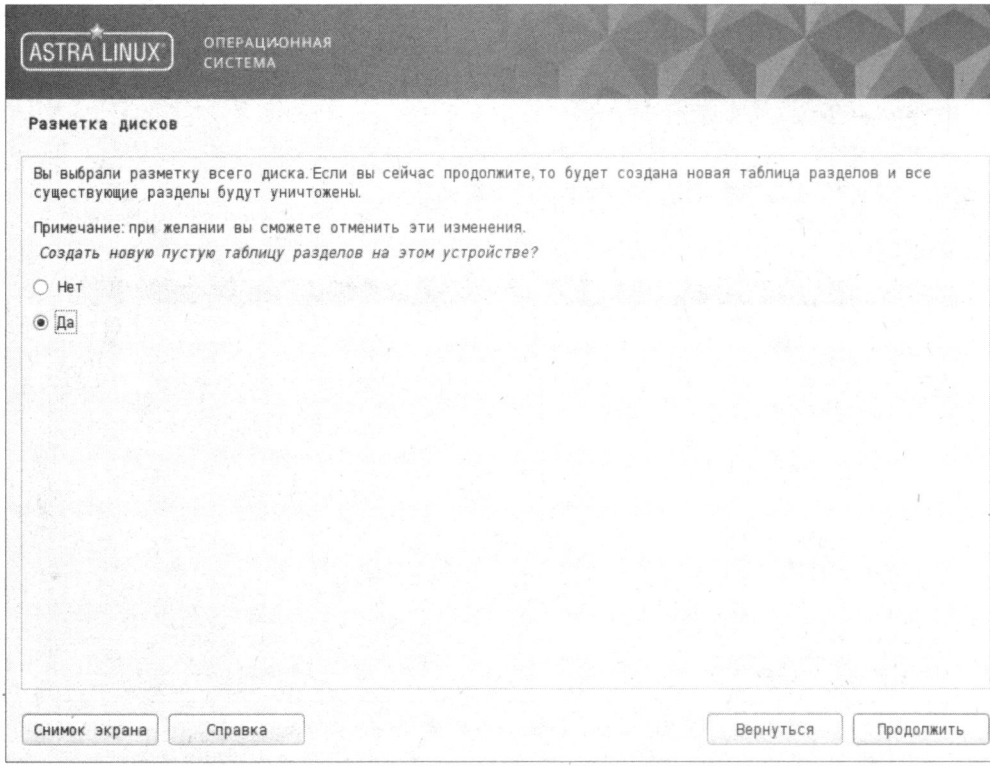

Рис. 2.20. Создание новой пустой таблицы разделов: выберите **Да**

Если же таблица разделов существует и не пуста, вы увидите список уже созданных разделов, где у вас будет возможность удалить ненужные разделы и создать на их месте разделы для Astra Linux. Далее на иллюстрациях мы предполагаем, что диск был новым (как при установке на виртуальную машину).

4. Выберите элемент списка, помеченный как **СВОБОДНОЕ МЕСТО** (рис. 2.21) и нажмите кнопку **Продолжить**.
5. Выберите **Создать новый раздел** (рис. 2.22) и нажмите кнопку **Продолжить**.
6. Установите размер нового раздела (рис. 2.23) и нажмите кнопку **Продолжить**. Учтите при этом, что вам понадобится место для раздела загрузчика и раздела подкачки.

Какими должны быть размеры разделов загрузчика и подкачки?

Размер раздела для загрузчика — 400 Мбайт, этого вполне достаточно. Впрочем, создавать его необязательно. При использовании MBR можно обойтись двумя разделами: один — для корневой файловой системы (/), второй — для раздела подкачки. Более того, даже раздел подкачки можно не создавать, но лучше создать, поскольку с разделом подкачки система работает быстрее, чем с файлом подкачки.

Размер раздела подкачки зависит от объема ОЗУ и задач, которые планируется выполнять на компьютере. Минимальный размер — 4 Гбайт при небольших объемах ОЗУ (2–8 Гбайт). При объеме ОЗУ 16 Гбайт и более (если имеется в виду домашнее/офисное применение дистрибутива) раздел подкачки можно вовсе не создавать

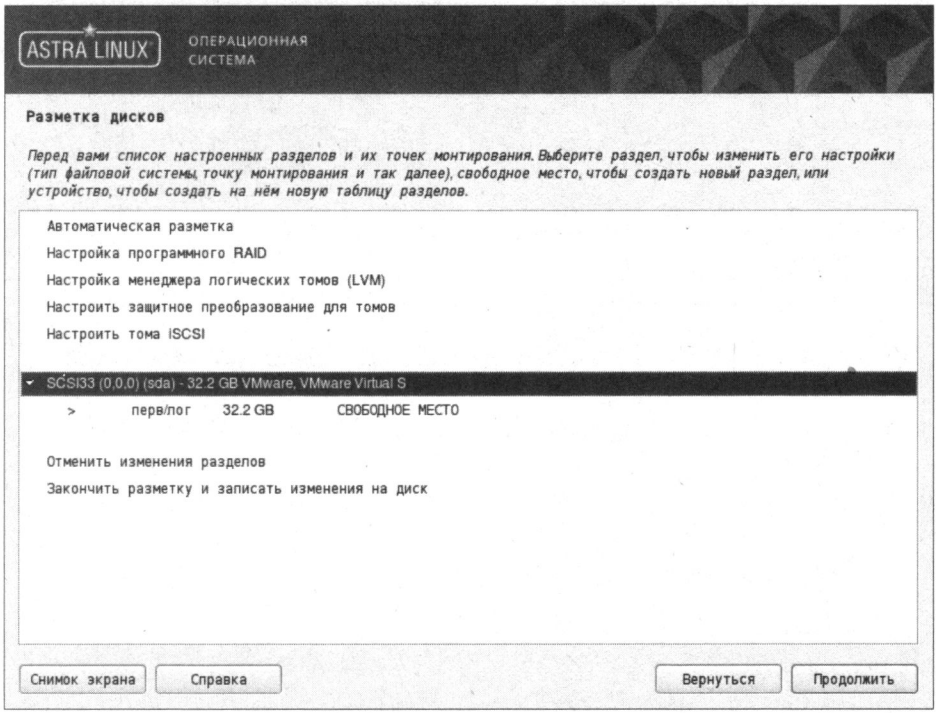

Рис. 2.21. Список разделов накопителя пуст

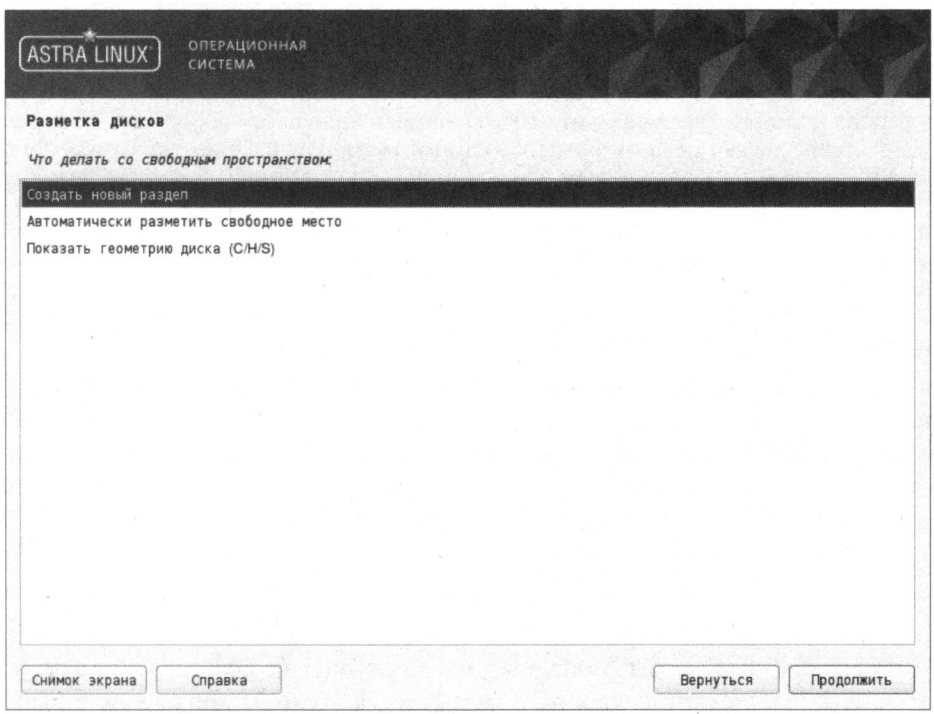

Рис. 2.22. Действия над свободным пространством

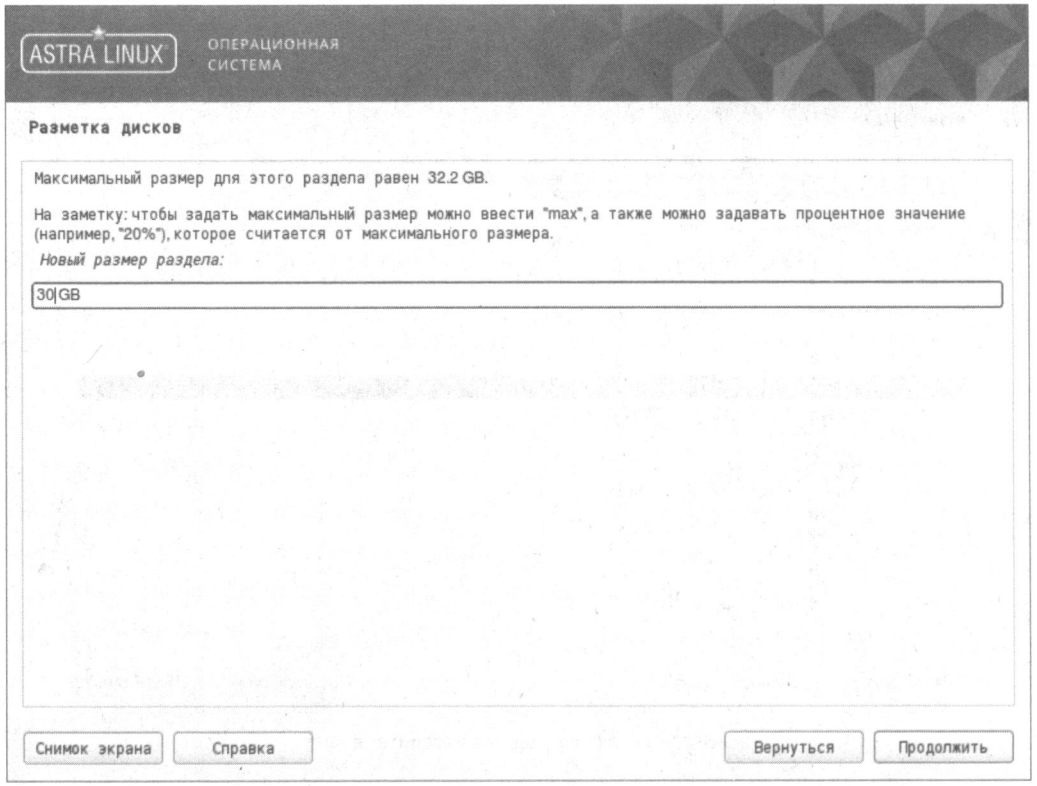

Рис. 2.23. Размер нового раздела

(при желании всегда можно будет создать *файл* подкачки). Максимальный размер раздела подкачки также зависит от выполняемых задач. При небольших объемах ОЗУ (2–4 Гбайт) можно создать раздел подкачки размером 8 Гбайт максимум (на таком слабом компьютере вряд ли вам понадобится раздел подкачки большего размера).

7. На вопрос, каким должен быть раздел: **Первичный** или **Логический**, выберите **Первичный** (рис. 2.24).
8. Поместите новый раздел в **Начало** свободного пространства (рис. 2.25) и нажмите кнопку **Продолжить**.
9. На сводной странице с информацией о новом разделе (рис. 2.26) убедитесь, что точка монтирования равна /, выполните двойной щелчок на элементе **Метка 'загрузочный'** — чтобы пометить раздел как загрузочный, затем выберите **Настройка раздела закончена** и нажмите кнопку **Продолжить**. Если вы заметили, что что-то сделали не так, выберите **Удалить раздел**, нажмите кнопку **Продолжить** и снова выполните пункты 4–8 для повторного создания нового раздела.
10. Повторите действия по пунктам 4–8 для создания еще одного раздела — раздела подкачки. Когда вы попадете на страницу со сводной информацией (см. рис. 2.26), щелкните двойным шелчком на **Использовать как** и выберите **раздел подкачки** (рис. 2.27).

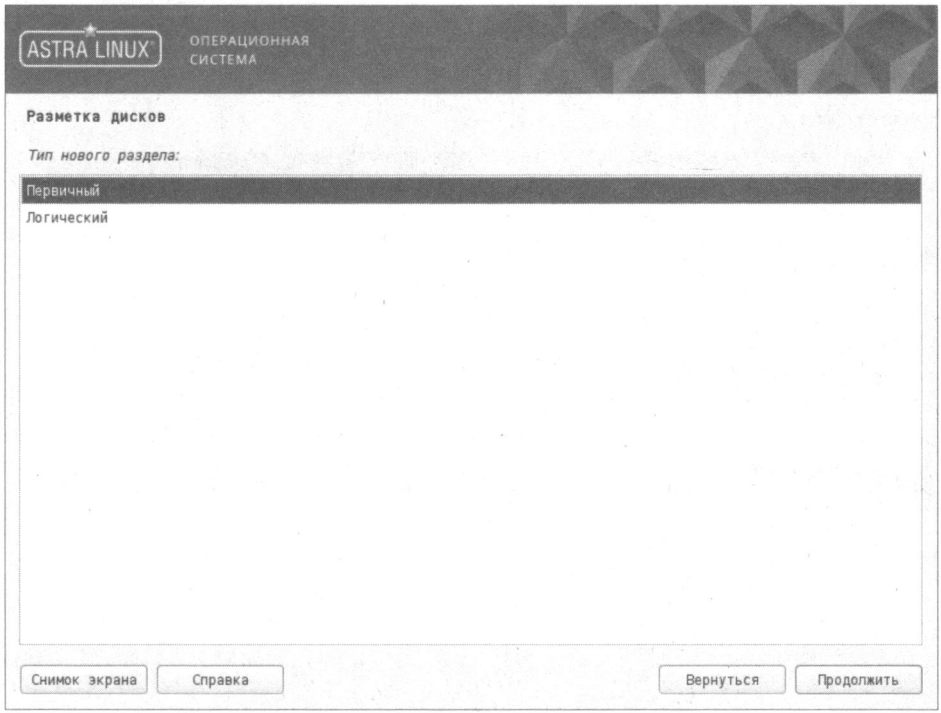

Рис. 2.24. Тип раздела: первичный

Рис. 2.25. Куда поместить раздел?

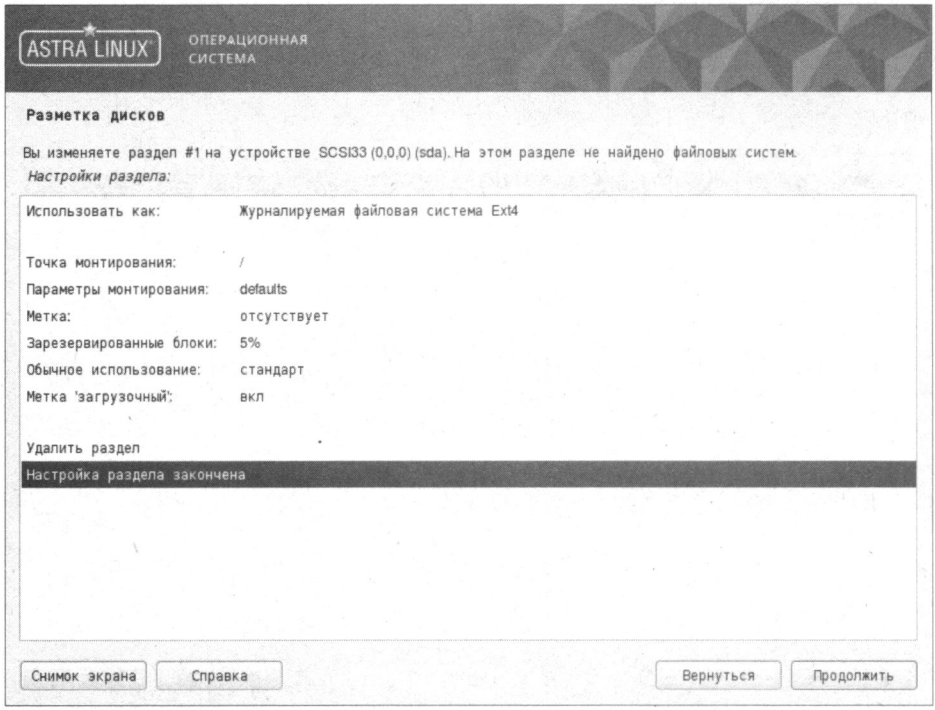

Рис. 2.26. Сводная страница с информацией о созданном разделе

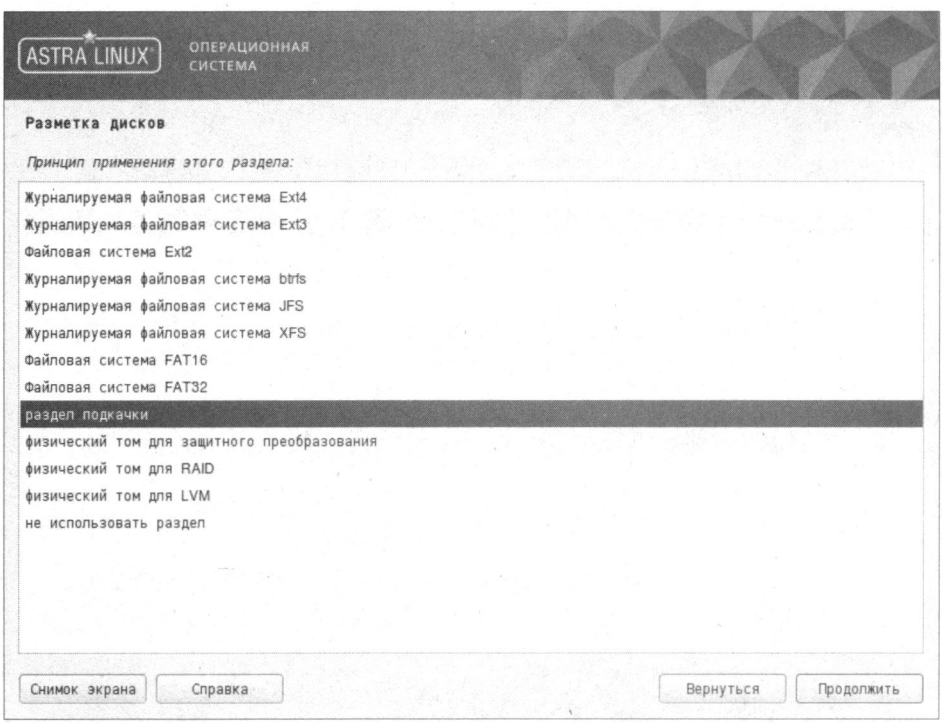

Рис. 2.27. Раздел подкачки

Раздел загрузчика

На домашнем компьютере, как правило, нет смысла создавать отдельные разделы для каталогов /home, /var и пр. Для корневой файловой системы вам понадобится только один раздел (/) и раздел подкачки.

11. В результате у вас должна получиться таблица разделов, показаная на рис. 2.28. Выберите **Закончить разметку и записать изменения на диск** и нажмите кнопку **Продолжить**. На следующем экране (рис. 2.29) выберите **Да** и нажмите кнопку **Продолжить**.

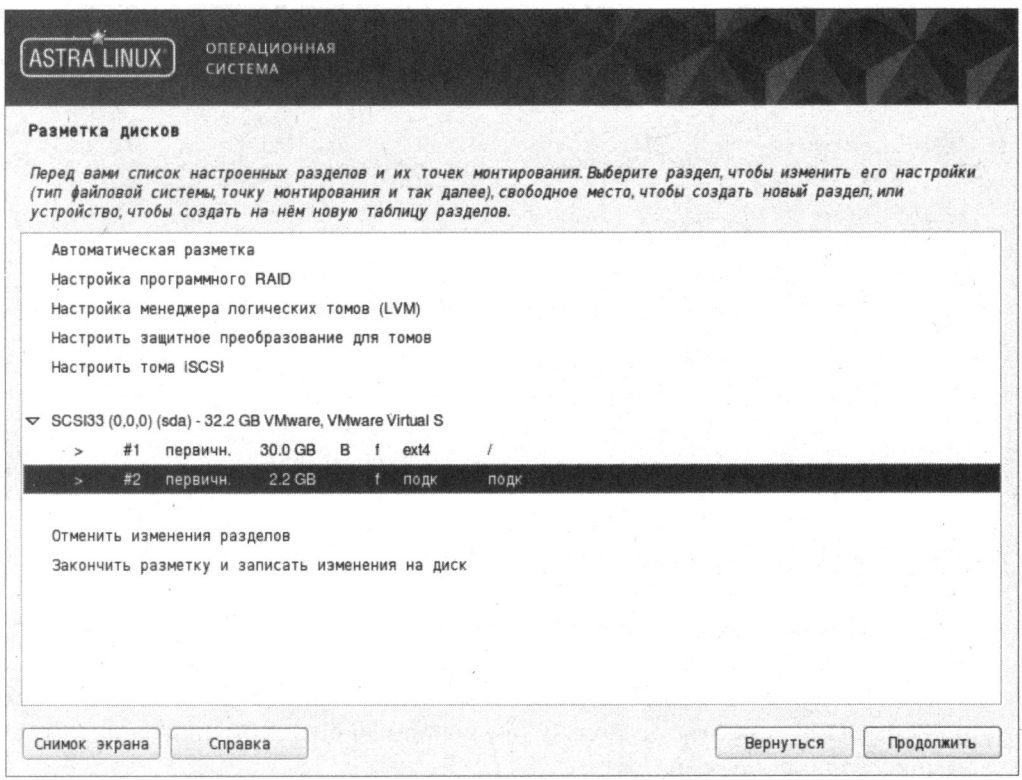

Рис. 2.28. Окончательная таблица разделов

Первичный или логический?

Таблица разделов главной загрузочной записи (Master Boot Record, MBR) подразумевает, что *первичных* разделов может быть максимум четыре. В нашем случае, когда мы создаем всего три раздела, для упрощения структуры диска можно их все сделать первичными. Если же количество создаваемых нами разделов будет больше четырех, без использования *логических* разделов нам не обойтись. В этом случае загрузочные разделы операционных систем создаются как первичные, а все остальные — как логические. В Linux первичным разделам первого диска (a) будут присвоены имена /dev/sda1, ..., /dev/sda4. Имя /dev/sda5 присваивается уже первому логическому разделу. В этой главе мы все разделы создали как первичные, чтобы имена устройств получились /dev/sda1, /dev/sda2 и /dev/sda3 — по эстетическим соображениям.

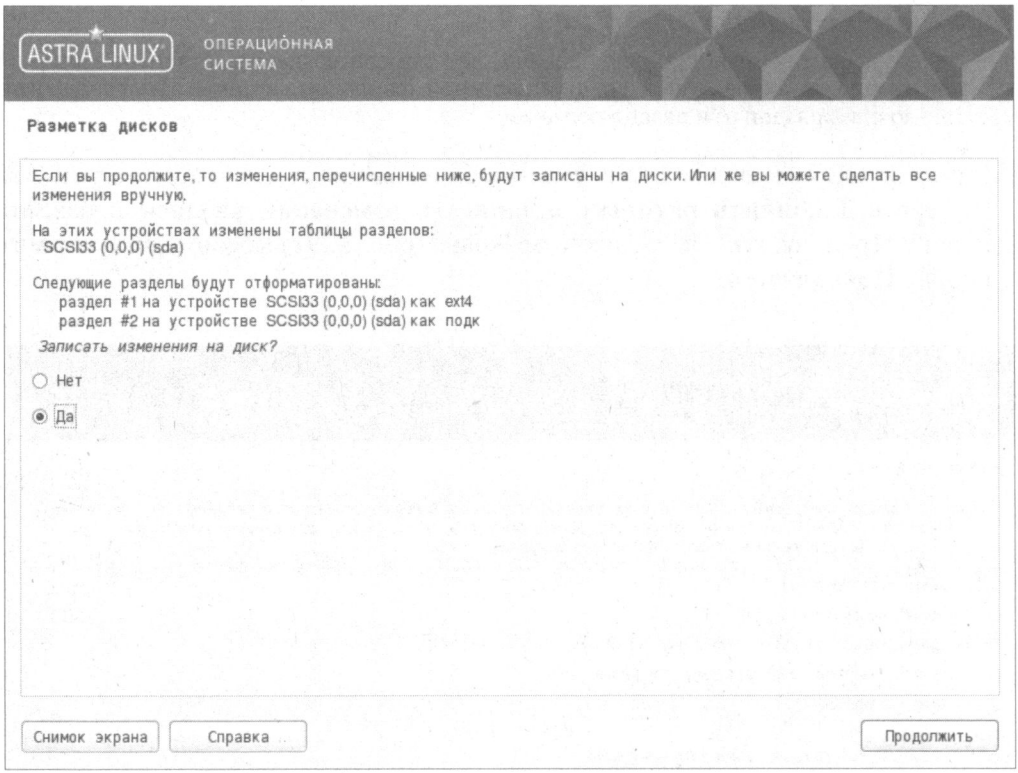

Рис. 2.29. Выберите **Да** и нажмите кнопку **Продолжить**

2.9. Выбор компонентов системы

На следующем этапе нам предстоит выбрать версию ядра и устанавливаемое программное обеспечение. Здесь все достаточно просто — выберите самую новую версию ядра (рис. 2.30), а на экране со списком компонентов (рис. 2.31) можно просто нажать кнопку **Продолжить**, поскольку предлагаемый системой список оптимален для домашнего использования.

Выбор версии ядра заслуживает отдельного разговора. Инсталлятор предлагает несколько версий ядра: 4.15, 5.4, 5.10, 5.15 и 6.1 (для версии 1.7.5 дистрибутива Special Edition), а также два варианта каждого ядра (кроме 6.1): generic и hardened. Что выбрать? Если вам не с руки вникать в подробности, то выбирайте версию 6.1 — самую новую версию ядра. А теперь немного подробностей.

Версия 4.15 очень древняя (уже есть версия 6.7), не является долгосрочной и ставить ее смысла нет. Долгосрочными признаны версии 5.4, 5.10, 5.15 и 6.1:

- 5.4 — поддержка до декабря 2025 г.
- 5.10 — поддержка до декабря 2026 г.
- 5.15 — поддержка до декабря 2026 г.

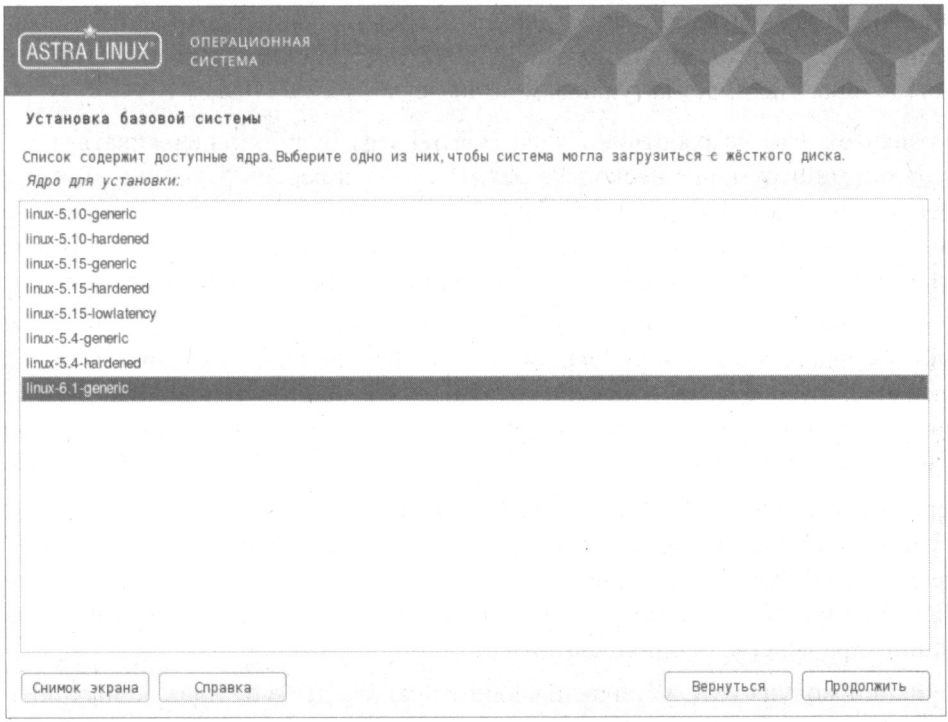

Рис. 2.30. Выбор версии ядра

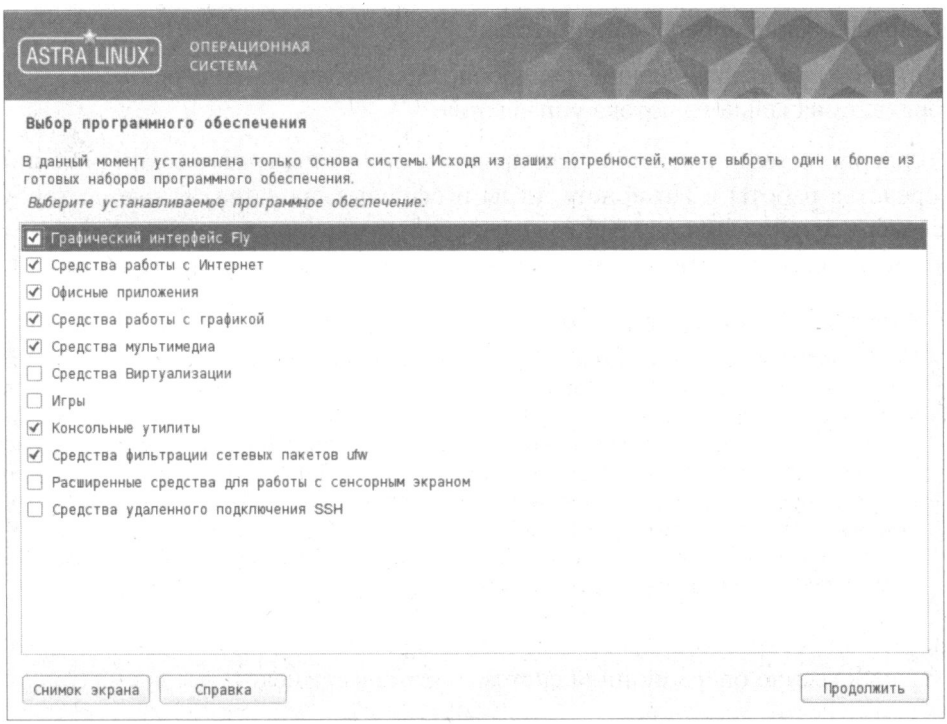

Рис. 2.31. Выбор компонентов системы (выбор по умолчанию)

- 6.1 — стабильное ядро с супердлинным сроком поддержки (SLTS, Super-long-term Stable). Ядро вышло 11 декабря 2022 года, срок поддержки — 10 лет, следовательно, оно будет актуальным до 11 декабря 2032 года.

Это означает, что разработчики ядра (**kernel.org**) будут поддерживать ядра этих версий как минимум еще несколько лет (год — для версии 5.4), и вы можете быть уверены, что всевозможные патчи безопасности для них в течение указанного времени будут доступны. Пока же все эти версии являются активно поддерживаемыми, и в их репозиториях имеются изменения по состоянию на январь 2024 г.

Учитывая, что мы устанавливаем систему не на месяц и не на два, лучше выбрать самую последнюю доступную версию — 6.1. Другие версии можно выбрать, если вы уверены, что поступаете правильно. Например, знаете, что какой-то нужный вам драйвер или приложение плохо работают с версией 6.1, — тогда можно попробовать выбрать версию 5.15.

Теперь о вариантах generic и hardened. Универсальное ядро generic подойдет как для рабочей станции, так и для сервера — это ядро общего назначения. А вот hardened подойдет больше для сервера, поскольку некоторые пользовательские приложения на этом ядре могут работать некорректно. В частности, кроме базовых функций ядра generic, оно поддерживает:

- функции модуля PaX, устанавливающие правила доступа прикладных программ к адресному пространству памяти;
- функции PIE (Position Independent Executables) и SSP (Stack Smashing Protector), предотвращающие переполнение стека;
- функции, предотвращающие внедрение вредоносного кода в процессы путем перехвата начального потока управления.

Для серверного же применения, как правило, не нужны графический рабочий стол Fly, средства работы в Интернете, игры и офисные средства. Также не требуются для сервера приложения для работы с сенсорным экраном и средства мультимедиа. Зато понадобятся средства удаленного доступа SSH и средства виртуализации.

> ***Сколько занимает система?***
>
> Сразу после установки системы в конфигурации, показанной на рис. 2.31, она заняла 7,3 Гбайт. Если вы выберете еще какие-либо компоненты, для них потребуется дополнительное место. Наоборот, если вы снимете отметку с каких-то компонентов, система займет места меньше. Также нужно понимать, что для полноценной работы системы, кроме места для компонентов, устанавливаемых на старте, понадобится дополнительное пространство, — наверняка, вам захочется установить еще какие-нибудь нужные вам программы, кроме того, в процессе работы система создает временные файлы, ну и не забывайте о своих собственных файлах и файле подкачки. Поэтому практический минимум — это 20 Гбайт.

После выбора стартового программного обеспечения начнется установка на компьютер собственно операционной системы, что займет некоторое время (рис. 2.32).

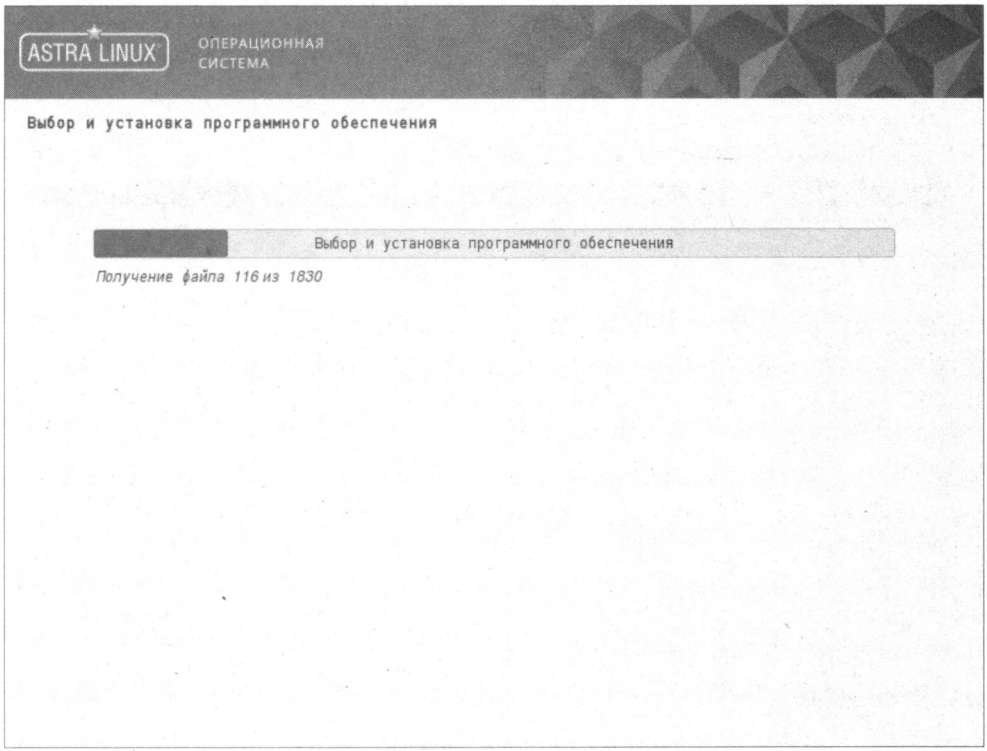

Рис. 2.32. Установка системы

2.10. Выбор уровня защищенности

При установке Astra Linux Special Edition необходимо выбрать уровень защищенности (рис. 2.33):

- базовый уровень защищенности «Орел» — здесь настройки будут такие же, как и в Common Edition;
- усиленный уровень защищенности «Воронеж» — более защищенный уровень, где по умолчанию включен мандатный контроль целостности (МКЦ), запрет установки бита выполнения, блокировка интепретаторов и т. д.;
- максимальный уровень защищенности «Смоленск» — вдобавок к предыдущему уровню поддерживается мандатное управление доступом (МРД).

Выбор уровня защищенности — это не только выбор другого набора дополнительных настроек ОС (рис. 2.34). Да, на более защищенных уровнях (по сравнению с базовым) задаются другие настройки — например, запрещен ввод `sudo` без пароля, а также устанавливается пароль на изменение конфигурации загрузчика GRUB2 (об этом позже).

Кроме всего этого, уровень «Воронеж» подразумевает установку СЗИ собственной разработки ГК «Астра», входящего в состав подсистемы безопасности PARSEC.

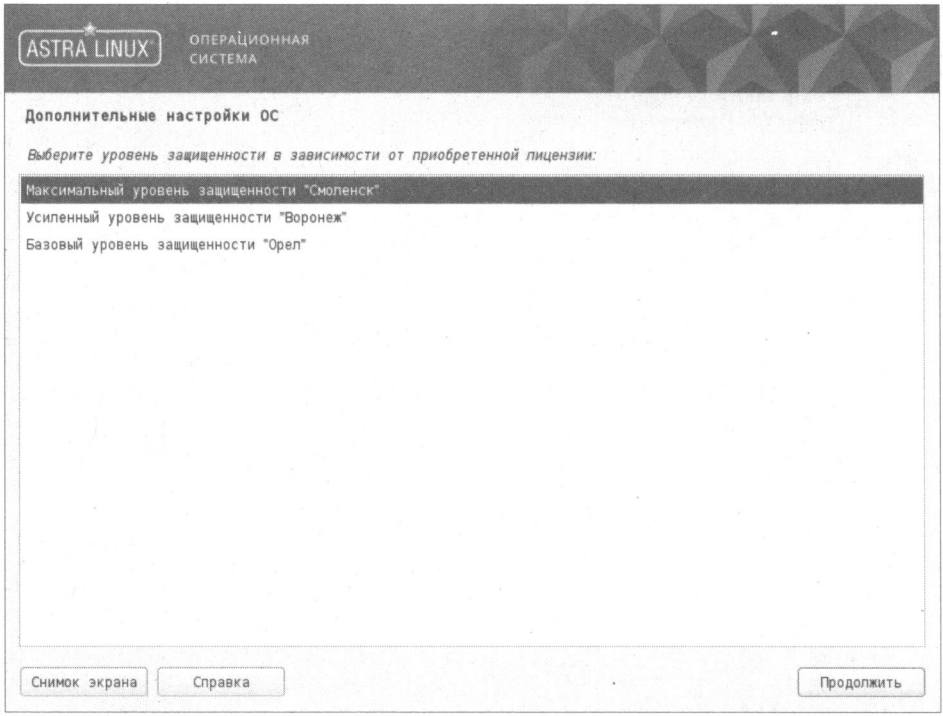

Рис. 2.33. Выбор уровня защищенности

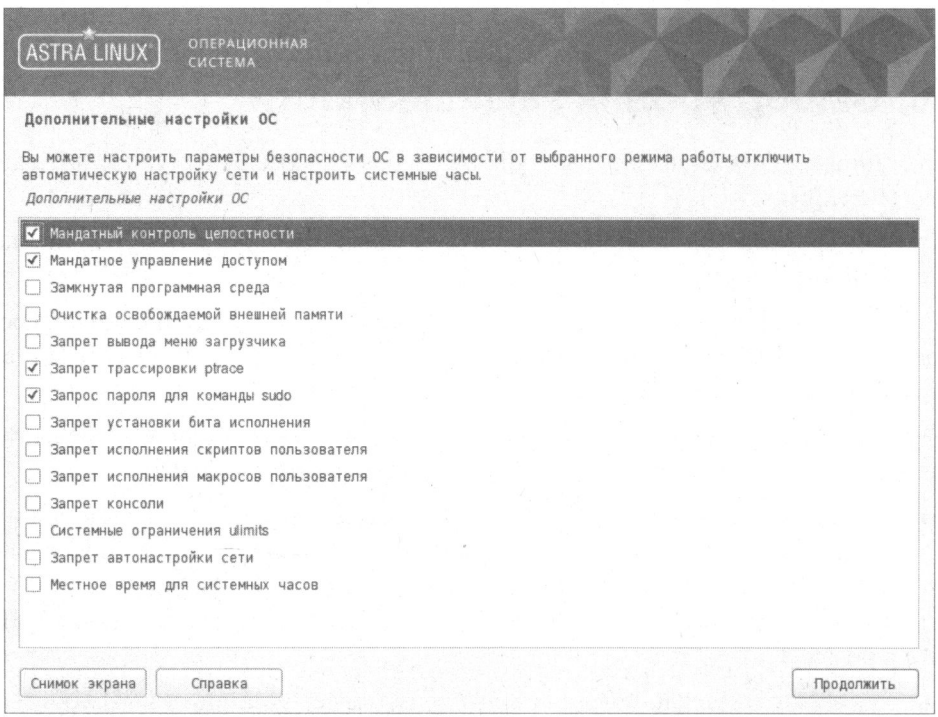

Рис. 2.34. Дополнительные настройки ОС

В первую очередь — это мандатный контроль целостности (МКЦ). МКЦ существенно повышает защищенность Astra Linux от взлома, заражения вредоносным ПО, внедрения программных закладок, получения несанкционированных прав и т. д.

Уровень «Смоленск» дополняет возможности режима «Воронеж» функциями мандатного управления доступом (МРД) для защиты от угрозы конфиденциальности информации. Смысл МРД в следующем:

- читать данные из файла/каталога могут только те процессы и пользователи, которые обладают не меньшим уровнем конфиденциальности (не целостности, а именно конфиденциальности), чем у запрашиваемых данных;
- записывать данные в файл или каталог могут только процессы с уровнем конфиденциальности, равным или меньшим, чем у этого файла/каталога.

Подробно МКЦ и МРД мы рассмотрим в *главах 26* и *27* соответственно.

2.11. Установка загрузчика GRUB и завершение установки

На этом этапе установщик спросит вас, нужно ли установить загрузчик GRUB в основную (главную) загрузочную запись (рис. 2.35). Как правило, следует вы-

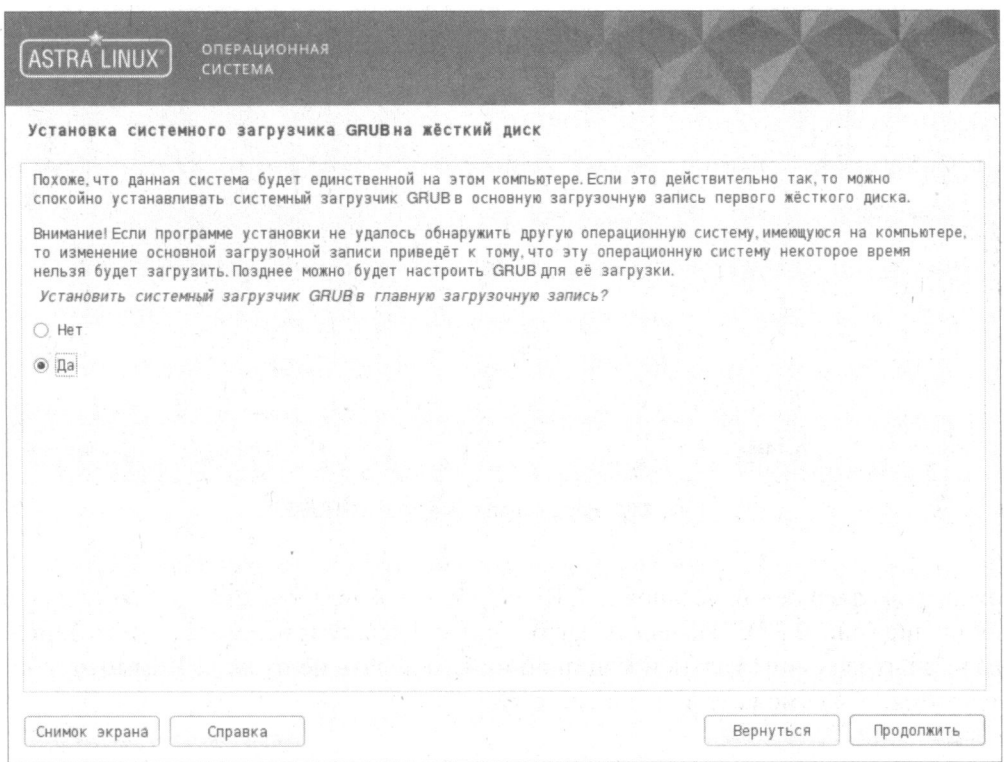

Рис. 2.35. Установка загрузчика

брать **Да**. Опция **Нет** допустима, только если в главной загрузочной записи уже есть загрузчик и вы планируете его настроить еще и для загрузки Astra (это тот случай, когда у вас на компьютере несколько операционных систем).

> ***Внимание!***
> Не перезагружайте компьютер до установки загрузчика, иначе вы не сможете потом загрузиться!

При установке Special Edition вам нужно будет ввести пароль для загрузчика GRUB2 (рис. 2.36) — это обязательно при установке уровня защищенности выше, чем базовый.

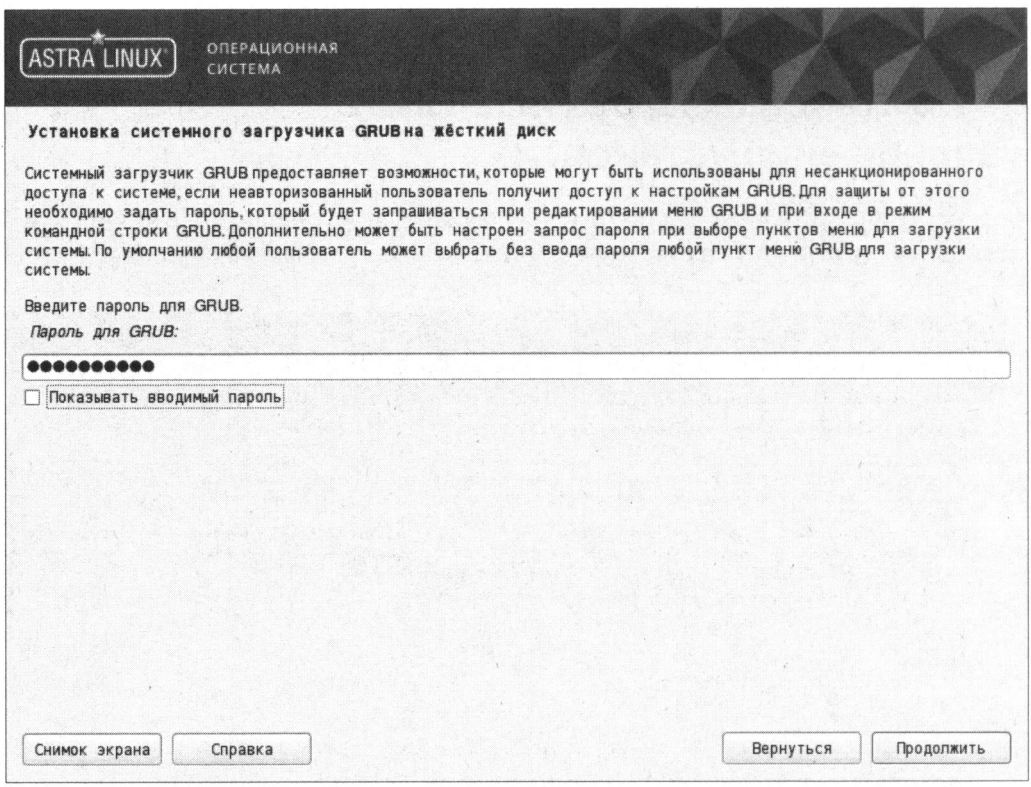

Рис. 2.36. Ввод пароля загрузчика GRUB2

Дождитесь завершения установки GRUB — об этом вас известит соответствующее сообщение (рис. 2.37). Извлеките загрузочную флешку и нажмите кнопку **Продолжить** (в случае с виртуальной машиной ничего делать не нужно). Компьютер перезагрузится, и вы сможете войти в систему.

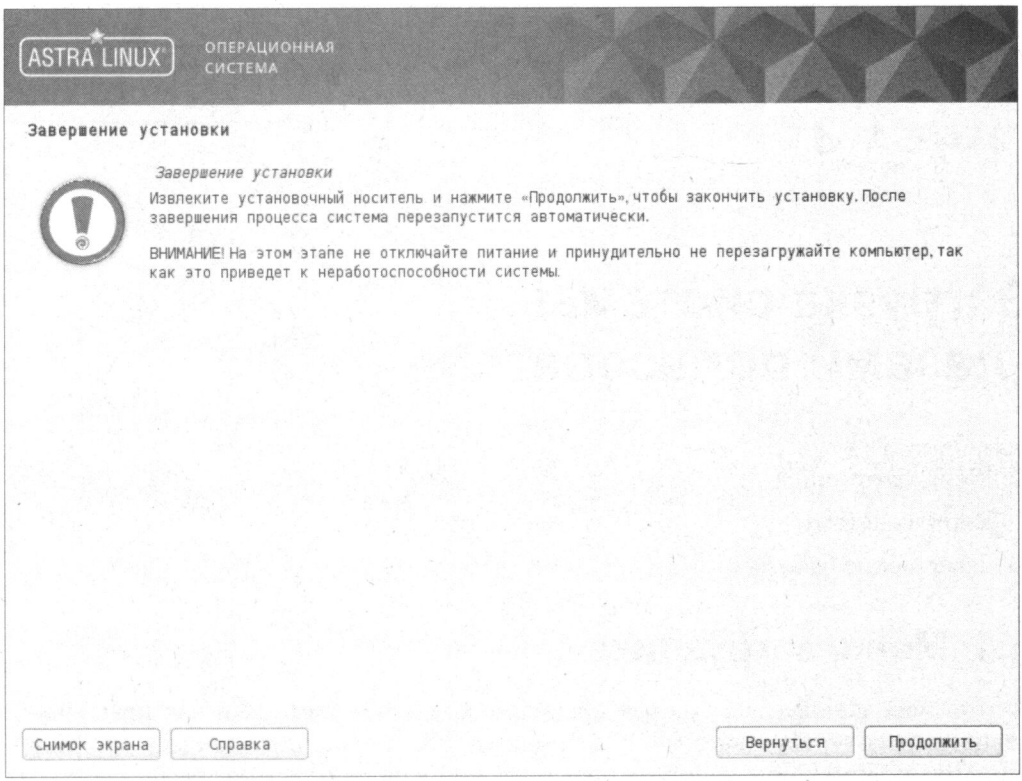

Рис. 2.37. Установка системы завершена

ASTRA LINUX®

ГЛАВА 3

Загрузка системы глазами пользователя

- Меню загрузчика
- Вход в систему
- Завершение работы

3.1. Меню загрузчика

В этой небольшой главе мы разберемся, как войти в систему и как правильно завершить работу. Начнем с меню загрузчика. Как только «прошивка» материнской платы выполнит процедуру тестирования, она начнет поиск загрузчика в соответствии с порядком загрузки, указанным в ее настройках. Когда загрузчик будет найден, «прошивка» передает ему управление.

На рис. 3.1 показано меню загрузчика GRUB2. Этот загрузчик используется во всех современных дистрибутивах Linux и альтернативы ему попросту нет. Однако внешний вид загрузчика зависит от его настроек. Разработчики Astra Linux заморочились с красивым фоном, и теперь загрузчик этого дистрибутива работает в графическом режиме — попытка побаловать пользователя, что ли? Ведь в той же Ubuntu меню загрузчика исходно не отображается вовсе, а просто загружается метка по умолчанию. Если же пользователь Ubuntu хочет увидеть это меню (которое, кстати, будет просто текстовым), ему нужно удерживать при загрузке клавишу <Shift>.

Содержимое меню опять-таки зависит от настроек загрузчика и установленных образов ядра. При установке системы вы могли выбрать другое ядро, поэтому меню по умолчанию у вас может отличаться от приведенного на рис. 3.1. Но это не столь важно. Меню позволяет:

- выбрать загрузочную метку.

  Для этого используйте клавиши со стрелками вверх/вниз и подтверждайте выбор нажатием клавиши <Enter>;

- отредактировать параметры загрузочной метки.

  Для этого нужно выбрать метку и нажать клвишу <e>. Далее загрузчик либо запросит имя пользователя/пароль для редактирования загрузочной метки, либо

сразу отобразит редактор. Обычно редактор используется для правки параметров ядра Linux — нужная строка начинается как раз со слова «linux». Для продолжения загрузки с вашими параметрами нажмите клавишу <F10> или комбинацию клавиш <Ctrl>+<X>.

Рис. 3.1. Astra Linux: меню загрузчика GRUB2

Впрочем, вы можете вообще ничего не делать, а просто подождать 5 секунд — будет загружена метка по умолчанию (обычно это Linux, если в настройках загрузчика не задано другое). Подробно о загрузке мы поговорим в следующей главе.

3.2. Вход в систему

По завершении загрузки вы увидите экран входа в систему, отображаемый менеджером рабочего стола Fly (рис. 3.2). Здесь можно сразу ввести имя пользователя, пароль и нажать кнопку **Войти**.

Однако этот экран предоставляет гораздо больше возможностей, чем просто вход в систему:

- в его нижнем левом углу есть кнопка **Клавиатура**, вызывающая экранную клавиатуру, которая может пригодиться, если обычная недоступна (например, на планшете);

Рис. 3.2. Astra Linux: экран входа в систему

- кнопки **Перезагрузка**, **Сон** и **Выключение** позволяют соответственно перезагрузить, перевести в режим сна и выключить компьютер;
- индикатор активной раскладки EN по нажатию на него позволяет переключать раскладку. Возле индикатора выводятся текущие дата/время;
- расположенное в верхнем левом углу экрана меню **Действия** содержит команды:
  - **Смена сессии** — если установлено несколько графических интерфейсов, здесь можно выбрать нужный;
  - **Перезапуск граф. окружения** — позволяет перезапустить графическую подсистему, если что-то работает не так, как следует;
  - **Консольный вход** — переключает вас на консоль, где вы сможете войти в систему и продолжить работу в текстовом режиме. Вы также можете переключиться на консоль, нажав комбинацию клавиш <Ctrl>+<Alt>+<F1. Вернуться обратно можно с помощью комбинации клавиш <Alt>+<F7>;
  - **Перезагрузка**, **Выключение**, **Сон**, **Гибернация** — команды завершения работы.

- Меню **Тип сессии** (рис. 3.3) позволяет выбрать тип сессии:
  - **Режим восстановления** — запускает режим восстановления;
  - **Десктоп** — обычный режим, запускаемый по умолчанию;
  - **Планшетный** — планшетный режим, подходящий для использования на планшете.

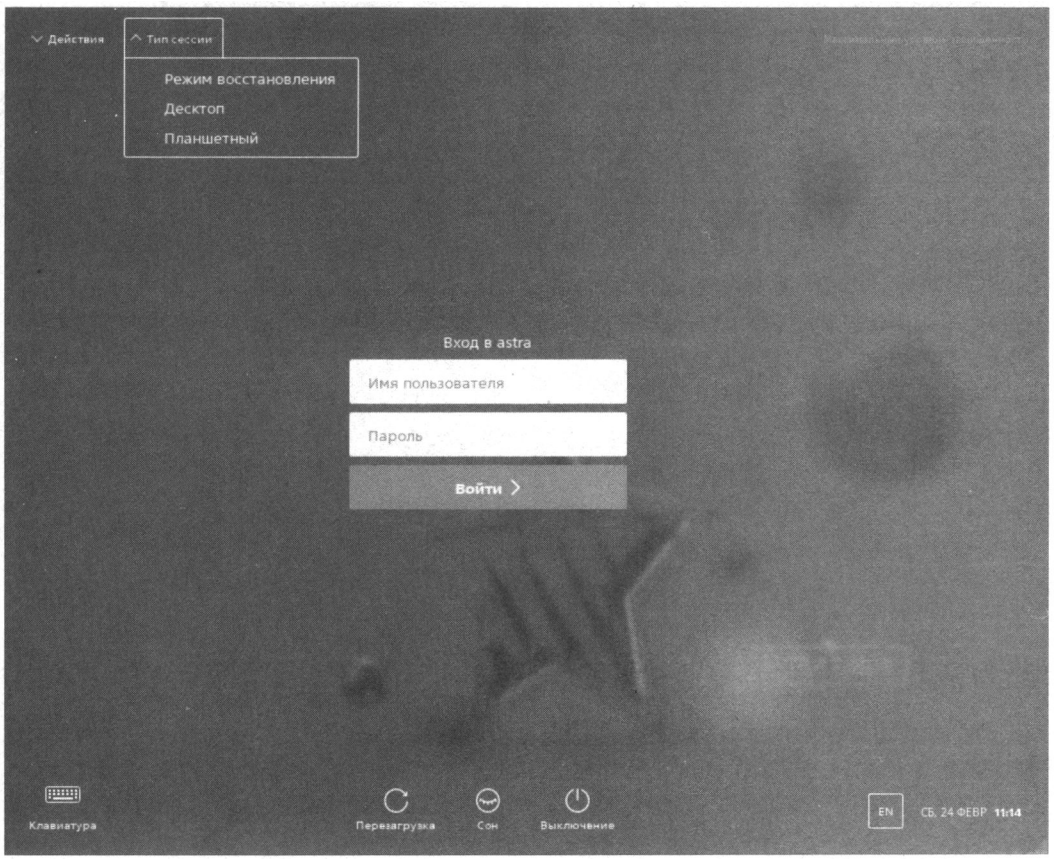

Рис. 3.3. Astra Linux: меню **Тип сессии**

При работе в Astra Linux Special Edition после ввода имени пользователя и пароля вы увидите меню выбора уровней конфиденциальности и целостности (рис. 3.4). Подробно об этих режимах мы поговорим в *главах 26* и *27*, а пока просто нажмите кнопку **Войти**.

3.3. Завершение работы

Для завершения работы следует выбрать в основном меню системы команду **Завершение работы** — откроется диалоговое окно **Выход или выключение** (рис. 3.5), где вы можете нажать одну из кнопок:

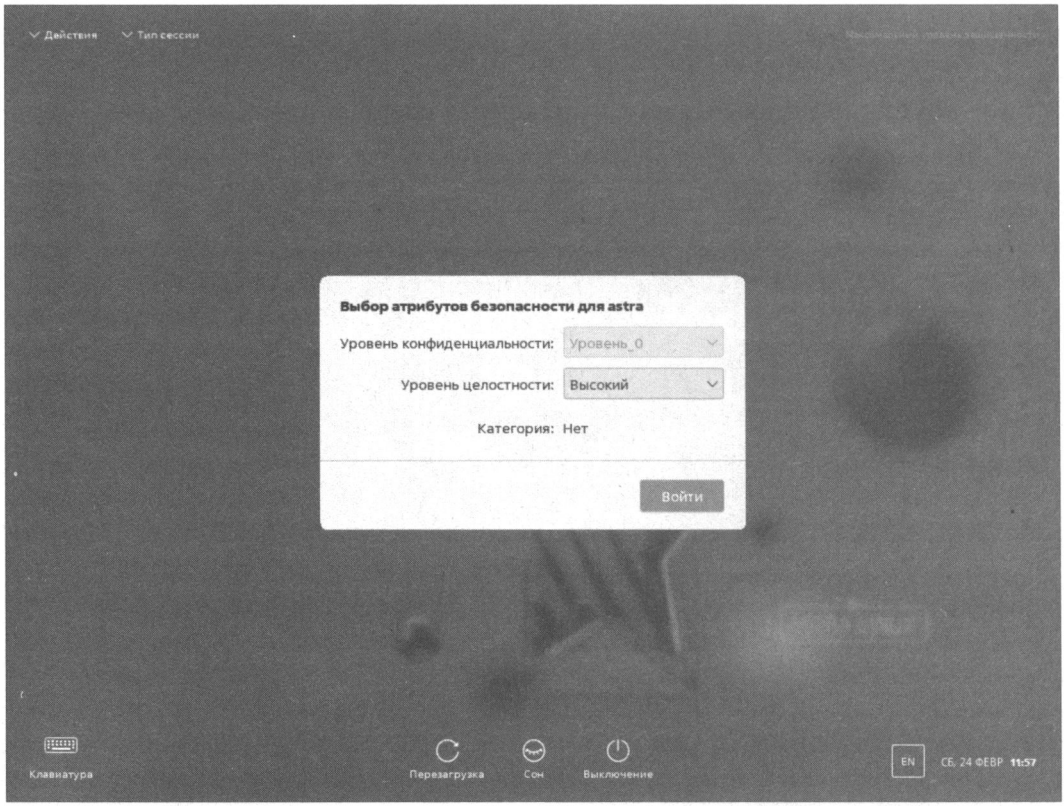

Рис. 3.4. Astra Linux Special Edition: выбор уровня целостности

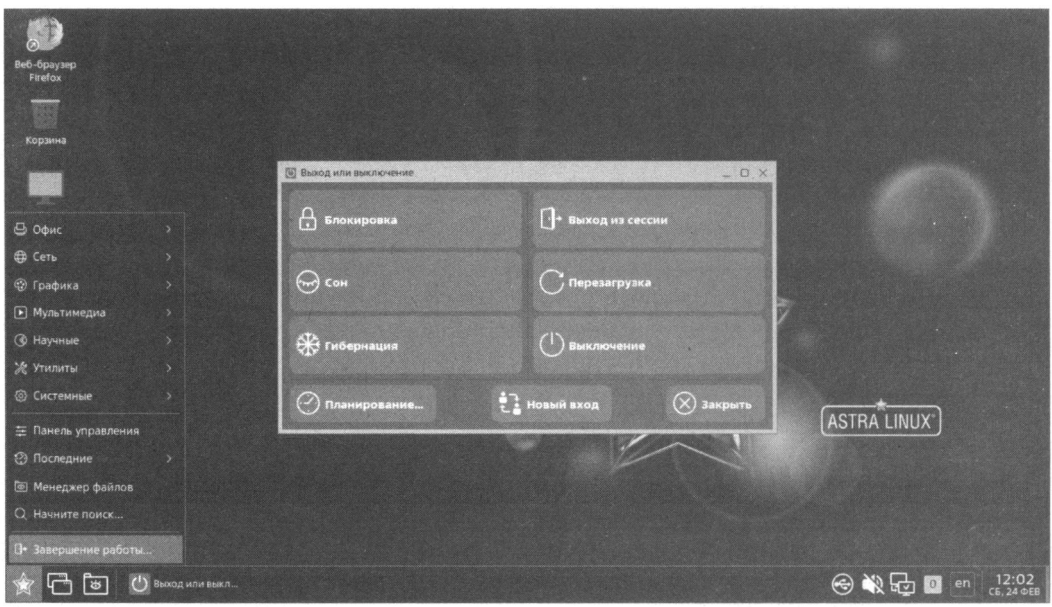

Рис. 3.5. Диалоговое окно завершения работы

- **Блокировка** — подойдет, если вы не хотите завершать работу компьютера, но вам ненадолго нужно от него отлучиться. Чтобы никто не смог за время вашего отсутствия воспользоваться вашим сеансом, нажмите эту кнопку;
- **Выход из сессии** — завершает текущий сеанс без выключения компьютера;
- **Сон** — переводит компьютер в режим сна. В этом режиме компьютер потребляет минимум энергии, но электропитание ему все еще необходимо. Выход из режима сна, как правило, моментальный, и можно быстро вернуться к работе. Все открытые на момент перевода компьютера в режим сна программы будут доступны;
- **Гибернация** — сохраняет дамп оперативной памяти на диск (соответственно на диске должно иметься свободное пространство, равное объему ОЗУ) и переводит компьютер в режим гибернации. Электропитание компьютера при этом отключается. Выход из режима гибернации занимает больше времени, чем из режима сна, но все равно происходит быстрее, чем обычная загрузка. Все программы, запущенные на момент перевода компьютера в режим гибернации, будут доступны (ведь у нас есть дамп памяти). Впрочем, не все компьютеры поддерживают этот режим;
- **Выключение** — обычное завершение работы с выключением питания. Как правило, тот вариант, который вы будете выбирать в большинстве случаев;
- **Перезагрузка** — перезагружает компьютер;
- **Планирование** — открывает диалоговое окно (рис. 3.6), в котором можно установить время, через которое будет выполнено действие завершения работы.

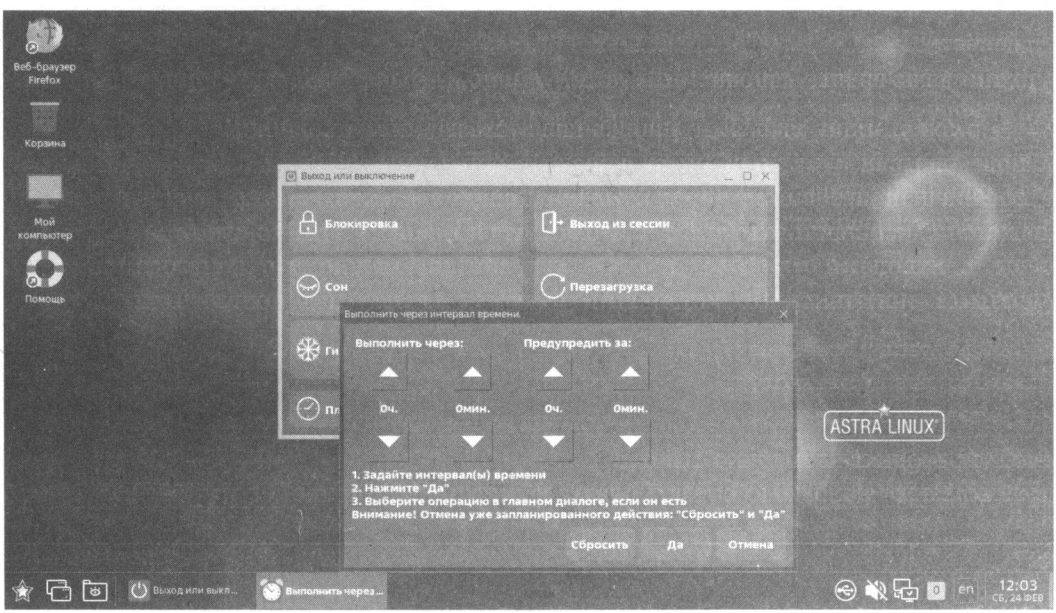

Рис. 3.6. Диалоговое окно планирования завершения работы

Алгоритм такой: сначала вы устанавливаете здесь режим выполнения, а потом нажимаете одну из кнопок диалогового окна завершения работы. Это позволяет запланировать время завершения работы компьютера;

- **Новый вход** — позволяет войти под другим именем пользователя, не завершая текущего сеанса;
- **Закрыть** — закрывает это диалоговое окно.

Кроме того, завершить работу системы можно и в терминале с помощью команд:

- `sudo reboot` — перезагрузка;
- `sudo poweroff` — выключение питания.
- `sudo shutdown [-r] <чч:мм|now>` — позволяет выключить или перезагрузить (если указана опция `-r`) систему в указанное время или прямо сейчас (если вместо времени указано `now`). Время задается в 24-часовом формате — например, команда: `sudo shutdown -r 17:30` перезагрузит компьютер в 17:30.

ГЛАВА 4

ASTRA LINUX®

Загрузка системы глазами администратора

- Особенности MBR и UEFI
- Настройка загрузчика GRUB2
- Система инициализации systemd
- Журналирование системы

4.1. Особенности MBR и UEFI

Теперь посмотрим на процесс загрузки системы глазами администратора. Здесь процесс кажется немного сложнее, чем просто выбор загрузочной метки и ввод пароля.

Начнем с самого начала — с включения питания. Все мы знаем, что на материнской плате компьютера имеется чип памяти, в котором находится «прошивка» (firmware) — специальная программа, которая выполняет тестирование аппаратных компонентов компьютера и запуск загрузчика операционной системы, — это если в двух словах...

На старых компьютерах эта прошивка называлась BIOS (Basic Input Output System, базовая система ввода/вывода), а на новых называется UEFI (Unified Extensible Firmware Interface, унифицированный расширяемый интерфейс микропрограммного обеспечения — это если перевести дословно).

Прошивка, помимо прочего, содержит программу настройки (SETUP). Для входа в SETUP на стационарных компьютерах обычно достаточно было нажать клавишу <Delete>, а вот для этого приходится выполнять ритуал танцев с бубном, который зависит от производителя ноутбука. Так, на ноутбуках для входа в SETUP часто используется клавиша <F2>, но могут быть и совсем другие комбинации (как правило, информация об этом есть на сайте производителя).

Любой пользователь хотя бы раз заходил в SETUP своего компьютера — скорее всего, для изменения порядка его загрузки (выбора загрузочного устройства) при установке или переустановке операционной системы (ОС). Раздел программы SETUP, в котором выбирается нужный порядок, называется **Boot Sequence** или

Boot Order. Обычно здесь надо выбрать жесткий диск, если ОС уже установлена. А вот если вам нужно установить или переустановить ОС, то следует выбрать носитель, на котором находится инсталлятор ОС, — как правило, это загрузочная флешка. И флешку эту нужно подключить к компьютеру до начала процесса загрузки, иначе ее не окажется в списке **Boot Order**, и вы не сможете ее выбрать.

Прошивка выполняет процедуру POST (Power-On Selft Test) — это процедура самотестирования, определяющая «здоровье» аппаратных компонентов. Если все здоровы, прошивка «берет» первое устройство из списка **Boot Order** и пытается на нем найти загрузчик операционной системы.

Если на вашем компьютере до сих пор используется старая схема разметки MBR (Master Boot Record, главной загрузочной записи), которая была характерна на компьютерах с BIOS, то первые 512 байтов на жестком диске как раз и занимает MBR. То есть MBR — это область на жестком диске, содержащая загрузчик операционной системы и таблицу разделов. Поскольку размер, который был в свое время выделен для таблицы разделов, составлял всего 64 байта, на жестком диске компьютера разрешалось создать всего четыре раздела. Если вам нужно было иметь на жестком диске больше разделов, создавался один *первичный* раздел (как правило, он содержал операционную систему и был помечен как загрузочный) и один *расширенный* — который сам содержал информацию о разделах, т. е. свою таблицу разделов. Внутри расширенного раздела можно было создать несколько *логических* разделов, чтобы получить в общей сложности 15 разделов, к которым может обращаться ядро Linux.

Эта схема позволяла при использовании MBR создать на диске более 4 разделов. Поскольку данные в этой записи хранились в 32-битных значениях и размер сектора по умолчанию составлял 512 байтов, этими ограничениями и определялся максимальный размер раздела — 2 Тбайт. Во времена, когда появилась MBR (а это начало 80-х годов прошлого века), такой размер казался чем-то из области научной фантастики. Это сейчас диском на 3 Тбайт никого не удивишь...

Начиная с 2010 года на новых компьютерах вместо BIOS стали прошивать UEFI. Это происходило не сразу, поэтому ваш компьютер даже 2011 года выпуска может быть еще с BIOS. Прошивка UEFI принесла другую схему разметки диска — GPT (GUID Partition Table). Эта таблица разделов была создана с запасом на долгое время:

- максимальный размер раздела — 8 Зебибайт (ZiB): 1024×1024×1024×1024 Гбайт;
- соответственно ограничения в 2 терабайта на раздел больше нет;
- GPT может адресовать до 128 разделов;
- больше нет надобности различать первичный, расширенный и логический разделы — пространства для хранения информации о разделах более чем достаточно;
- в GPT используется 128-битный глобальный уникальный идентификатор (GUID) для идентификации разделов;
- резервная копия GPT хранится в конце диска — на случай, если что-то случится с основной таблицей разделов.

В начале накопителя, использующего схему разметки GPT, создается загрузочный раздел небольшого размера, в который и помещается код загрузчика. На рис. 4.1показана таблица разделов в стиле DOS (она же MBR), а на рис. 4.2 — таблица разделов GPT. Как можно видеть, в GPT-варианте есть раздел размером 1 Мбайт — здесь и «живет» загрузчик операционной системы.

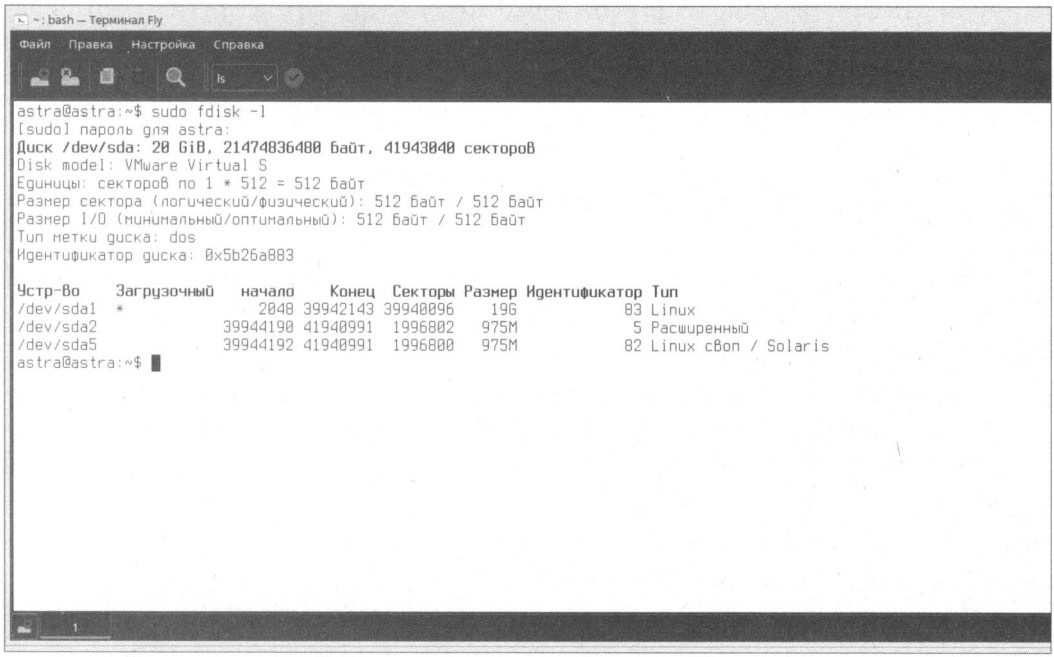

Рис. 4.1. Astra Linux: схема разметки MBR

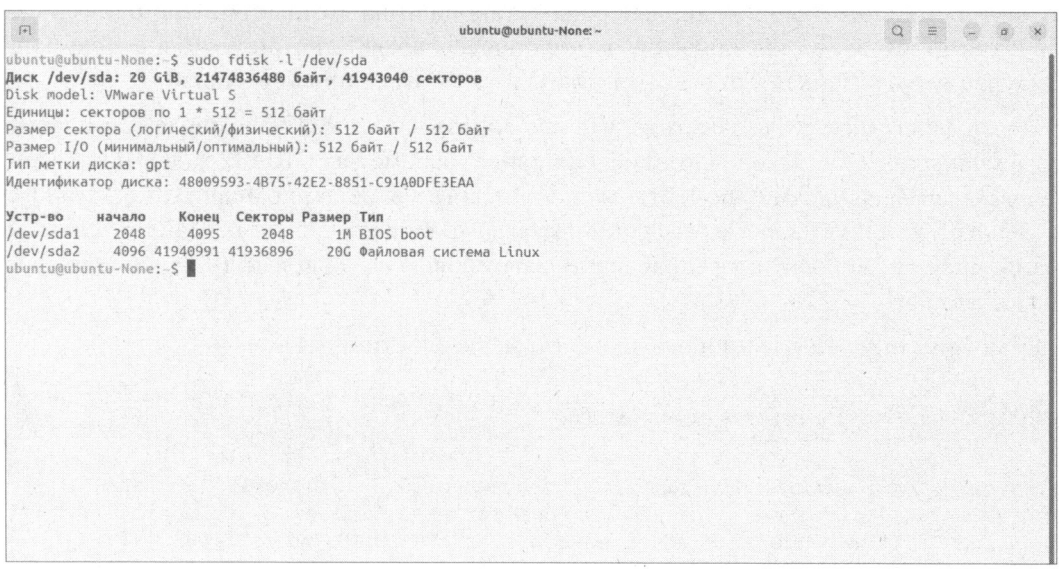

Рис. 4.2. Ubuntu: схема разметки GPT

4.2. Настройка загрузчика GRUB2

Главное назначение загрузчика — запуск выбранной пользователем операционной системы, и единственный загрузчик, с которым вам придется иметь дело на сегодняшний день, — GRUB2.

Загрузчик GRUB2, помимо загрузки Linux, может загружать и другие операционные системы, и в частности Windows. Но сегодня, когда ОС Linux стала вполне удобной для повседневного использования, необходимость иметь на компьютере две операционные системы отпала, однако возможность такая есть.

4.2.1. Конфигурационный файл GRUB2

Основным конфигурационным файлом загрузчика GRUB2 является файл /boot/grub/grub.cfg. Он весьма большой (и с каждым годом становится все больше и больше), поэтому в книге я приводить его не стану. Если вам захочется его увидеть, вы можете сделать это на своем компьютере.

Конфигурационный файл /boot/grub/grub.cfg, как правило, не редактируют вручную — для его создания используется утилита /usr/sbin/grub-mkconfig, которая генерирует этот файл на основе шаблонов, хранящихся в каталоге /etc/grub.d, и настроек из файла /etc/default/grub.

Впрочем, при особом желании и понимании того, что вы делаете, можно редактировать этот файл и без каких-либо утилит.

В конфигурационном файле /boot/grub/grub.cfg содержится описание поведения GRUB2 (что нам совсем не интересно), а также элементов загрузочного меню. Собственно, ради элементов загрузочного меню часто и возникает необходимость отредактировать этот файл. Например, вы установили на компьютер еще один дистрибутив Linux, и его инсталлятор не «прописал» новый дистрибутив в файле конфигурации загрузчика (или вы сами отказались от этого при установке).

Открыв файл grub.cfg, вы заметите, что его синтаксис весьма напоминает синтаксис bash-сценариев. Как уже было отмечено ранее, параметры GRUB2 задаются в файле /etc/default/grub, а сами элементы меню описаны в файлах, хранящихся в каталоге /etc/grub.d. Утилита grub-mkconfig/grub2-mkconfig «собирает» из этих файлов единый файл grub.cfg, корректируя поведение загрузчика на основании параметров из /etc/default/grub.

Рассмотрим описание типичного элемента меню (листинг 4.1).

Листинг 4.1. Фрагмент загрузочной метки

```
menuentry 'AstraLinuxCE GNU/Linux, with Linux 5.15.0-70-generic' --class
astralinuxce --class gnu --class os --unrestricted $menuentry_id_option
'gnulinux-5.15.0-70-generic-advanced-8c39e0af-c808-48d1-02aa89240931' {
        load_video
        insmod gzio
```

```
        if [ x$grub_platform = xxen ]; then insmod xzio; insmod lzopio; fi
        insmod part_msdos
        insmod ext2
        set root='hd0,msdos1'
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-
          efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1  0cfcfdfc-d3e4-4755-
          a0ea-5d44470dee4f
        else
          search --no-floppy --fs-uuid --set=root 0cfcfdfc-d3e4-4755-a0ea-
          5d44470dee4f
        fi
        echo 'Loading Linux 5.15.0-70-generic'
        linux   /boot/vmlinuz-5.15.0-70-generic root=UUID=0cfcfdfc-d3e4-4755-
            a0ea-5d44470dee4f ro net.ifnames=0
        echo 'Loading initial ramdisk...'
        initrd /boot/initrd.img-5.15.0-70-generic
}
```

В кавычках после `menuentry` приводится описание элемента меню — можете заменить этот текст на все, что вам больше нравится. После названия элемента меню следуют различные необязательные параметры — в других дистрибутивах, они, как правило, могут не указываться.

Далее прописаны команды GRUB2. Например, команда `insmod ext2` загружает модуль ext2. Но это не модуль ядра Linux! Это модуль GRUB2 — файл ext2.mod, находящийся в каталоге /boot/grub.

Команда `set root` устанавливает загрузочное устройство. Формат имени устройства такой же, как в случае с GRUB.

> ***Внутреннее имя устройства GRUB***
>
> Мы знаем, что даже ATA-диски в новых дистрибутивах имеют имена вида /dev/sda*. Но команда `set root` загрузчика GRUB2 содержит имя `hd`. Это не опечатка! Это внутреннее имя устройства GRUB, а не имя системного устройства.

После служебного слова `linux` задается ядро (файл ядра) и параметры, которые будут переданы ядру. Служебное слово `initrd` указывает на файл initrd.

Теперь рассмотрим файл /etc/default/grub, содержащий параметры GRUB2 (листинг 4.2). Поскольку этот файл вы будете редактировать чаще, чем grub.cfg, то комментарии для большего удобства я перевел на русский язык.

Листинг 4.2. Файл /etc/default/grub

```
# Если вы измените этот файл, введите команду 'update-grub'
# для обновления вашего файла /boot/grub/grub.cfg.

# Элемент по умолчанию, нумерация начинается с 0
GRUB_DEFAULT=0
```

```
# Чтобы увидеть меню GRUB, надо или закомментировать следующую опцию,
# или установить значение больше 0, но в этом случае нужно изменить
# значение GRUB_HIDDEN_TIMEOUT_QUIET на false
GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
# Тайм-аут (в секундах)
GRUB_TIMEOUT="5"
# Название дистрибутива - вывод команды lsb_release или просто Debian
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
# Параметры ядра по умолчанию
GRUB_CMDLINE_LINUX_DEFAULT="quiet net.ifnames=0"
GRUB_CMDLINE_LINUX=""

# Раскомментируйте для отключения графического терминала
# (только для grub-pc)
#GRUB_TERMINAL=console

# Разрешение графического терминала
#GRUB_GFXMODE=640x480

# Раскомментируйте следующую опцию, если вы не хотите передавать
# параметр "root=UUID=xxx" ядру Linux
#GRUB_DISABLE_LINUX_UUID=true

# Раскомментируйте, если нужно отключить генерацию элемента меню
# режима восстановления
#GRUB_DISABLE_LINUX_RECOVERY="true"
# Раскомментируйте, чтобы получить гудок при запуске GRUB
#GRUB_INIT_TUNE="480 440 1"
```

После изменения файла **/etc/default/grub** не забудьте выполнить команду `update-grub` для обновления вашего файла **/boot/grub/grub.cfg**. Команда `update-grub` — это просто сценарий, вызывающий утилиту **grub-mkconfig/grub2-mkconfig** и передающий ей параметр — имя выходного файла (по умолчанию **/boot/grub/grub.cfg**). Ничего не мешает вам вызвать утилиту **grub-mkconfig/grub2-mkconfig**[1] вручную, но использовать команду `update-grub` более удобно.

РЕДАКТИРОВАНИЕ ВРУЧНУЮ ФАЙЛА GRUB.CFG

Чуть ранее было сказано, что файл *grub.cfg* не следует редактировать вручную. Да это и разумно — если не знаешь, что делаешь, лучше использовать утилиту *grub-mkconfig* или команду `update-grub`, но когда понимаешь, о чем речь, дело движется быстрее... В любом случае при желании его редактировать можно, но только если вы уверены в своих силах...

[1] Сначала утилита **grub2-mkconfig** называлась **grub-mkconfig**, но, видимо, чтобы подчеркнуть ее принадлежность именно к GRUB2, она была переименована.

Таким образом, при редактировании конфигурации GRUB2 нужно придерживаться одной стратегии из двух возможных:

- согласно первой вы редактируете файл grub.cfg вручную и не используете утилиту grub2-mkconfig или команду `update-grub`;
- вторая стратегия заключается в использовании вспомогательных программ, но тогда не нужно редактировать файл grub.cfg вручную, иначе при последующем изменении файла grub.cfg утилитой grub2-mkconfig или командой `update-grub` все изменения, внесенные вручную, будут уничтожены. Поступая согласно второй стратегии, нужно редактировать файлы из каталога /etc/grub.d (там содержатся файлы, формирующие загрузочное меню) и файл /etc/default/grub, содержащий общие параметры GRUB2.

Вызов утилиты grub2-mkconfig

По умолчанию утилита *grub2-mkconfig* генерирует конфигурационный файл на консоль, поэтому вызывать ее нужно так:

```
sudo grub2-mkconfig > /boot/grub/grub.cfg
```

4.2.2. Передача параметров ядра при загрузке

Как уже было отмечено в *главе 3*, для редактирования параметров ядра нужно выбрать загрузочную метку и нажать клавишу <e>. Если вы работаете в версии Special Edition, то вам будет предложено ввести имя пользователя и пароль (рис. 4.3). Пароль — это пароль загрузчика, который устанавливается при установ-

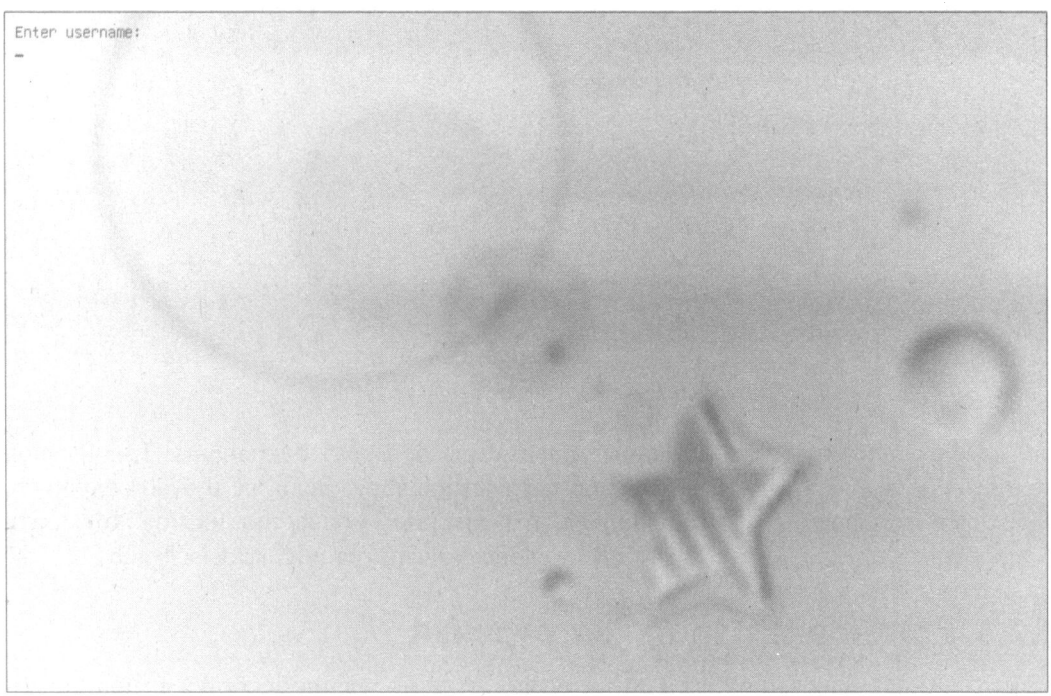

Рис. 4.3. Ввод имени пользователя (вызов редактора загрузчика)

ке системы. А вот имя пользователя — это имя первого пользователя, которого вы создали при установке. В идеале пароли учетной записи и загрузчика должны отличаться друг от друга.

Если имя пользователя и пароль правильные, тогда вы увидите редактор конфигурации загрузчика. При использовании Common Edition пароль не запрашивается (если вы, конечно, не установили его самостоятельно, — далее будет рассказано, как это сделать).

На рис. 4.4 показано, что были добавлены параметры ядра `rw init=/bin/bash` (после служебного слова linux). Для загрузки с этой конфигурацией нужно нажать клавишу <F10>.

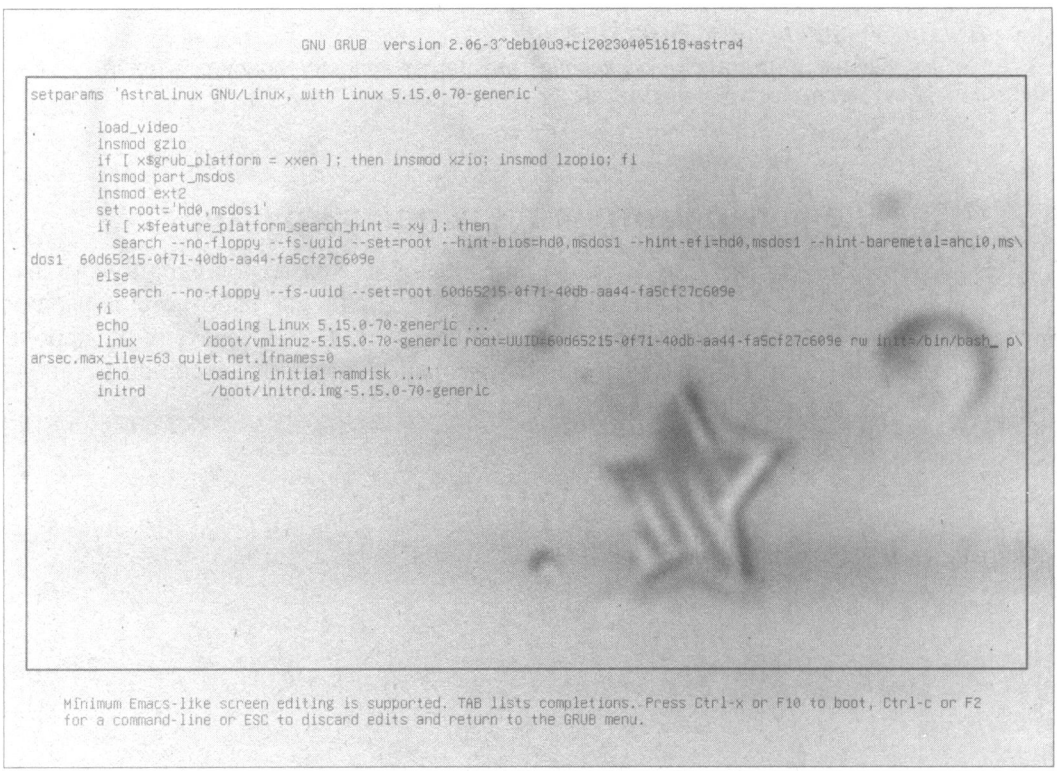

Рис. 4.4. Редактор конфигурации загрузчика

Вы должны помнить, что указанные параметры не будут сохранены. Такой трюк можно использовать для тестирования параметров ядра. Если же нужно сохранить исправленные параметры, чтобы они загружались по умолчанию, их можно указать в директиве `GRUB_CMDLINE_LINUX_DEFAULT` файла конфигурации /etc/default/grub.

4.2.3. Установка пароля загрузчика

Теперь самое время защитить наш загрузчик. По умолчанию любой желающий может изменить параметры ядра. Достаточно злоумышленнику передать ядру пара-

метры `rw, single` или `rw, init=/bin/bash`, и после загрузки он сможет сделать с системой все, что захочет, — например, изменить пароль root. А получив root-доступ, настроить систему так, как ему это выгодно, или полностью уничтожить ее (хотя это можно было бы сделать и на этапе загрузки).

Поэтому мы должны защитить загрузчик паролем. Собственно загрузка операционных систем будет осуществляться без пароля, однако если кто-то захочет изменить параметры ядра, то у него ничего не получится, — загрузчик попросит ввести пароль. Для самых «продвинутых» доброжелателей, которые смогут подключить жесткий диск вашего компьютера к Windows-системе и с помощью, например, файлового менеджера Total Commander просмотреть конфигурационный файл GRUB2, мы закодируем наш пароль с помощью алгоритма MD5 — это самый стойкий алгоритм шифрования на сегодняшний день. Поэтому, даже если злоумышленник и просмотрит конфигурационный файл загрузчика, пароль он все равно не узнает.

По сравнению с ранее использовавшимся загрузчиком GRUB, загрузчик GRUB2 одновременно и проще в обращении, и сложнее в настройке. Настраивать GRUB2 придется реже, но к его сложной настройке надо привыкнуть, — практически все современные дистрибутивы перешли на GRUB2.

В GRUB можно было задать общий пароль для всех загрузочных меток, а также установить пароль только на некоторые загрузочные метки. В GRUB2 можно сделать то же самое, но, кроме самого пароля, понадобится указать еще и имя пользователя (логин), что усложняет злоумышленнику взлом системы, поскольку ему нужно будет знать не только пароль, но и то, кому он принадлежит. Защита отдельных загрузочных меток, как правило, используется редко, — чаще устанавливается пароль на все метки сразу, что и будет продемонстрировано далее.

Сначала установим простой (незашифрованный) пароль, а затем зашифруем его, чтобы никто не смог его прочитать, загрузившись с LiveCD. Прежде всего, откройте файл /etc/grub.d/07_password:

```
sudo nano /etc/grub.d/00_header
```

В конец файла добавьте строки:

```
cat << EOF
set superusers="user"
password user 12345
EOF
```

> ***Примечание***
>
> Если файл */etc/grub.d/07_password* не существует, его нужно создать!

Здесь имя пользователя `user`, пароль мы придумали для примера такой: `12345`.

Теперь обновите GRUB2:

```
sudo grub2-mkconfig > /boot/grub/grub.cfg
```

Можно также напрямую редактировать файл конфигурации GRUB2 grub.cfg. В него следует добавить вот такие строки:

```
set superusers="user1"
password user1 password1
password user2 password2
```

Обратите внимание, что командами `password` заданы два пользователя: `user1` и `user2` с паролями `password1` и `password2` соответственно. Но пользователь `user1` является суперпользователем, т. е. может редактировать загрузочные метки GRUB2, а обычный пользователь (`user2`) может только загружать метки. Таким образом, у пользователя `user1` получится передать ядру новые параметры, а пользователь `user2` сможет лишь загрузить Linux с параметрами по умолчанию.

Можно даже задать условие, что метку Windows будет загружать только пользователь `user2`:

```
menuentry "Windows" --users user2 {
        set root=(hd0,2)
        chainloader +1
}
```

Теперь разберемся с шифрованием пароля. Команда `password` поддерживает только незашифрованные пароли. Если вы хотите использовать зашифрованные пароли, нужно применить команду `password_pbkdf2`. Например:

```
password_pbkdf2 user зашифрованный_пароль
```

Получить зашифрованный пароль можно командой:

```
grub-mkpasswd-pbkdf2
```

Программа запросит у вас пароль (придумайте и введите пароль в ответ на запрос), закодирует его и выведет на экран хеш (шифр) введенного вами пароля:

```
Your PBKDF2 is grub.pbkdf2.зашифрованный_пароль
```

Вот пример такого шифра:

```
grub.pbkdf2.sha512.10000.9290F727ED06C38BA4549EF7DE25CF5642659211B7FC076F2D28FE
FD71784BB8D8F6FB244A8CC5C06240631B97008565A120764C0EE9C2CB0073994D79080136.887C
FF169EA8335235D8004242AA7D6187A41E3187DF0CE14E256D85ED97A97357AAA8FF0A3871AB9EE
FF458392F462F495487387F685B7472FC6C29E293F0A0
```

Весь этот хеш нужно скопировать в конфигурационный файл GRUB2:

```
password_pbkdf2 user
grub.pbkdf2.sha512.10000.9290F727ED06C38BA4549EF7DE25CF5642659211B7FC076F2D28FE
FD71784BB8D8F6FB244A8CC5C06240631B97008565A120764C0EE9C2CB0073994D79080136.887C
FF169EA8335235D8004242AA7D6187A41E3187DF0CE14E256D85ED97A97357AAA8FF0A3871AB9EE
FF458392F462F495487387F685B7472FC6C29E293F0A0
```

Если вы не использовали файл 07_password, а редактировали непосредственно файл grub.cfg, то команду `update-grub` вводить не нужно!

Еще раз, коротко. Для изменения пароля загрузчика GRUB2 нужно выполнить следующие действия:

1. Сгенерировать хеш пароля командой `grub-mkpasswd-pbkdf2`.
2. Полученный хеш скопировать в буфер обмена (выделите его, щелкните правой кнопкой мыши и выберите **Копировать**).
3. Создать файл /etc/grub.d/07_password или открыть его, если он существует.
4. Вставить в файл инструкции:

```
cat << EOF
set superusers="username"
password_pbkdf2 username grub.pbkdf2.sha512.10000.9C319610666.....
EOF
```

5. Установить права 700 для этого файла:

```
sudo chmod 700 /etc/grub.d/07_password
```

6. Обновить загрузчик командой `update-grub`.

4.2.4. Восстановление загрузчика GRUB/GRUB2

Что делать, если вы переустановили на своем компьютере Windows, а она установила в MBR вместо GRUB2 свой загрузчик, и теперь вы не можете загрузить Linux? Не переустанавливать же еще и Linux из-за такой мелочи!

Для восстановления загрузчика GRUB/GRUB2 нужно загрузиться с LiveCD (подойдет любой LiveCD с любым дистрибутивом Linux) и ввести следующие команды:

```
mkdir /old
mkdir /old/dev
mount /dev/sdaN /old
```

Все команды следует вводить от имени root, для чего использовать команды `su` или `sudo`.

В частности, в LiveCD Ubuntu нужно вводить все команды с использованием команды `sudo` — например, так:

```
sudo mkdir /old
sudo mkdir /old/dev
...
```

Разберемся, что означают эти команды:

- первая из них создает каталог /old, который будет использоваться в качестве точки монтирования;
- вторая — создает в этом каталоге подкаталог dev, который пригодится для монтирования devfs — псевдофайловой системы;
- третья — служит для монтирования корневой файловой системы дистрибутива Linux, установленного на жестком диске в разделе /dev/sda*N* (где *N* — номер раздела), к каталогу /old. Предположим, что на вашем компьютере дистрибутив

Linux был установлен в раздел /dev/sda5. Тогда вам нужно ввести следующую команду:

```
mount /dev/sda5 /old
```

После этого надо подмонтировать каталог /dev к каталогу /old/dev — это делается с помощью все той же команды `mount`, но с параметром `--bind`:

```
mount --bind /dev /old/dev
chroot /old
```

Команда `chroot` заменяет корневую систему нашего LiveCD на корневую систему дистрибутива, установленного на винчестере. Вам остается лишь ввести команду:

```
/sbin/grub2-install /dev/sda
```

Эта команда установит загрузчик GRUB/GRUB2 так, как он был установлен до переустановки Windows.

После установки загрузчика следует перезагрузить компьютер командой `reboot`.

4.2.5. Запрет загрузки в режиме восстановления. Задание тайм-аута

Для запрета загрузки в режиме восстановления выполните следующие действия:

1. Откройте файл /etc/default/grub командой:

   ```
   sudo nano /etc/default/grub
   ```

2. Раскомментируйте в нем строку: `GRUB_DISABLE_RECOVERY="true"`.
3. При желании отредактируйте параметр `GRUB_TIMEOUT` — можно установить его в 0, чтобы по умолчанию загружалась Linux (точнее, метка по умолчанию) и меню загрузчика не отображалось.
4. Для скрытия загрузчика с сохранением возможности переключения клавишей <Shift> на этапе выбора ОС/ядра к параметру `GRUB_TIMEOUT=0` следует добавить: `GRUB_HIDDEN_TIMEOUT=5`.
5. Сохраните изменения.
6. Введите команду: `sudo update-grub`

4.3. Система инициализации systemd

Загрузчик GRUB2 загружает ядро Linux и передает ему управление. Ядро, в свою очередь, должно запустить систему инициализации. Раньше в качестве системы инициализации использовалась SysVinit, но на сегодняшний день она безнадежно устарела и больше не соответствует требованиям современных дистрибутивов. Во всех современных дистрибутивах сейчас используется система инициализации systemd.

4.3.1. Сервисы, службы, демоны

Служба (*англ.* service — сервис) — это программа, выполняющаяся в фоновом режиме и предоставляющая определенный сервис. Например, служба apache2 предоставляет сервис веб-сервера. Она в фоновом режиме прослушивает заданный в настройках порт (чаще всего 80 или 443) и обслуживает поступающие на него запросы, отправляя в ответ на запрос HTML-код веб-страницы.

Ранее сервисы также назывались *демонами* (daemon). Именно поэтому в названии некоторых служб присутствует буква d — например, демон, обслуживающий снапы, называется snapd.

Для управления сервисами (автоматический запуск, перезапуск, остановка) как раз и служит система инициализации. Как уже отмечено ранее, в древних дистрибутивах Linux использовалась система инициализации init.

Программа init — это процесс с UID 1, он первым запускается ядром и выступает родителем для всех процессов, у которых нет собственного родителя. Основная задача такого процесса (помимо всех остальных задач) — инициализация системы. А значит, он должен это сделать быстро. Как запустить систему быстро? Во-первых, запускать не все, что можно, а только самое необходимое. Во-вторых, запускать сервисы параллельно. Существует еще одна система инициализации, запускающая сервисы параллельно, — это upstart (но она больше не используется, хотя и неплоха сама по себе).

Давайте разберемся теперь, как запустить минимум сервисов, а запуск всех остальных отложить до тех пор, пока они не понадобятся. В некоторых случаях мы знаем, какие сервисы нам понадобятся заранее. Как правило, это syslog, dbus и т. д. Но представьте, что на своем ноутбуке мы хотим передавать и принимать файлы по Bluetooth. Нужен нам демон bluetoothd? Да, потому что мы хотим использовать Bluetooth. Но в настоящий момент, пока мы ничего не передаем и не принимаем, он не нужен. То есть, пока не включен Bluetooth-адаптер (и пока одно из пользовательских приложений не захочет с ним общаться через D-Bus), загружать сервис bluetoothd не требуется. Аналогично мы послупаем и в отношении системы печати cups — хотя принтер и подключен к компьютеру, но сервис cupsd нужен лишь тогда, когда происходит печать. То же и для сетевых сервисов — пока никто не обращается к вашему веб- или FTP-серверу, нет необходимости их запускать, что сэкономит не только лишние секунды при запуске системы, но и снизит нагрузку на процессор в процессе работы системы. Да и память сбережет.

Теперь поговорим о параллельном запуске нужных нам сервисов. Современные процессоры настолько мощные, что система инициализации может попытаться запустить все нужные сервисы одновременно. Этим она полностью загрузит имеющиеся ресурсы, но и сократит время запуска системы.

Однако запустить все и сразу нельзя — необходимо синхронизировать запуск сервисов. Иначе получится, что сервису Б требуется сервис А, который еще не запустился. «Вес» сервисов разный, действия, выполняемые при запуске, — тоже разные. Даже если вы сначала запустите сервис А, а потом — Б, сервис Б может запус-

титься быстрее базового сервиса А. Приведу для конкретики некоторые примеры: практически всем службам нужен syslog (иначе, как они будут вести протоколирование?), поэтому им необходимо дождаться его запуска. Многим службам (например, тому же Avahi) нужен D-Bus, поэтому, пока D-Bus не будет запущен, сервису Avahi приходится ждать.

В результате на практике получается, что большая часть сервисов запускается все равно последовательно, а не параллельно, и выигрыш времени по сравнению с init — мизерный.

Как снять ограничения синхронизации? Для этого нам надо понять, что нужно сервису Б от сервиса А, и как он вообще проверяет, что сервис А запущен? Оказывается, сервису Б всего лишь нужен *сокет сервиса* А (socket service). А что такое сокет сервиса? Это всего лишь файл. Например, все сервисы, которым нужен D-Bus, ждут возможности подключения к файлу /var/run/dbus/system_bus_socket. Все, кому нужен syslog, ждут возможности подключения к устройству /dev/log и т. д.

По сути, всё что нужно — это сделать доступными сокеты сервисов до запуска самих сервисов. А когда сервис фактически запустится, мы можем передать ему сокет с помощью команды `exec()`. Получается, что для параллельного запуска всех демонов нам сначала необходимо создать для них все сокеты, а потом параллельно запустить все демоны.

Что произойдет в нашей ситуации, когда сервис Б требует запуска сервиса А? Ничего страшного — сервис Б станет в очередь и будет ждать, пока сервис А запустится. Главное, что сокет сервиса открыт.

А что если сервис А (пусть это будет syslog) требуется сразу нескольким сервисам: Б, В, Г и Д? Тоже не страшно. Каждый из этих сервисов отправит свое сообщение в буфер сокета /dev/log, затем запустится syslog и обработает все эти сообщения.

Как видите — все гениальное просто. Но не нужно думать, что это какое-то совсем новое изобретение. Подобная система инициализации работает в macOS — там она называется launchd. Но поскольку не все знакомы с macOS, эти идеи Apple известны не многим.

4.3.2. Терминология systemd

Система инициализации systemd контролирует всю систему — отсюда ее название. В настоящее время она используется в последних версиях популярных дистрибутивов: Fedora, openSUSE, Ubuntu, а также и Astra Linux.

Система systemd построена на концепции *модулей* (units). У каждого модуля есть свое имя и тип. Например, модуль типа nscd.service управляет сервисом (демоном) nscd.

Основными типами модулей являются:

- *service* (сервис) — демоны, которые можно запустить, перезапустить, остановить;

- *socket* (сокет) — реализует сокет, расположенный в файловой системе или в Интернете. Поддерживаются сокеты AF_INET, AF_INET6, AF_UNIX. У каждого сокета есть связанный с ним сервис. Например, при попытке установки соединения с сокетом nscd.socket будет запущен сервис nscd.service. Вам это ничего не напоминает? А я вспоминаю старый суперсервер inetd и его более новую версию xinetd — они работали именно так;
- *device* (устройство) — реализует устройство в дереве устройств. Если устройство описано через правила udev (системы управления устройствами для новых версий ядра Linux), то его можно представить в systemd как модуль типа device;
- *mount* (точка монтирования) — реализует точку монтирования в файловой системе. Демон systemd контролирует все точки монтирования, их подключение и отключение. Теперь файл /etc/fstab не главный, а служит дополнительным источником информации о точках монтирования, хотя вы по-прежнему можете описывать в нем свои точки монтирования;
- *automount* (автоматическая точка монтирования) — реализует автоматическое монтирование файловой системы. Такой модуль имеет соответствующий ему модуль типа mount, который будет запущен, как только файловая система станет доступной;
- *target* (цель) — служит для логической группировки модулей других типов. Этот тип модуля очень важен, но в то же время он ничего не делает, а просто группирует другие модули. В systemd больше нет уровней запуска (которые были в init), вместо них используются *цели*. Например, цель multi-user.target описывает, какие сервисы (точнее модули, а не только сервисы) должны быть запущены в многопользовательском режиме. По сути, цель multi-user.target аналогична 3-му уровню запуска;
- *snapshot* (снимок) — также ничего не делает, а только ссылается на другие модули. Снимки используются в двух случаях: первый случай — временный перевод системы в какое-то состояние (например, в однопользовательский режим) и последующий возврат из этого состояния, второй — поддержка режима suspend. Многие демоны не могут правильно переходить в этот режим, поэтому в ряде случаев их лучше остановить и запустить заново после того, как система проснется.

Приведем основные особенности systemd:

- позволяет контролировать для каждого процесса: среду исполнения, ограничение ресурсов, рабочий каталог, корневой каталог, umask, параметр nice, ID пользователя и группы и многое другое;
- синтаксис файлов конфигурации systemd очень похож на синтаксис файлов .desktop и поэтому будет знаком многим Linux-пользователям;
- наличие совместимости со сценариями SysV (правда, не могу пока сказать — временной или постоянной). Если есть старая (например, /etc/init.d/avahi) и новая (/etc/systemd/system/avahi.service) конфигурации, то используется новая;
- для удобства и обратной совместимости поддерживается и обрабатывается файл /etc/fstab;

- совместимость с /dev/initctl. На практике это означает, что многие системные команды вроде `poweroff` или `shutdown` будут работать с новой системой systemd;
- монтирование виртуальных файловых систем, установка имени узла — все это и многое другое делается теперь без shell-сценариев;
- состояние сервиса может контролироваться через D-Bus.

Конечно, это далеко не все особенности systemd, но остальные знать и не обязательно.

4.3.3. Управление службами в Astra Linux

Для управления сервисами из консоли используется утилита systemctl. Например, вот как можно запустить, остановить и перезапустить сервис:

```
systemctl start <название>
systemctl stop <название>
systemctl restart <название>
systemctl enable <название>
systemctl disable <название>
```

Команда `enable` включает автоматическую загрузку сервиса, а `disable` — отключает ее.

Для перезапуска веб-сервера служат команды:

```
systemctl start apache2
systemctl stop apache2
systemctl restart apache2
```

Однако Astra Linux содержит очень удобный графический конфигуратор, позволяющий управлять сервисами более просто:

1. Откройте панель управления.
2. Перейдите в раздел **Система**.
3. Откройте конфигуратор **Инициализация системы**.
4. Для управления службой щелкните правой кнопкой мыши — в появившемся меню вы увидите следующие команды (рис. 4.5):
   - **Запустить юнит** — запускает службу;
   - **Остановить юнит** — останавливает службу.
   - **Перезапустить юнит** — перезапускает службу;
   - **Перечитать конфигурацию юнита** — заставляет службу перечитать свой файл конфигурации. Полезно, если вы внесли незначительные изменения в файл конфигурации и теперь хотите, чтобы они вступили в силу, но при этом перезагружать весь сервис нежелательно, чтобы не прерывать обслуживание пользователей;
   - **Включить юнит** — включает автоматическую загрузку службы;

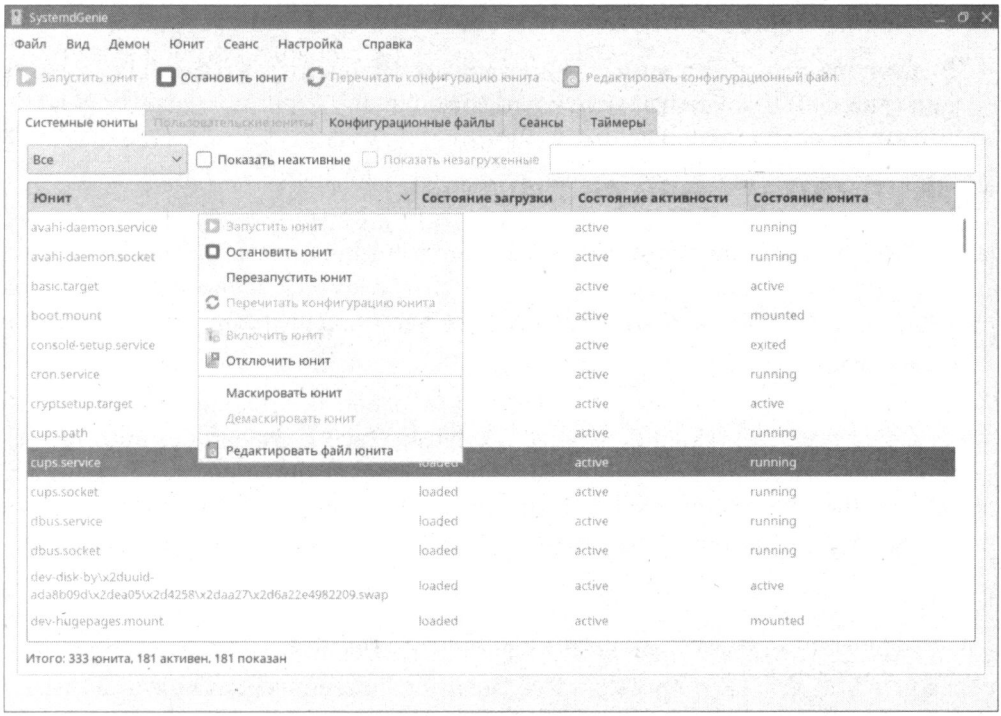

Рис. 4.5. Меню управления сервисами

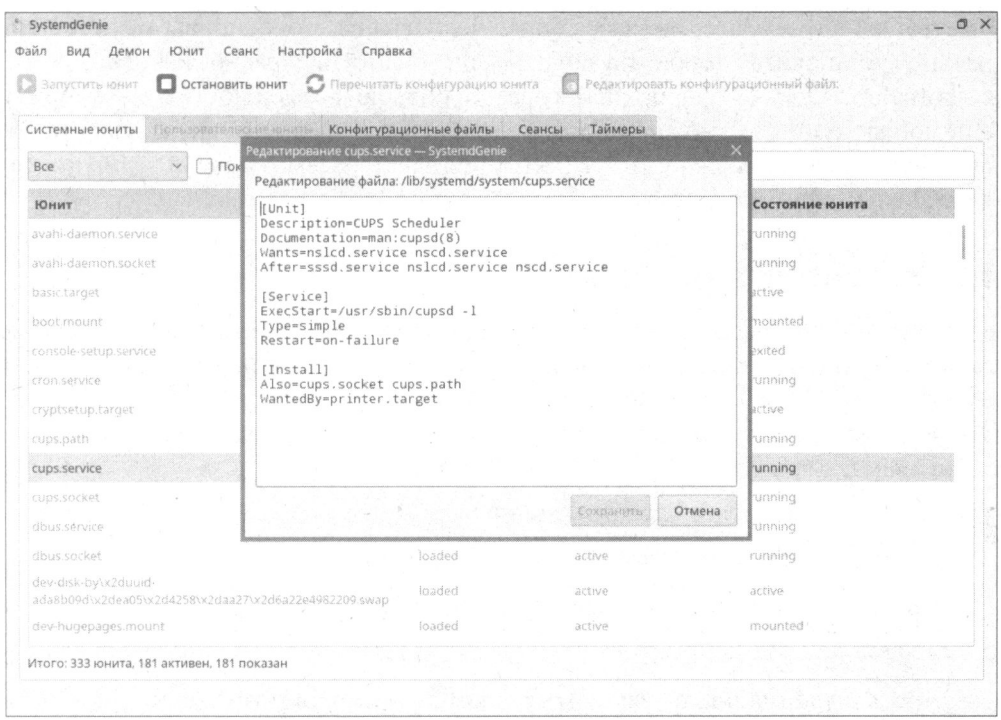

Рис. 4.6. Файл конфигурации сервиса

- **Отключить юнит** — отключает автоматическую загрузку службы;
- **Редактировать файл юнита** — позволяет отредактировать файл конфигурации (рис. 4.6) и показывает его расположение.

4.4. Журналирование системы

В любой UNIX-подобной системе, коей является и Linux, имеются так называемые *демоны протоколирования* (далее просто демоны). Демоны записывают в протоколы (журналы, логи) сообщения, генерируемые ядром, сервисами, пользовательскими программами.

Если вы уже работали с Linux, то знаете, что ранее протоколирование системы осуществляли демоны syslogd и rsyslogd. Сейчас же все устроено немного иначе — во многих современных дистрибутивах протоколированием системы занимается сама система инициализации systemd, а точнее — ее сервис systemd-journald.service. При этом запрос системного лога (журнала) организуется через утилиту journalctl.

Вот что нужно знать о протоколировании в современных дистрибутивах:

- все журналы по-прежнему хранятся в каталоге /var/log;
- серверы (WWW, FTP и пр.) могут создавать собственные каталоги/файлы журналов в каталоге /var/log;
- для просмотра системных журналов используется утилита journalctl — привычные файлы вроде /var/log/messages более недоступны. Конечно, вы можете параллельно установить демон rsyslogd, и он будет прекрасно работать в паре с journalcrl. Однако у journalctl гораздо больше возможностей — например, с помощью опции `-b` можно просмотреть логи текущей или предыдущей загрузки. Некоторые возможности утилиты journalctl рассмотрены в этой книге, а с остальными вы сможете ознакомиться в справочной системе;

4.4.1. Установка времени

Первое, что нужно сделать, — это установить правильный часовой пояс. Основной недостаток syslogd/rsyslogd заключался в том, что эти демоны сохраняли записи в журнале без учета часового пояса, и было непонятно, когда именно произошло то или иное событие. В journalctl этот недостаток устранен — можно использовать как местное время, так и UTC.

Для выбора часового пояса служит команда:

```
$ timedatectl set-timezone <часовой пояс>
```

Просмотреть список часовых поясов можно командой:

```
$ timedatectl list-timezones
```

Просмотреть информацию о текущем часовом поясе можно командой:

```
$ timedatectl status
```

4.4.2. Просмотр и фильтрация логов

Для просмотра логов введите команду `sudo journalctl` — будет выведен огромный список различных записей. Использовать команды постраничного вывода вроде `more` необходимости нет, поскольку подобные средства просмотра журнала уже встроены в саму утилиту journalctl (рис. 4.7).

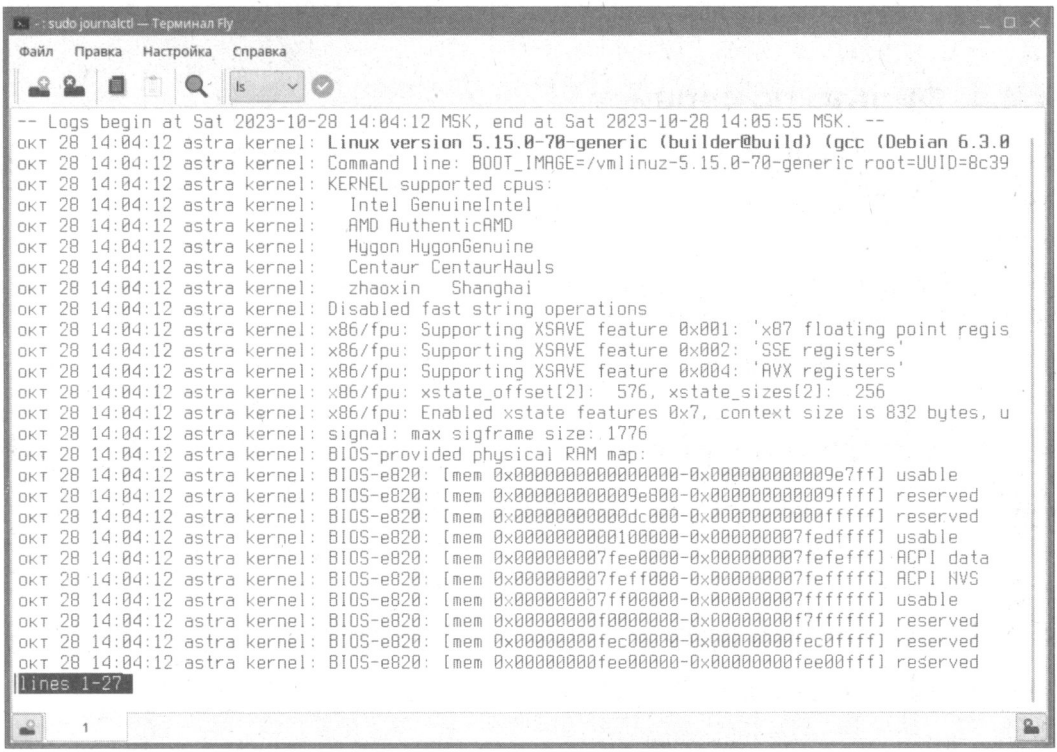

Рис. 4.7. Вывод команды `journalctl`

Обратите внимание на самую первую строку — она говорит, с какого момента начинается ведение логов. Как правило, это дата установки системы. Понятно, что с самого начала будет очень много записей, и их как-то нужно фильтровать.

4.4.3. Фильтр по дате

Фильтрацию журналов можно выполнить и по дате — для этого используются опции `--since` и `--until`. Например, для просмотра логов начиная с 11.10.24 7:00 введите команду:

```
$ journalctl --since "2024-10-11 07:00:00"
```

Если вы указали опцию `--since`, но не указали дату, будет взята текущая дата. Если указана дата, но не указано время, будет взято время 00:00:00.

Еще несколько примеров:

```
$ journalctl --since yesterday
$ journalctl --since 09:00 --until now
$ journalctl --since 10:00 --until "1 hour ago"
```

Первая команда показывает логи, начиная со вчерашнего дня и по текущий момент. Вторая — отображает журналы, начиная с 9 утра и по текущий момент. Третья — начиная с 10 утра и до прошлого часа (т. е. если сейчас 19:00, то до 18:00).

4.4.4. Фильтр по сервису

Выполнить фильтрацию можно и по определенному сервису (когда нужно просмотреть не все журналы, а только определенной службы) — например:

```
$ journalctl -u nginx.service
```

Тип фильтрации можно комбинировать — например, следующая команда выводит журналы nginx начиная со вчерашнего дня:

```
$ journalctl -u nginx.service --since today
```

4.4.5. Фильтр по пути

Просмотреть журналы какого-то процесса можно путем указания пути к нему:

```
$ journalctl /usr/bin/docker
```

4.4.6. Фильтр по процессу или пользователю

Можно отфильтровать журнал по PID процесса:

```
$ journalctl _PID=<ИД процесса>
```

А также и по UID пользователя:

```
$ journalctl _UID=33
```

Узнать UID пользователя можно так:

```
$id -u <имя пользователя>
```

4.4.7. Просмотр сообщений ядра

Для просмотра сообщений ядра используйте опции `-k` или `--dmesg`:

```
$ sudo journalctl -k
```

Эта команда покажет все сообщения ядра для текущей загрузки. Такую команду можно комбинировать с опцией `-b`, чтобы просмотреть сообщения ядра во время предыдущей загрузки:

```
$ sudo journalctl -k -b -2
```

4.4.8. Фильтр по уровню ошибки

В journalctl, как и в syslogd, используется та же класификация уровней ошибок:

- 0 — EMERG (система неработоспособна);
- 1 — ALERT (требуется немедленное вмешательство);
- 2 — CRIT (критическое состояние);
- 3 — ERR (ошибка);
- 4 — WARNING (предупреждение);
- 5 — NOTICE (просто обратите внимание);
- 6 — INFO (информационное сообщение);
- 7 —DEBUG (отложенная печать).

Например:

```
$ sudo journalctl -p err -b
```

Эта команда выводит все ошибки при текущей загрузке.

4.4.9. Журналы в реальном времени

Журналы системы можно просматривать в реальном времени. Для этого используется опция `-f`:

```
$ sudo journalctl -f
```

ASTRA LINUX®

ГЛАВА 5

Командная строка

- Способы доступа к командной строке
- Автодополнение командной строки
- Получение справки по команде
- Некоторые базовые команды Linux

5.1. Способы доступа к командной строке

Многие задачи администрирования системы подразумевают использование командной строки. Хотя разработчики Astra Linux предоставили набор достаточно удобных конфигураторов, позволяющих выполнить настройку системы, все же умение использовать командную строку существенно расширяет возможности администратора.

Самый простой способ получить доступ к командной строке — это использовать приложение **Терминал Fly** (рис. 5.1). Терминал предоставляет доступ к командной строке из графического окна — т. е., по сути, вы сможете использовать консоль, не выходя из графического режима.

Второй способ — использовать *консоли* — более радикальный и пригодится пользователям, которые хотят почувствовать, что такое «True Linux».

Операционная система Linux предоставляет всем пользователям 6 консолей (вообще-то их может быть 12, но в современных дистрибутивах 6, поскольку есть еще и графический интерфейс), которые пронумерованы от 1 до 6 (tty1–tty6). Для переключения на конкретную консоль нажмите комбинацию клавиш <Ctrl>+<Alt>+ +<F*n*>, где *n* — номер консоли. То есть для переключения на первую консоль надо нажать <Ctrl>+<Alt>+<F1> (рис. 5.2). Для возврата обратно нажмите комбинацию клавиш <Alt>+<F7> (правильно, без <Ctrl>).

5.2. Автодополнение командной строки

Работа в консоли заключается во вводе нужной команды — вы вводите команду (например, создания каталога, просмотра файла, вызова редактора и т. д.) и нажи-

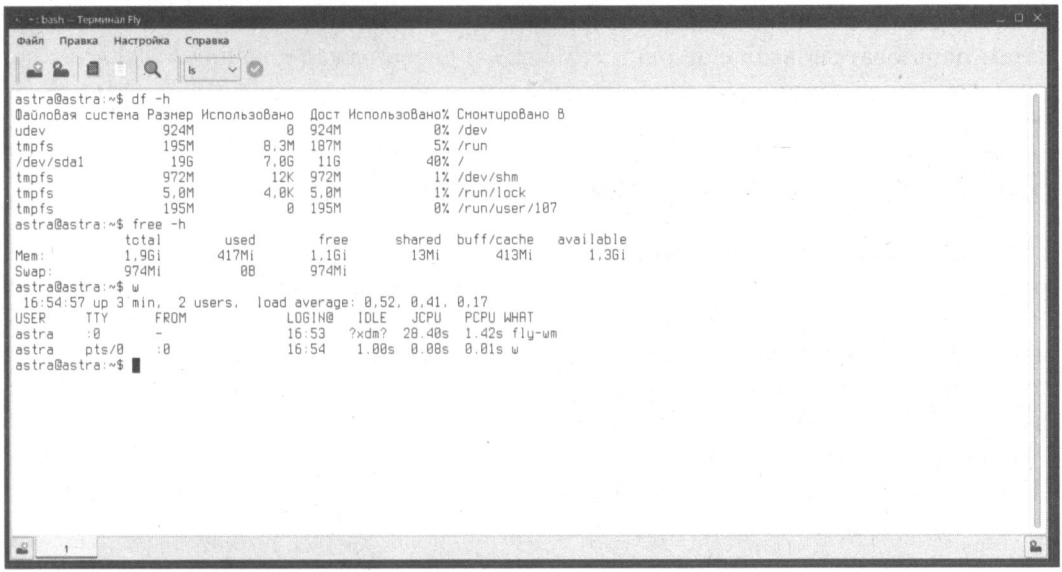

Рис. 5.1. Терминал Fly

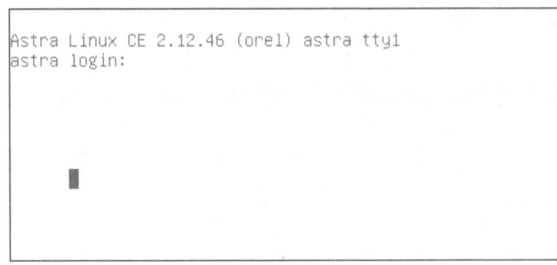

Рис. 5.2. Первая консоль (tty1)

маете клавишу <Enter>. Команда содержит как минимум имя запускаемой программы. Кроме имени программы команда может содержать параметры, которые будут переданы программе, а также символы перенаправления ввода/вывода (об этом чуть позже). Естественно, вам нужно знать имя программы, а также параметры, которые необходимо ей передать.

Посмотрите на рис. 5.3, чтобы понять, как работает автодополнение. Пользователь ввел первые символы команды: `na` — и нажал клавишу <Tab>. Система предоста-

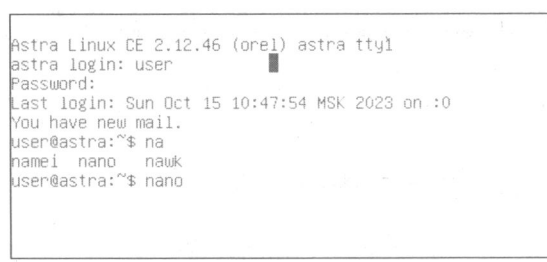

Рис. 5.3. Автодополнение в действии

вила список доступных команд, начинающихся с этих символов: `namei`, `nano`, `nawk`. Затем пользователь ввел еще один символ `n` и снова нажал клавишу <Tab> — система дополнила команду до единственной возможной — `nano`.

5.3. Получение справки по команде

Если вы помните название программы, а назначение параметров забыли, поможет команда `man`. Man (от *англ.* manual) — это справочная система Linux. В ней имеется информация о каждой программе, которая установлена в системе. Откуда система знает обо всех программах? Всё очень просто — разработчики программ под Linux договорились, что вместе с программой будет поставляться специальный man-файл — файл справочной системы. Понятно, если разработчик недобросовестный, он может и не создать файл справочной системы, но так происходит очень редко. И чтобы получить справку по какой-либо программе, нужно ввести команду (рис. 4.4):

`man` *имя_программы*

К сожалению, справочная система в Astra Linux не вся русифицирована, и вам понадобится знание английского языка, хотя бы в зачаточном состоянии.

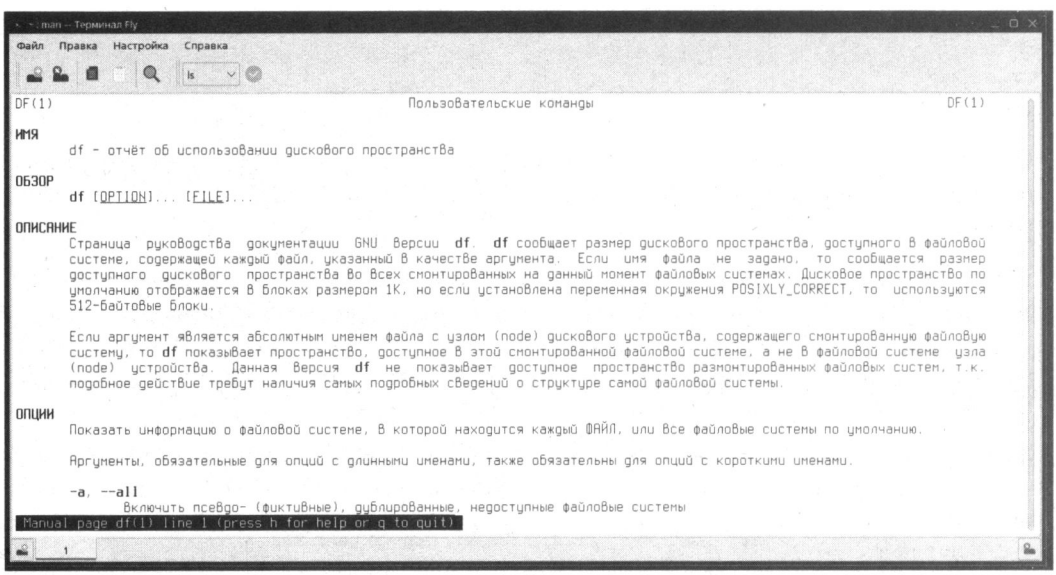

Рис. 5.4. Справочная система Man (вывод в терминале Fly)

5.4. Некоторые базовые команды Linux

Надеюсь, приведенная в этом разделе информация сделает вашу работу в командной строке максимально комфортной.

5.4.1. Перенаправление ввода/вывода

С помощью *перенаправления ввода/вывода* мы можем перенаправить вывод одной программы в файл или на стандартный ввод другой программы. Например, у вас не получается настроить сеть, и вы хотите перенаправить вывод команды `ifconfig` в файл, а затем разместить этот файл на форуме, где вам помогут разобраться с проблемой. Можно также командой `ps -ax` перенаправить список всех процессов компьютера команде `grep`, которая найдет в списке интересующий вас процесс.

Рассмотрим следующую команду:

```
echo "some text" > file.txt
```

Символ > означает, что вывод команды, находящейся слева от этого символа, будет записан в файл, находящийся справа от символа, при этом файл будет перезаписан.

Чуть ранее мы говорили о перенаправлении вывода программы `ifconfig` в файл. Соответствующая команда будет выглядеть так:

```
ifconfig > ifconfig.txt
```

Если вместо > указать >>, то исходный файл не будет перезаписан, а вывод команды добавится в конец файла:

```
echo "some text" > file.txt
echo "more text" >> file.txt
cat file.txt
some text
more text
```

Кроме символов > и >> для перенаправления ввода/вывода часто употребляется вертикальная черта |. Предположим, что мы хотим вывести содержимое файла big_text:

```
cat big_text
```

Но в файле big_text много строк, они быстро проскочат по экрану, и мы ничего не успеем прочитать. Следовательно, целесообразно отправить вывод команды `cat` какой-то программе, которая будет выводить файл на экран постранично, например:

```
cat big_text | more
```

Конечно, этот пример не очень убедительный, потому что для постраничного вывода гораздо удобнее команда `less`:

```
less big_text
```

Вот еще один интересный пример. Допустим, мы хотим удалить файл file.txt без запроса — для этого можно указать команду:

```
echo y | rm file.txt
```

Команда `rm` должна была бы запросить подтверждение удаления (нужно было бы нажать клавишу <Y>), но за нас это сделает команда `echo`.

И еще один пример. Пусть имеется большой файл, и нам нужно найти в нем все строки, содержащие подстроку 555-555. Чтобы не делать это вручную, можно воспользоваться командой:

```
cat file.txt | grep "555-555"
```

5.4.2. Команда *clear* — очистка экрана

Команда `clear` очищает экран при работе в консоли (терминале).

Пример ее использования:

```
$ clear
```

5.4.3. Команды *free* и *df* — информация о системных ресурсах

Команда `free` выводит информацию об использовании оперативной и виртуальной памяти, а `df` — об использовании дискового пространства. На рис. 5.5 показано, что в тестовой системе 2 Гбайт памяти, из них используется 410 Мбайт, свободно 1,1 Гбайт. Также в этой же системе имеются два раздела на диске /dev/sda: sda1 и sda2, первый из них — 26 Гбайт, второй — 359 Мбайт. Первый используется в качестве корневой файловой системы (/), второй — для хранения файлов загрузчика (/boot).

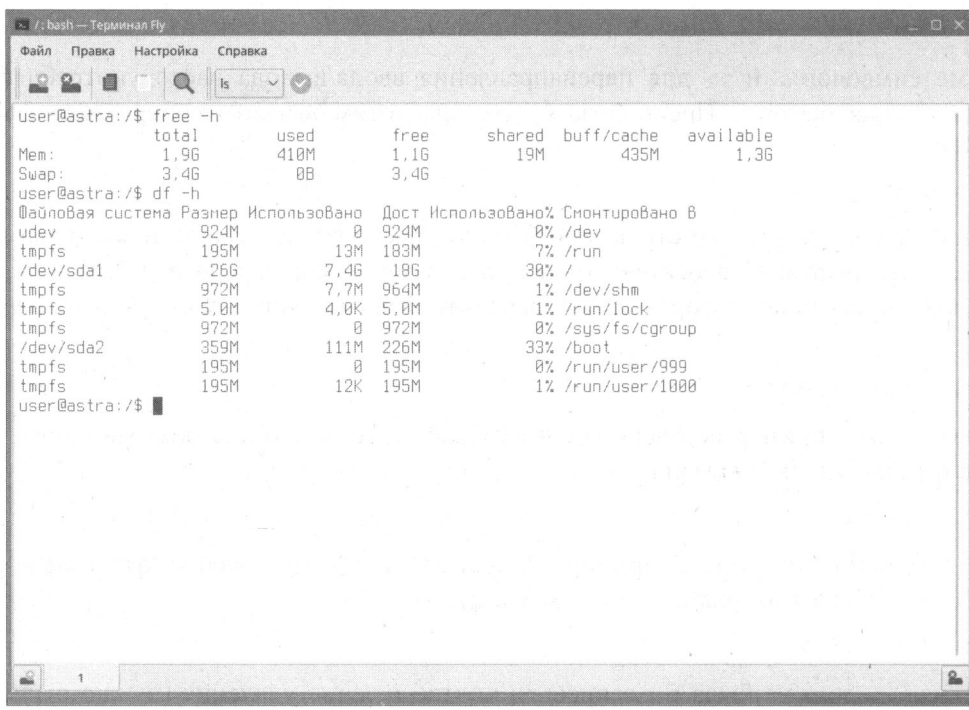

Рис. 5.5. Информация о системных ресурсах

Параметр -h

Параметр `-h`, присутствующий в обеих командах, показанных на рис. 5.5, обеспечивает вывод в удобном для чтения человеком формате. По умолчанию обе команды выводят информацию о ресурсах в байтах, но человеку проще ориентироваться в гигабайтах, мегабайтах и т. д.

5.4.4. Команды *w*, *who* и *whoami* — информация о пользователях

Эти три родственные команды выводят следующую информацию (рис. 5.6):

- команда `w` — список пользователей, подключенных к системе, виртуальный терминал, с которого работает пользователь, время входа в систему для каждого пользователя, статистику использования системы (IDLE — время простоя, JCPU — использование процессора), выполняемые каждым пользователем задачи;
- команда `who` — список пользователей, подключенных к системе, время и дату входа каждого пользователя;
- команда `whoami` — имя пользователя, который ввел команду.

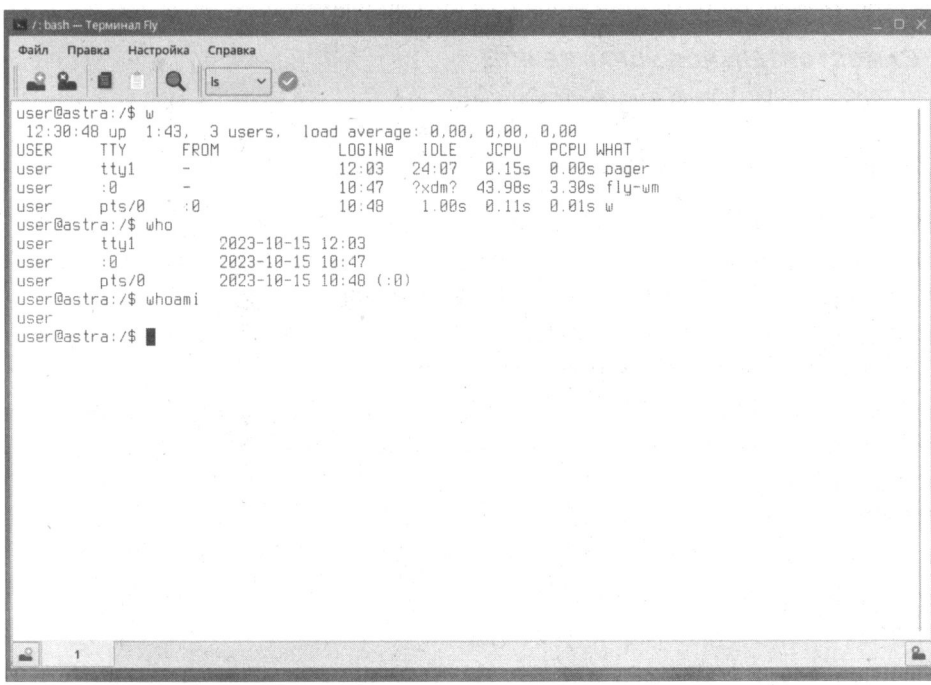

Рис. 5.6. Команды `w`, `who`, `whoami`

Небольшая сага о терминалах

Когда-то, во времена суперкомпьютеров, которые назывались мейнфреймами и занимали немалую площадь, к ним подключались пользовательские терминалы. Такие терминалы назывались *физическими* и состояли из клавиатуры и монитора. Хотя в то

время они назывались просто терминалами. Да, к одному мейнфрейму могло подключаться несколько терминалов.

Когда же операционная система UNIX постепенно перекочевала на персональные компьютеры, где монитор и клавиатура были всего в одном комплекте, она все равно оставалась многопользовательской и многозадачной. Тогда и было введено понятие *виртуального терминала* (tty). Для переключения между виртуальными терминалами использовались клавиши <Alt>+ <F*n*>. Начиная с 1993 года многие дистрибутивы обзавелись графическим интерфейсом. Поначалу набор графических программ был небольшой, но обязательно присутствовала программа-терминал, подобная современному Fly. Когда вы ее запускаете, с точки зрения системы это приравнивается к отдельному входу в систему, но уже с псевдотерминала.

Посмотрите на рис. 5.6. Пользователь `user` вошел как с виртуального терминала (`tty1`), так и с псевдотерминала (`pts/0`). А вот запись `:0` означает, что пользователь вошел, используя графический менеджер входа, т. е. он попросту задействовал графический интерфейс (экран с номером 0).

5.4.5. Команды *top* и *htop* — вывод информации о запущенных процессах

Команда `top` выводит информацию о запущенных процессах и о ресурсах системы, которые потребляются запущенными процессами. Команда `htop` — более информативный аналог `top`, но не устанавливается по умолчанию.

САМОСТОЯТЕЛЬНОЕ УПРАЖНЕНИЕ

Установите программу `htop` командой:

```
sudo apt install htop
```

Подробнее об установке программ мы поговорим далее.

Вывод команды `htop` приведен на рис. 5.7:

- количество ядер процессора — 2. Обратите внимание на два псевдографических индикатора **1** и **2**, показывающих нагрузку на каждое ядро;
- индикатор **Mem** показывает информацию об использовании оперативной памяти, а **Swp** — об использовании виртуальной памяти;
- индикатор **Tasks** — информацию о количестве запущенных задач и потоков (**thr**). В текущий момент запущен (находится в состоянии **running**) один (**1**) процесс, а именно — сама команда `htop`. Остальные процессы находятся в состоянии sleep;
- индикатор **Load average** — информацию о средней нагрузке на систему;
- индикатор **Uptime** — информацию о том, сколько система работает: 1 час 49 минут;
- далее приведена информация о запущенных процессах, отсортированная по использованию процессорного времени. Колонки списка процессов означают следующее:
  - **PID** — уникальный идентификатор процесса;
  - **USER** — имя пользователя, запустившего процесс;
  - **PRI** — внутренний приоритет ядра, обычно это **NI** + 20;

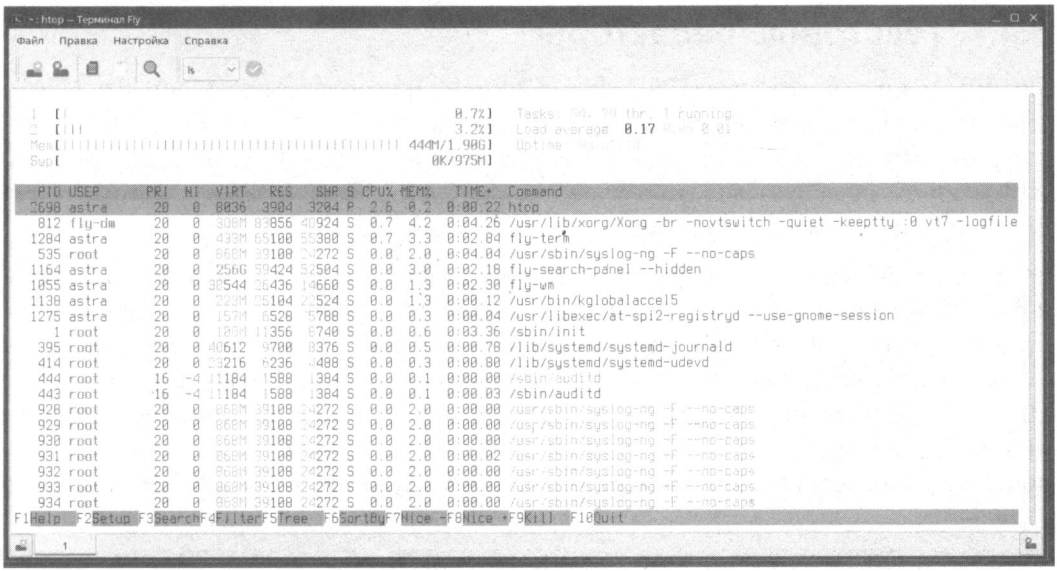

Рис. 5.7. Команда `htop`

- **NI** — информация о приоритете выполняемого процесса от 19 (самый низкий приоритет) до –20 (наивысший приоритет);
- **VIRT** — использование виртуальной памяти;
- **RES** — резидентный объем процесса (текст + данные + стек);
- **SHR** — размер разделяемых страниц;
- **S** (STATE) — состояние процесса: S — сон, R — запуск, I — простой, Z — зомби и др.;
- **CPU%** — процент использования процессорного времени;
- **MEM%** — процент использования памяти;
- **TIME+** — количество времени, проведенного процессом в пользовательском режиме (все, кроме системных вызовов), измеренное в тиках часов;
- **Command** — команда, используемая при запуске процесса.

Примечание

Нужно отметить, что Astra Linux очень эффективно использует оперативную память, что и показано на рис. 5.7, — при объеме ОЗУ 2 Гбайт файл подкачки не используется вообще.

5.4.6. Команды *more* и *less* — постраничный вывод

Большой текстовый файл намного удобнее просматривать с помощью команд `less` или `more`. Программа `less` (если она есть в вашей системе) удобнее, чем `more`:

```
sudo tac /var/log/messages | grep ppp | less
```

5.4.7. Текстовые редакторы

По умолчанию во многих UNIX/Linux-системах используется очень неудобный текстовый редактор vi. Я уверен, что если вы попытаетесь к нему обратиться, это напрочь отпугнет вас от Linux. Он создавался в начале 70-х годов прошлого века (когда никто не задумывался об удобстве использования), а затем почему-то мигрировал в Linux и каким-то образом дожил до наших дней.

Но самое печальное, что во многих дистрибутивах он используется по умолчанию в различных системных утилитах вроде visudo или crontab. К счастью, разработчики Astra Linux действительно молодцы, и при запуске таких утилит вам предоставляется возможность выбора редактора, который вы будете использовать (рис. 5.8). Ни в коем случае не выбирайте vim.basic, иначе попросту не сумеете завершить процесс редактирования без изучения справочной системы. Оптимальный выбор — редактор nano, который и предлагается в списке под номером 1 (рис. 5.9).

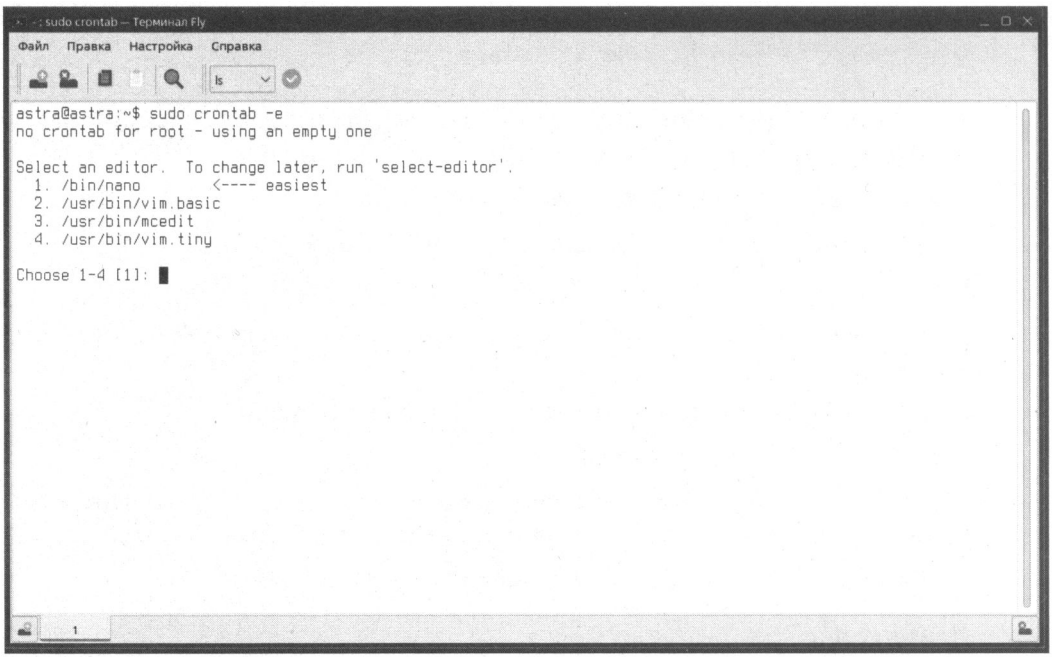

Рис. 5.8. Выбор текстового редактора

Как его использовать, думаю, всем понятно. Символ ^ в строке меню означает нажатие клавиши <Ctrl> — т. е., например, чтобы сохранить файл, нужно нажать комбинацию клавиш <Ctrl>+<O>.

Astra Linux содержит удобные графические редакторы текста. А вот текстовые вам придется использовать редко — разве что для незначительного редактирования того или иного файла конфигурации. Поэтому все, что вам нужно здесь знать, — это: <Ctrl>+<O> — запись, а <Ctrl>+<X> — выход из редактора.

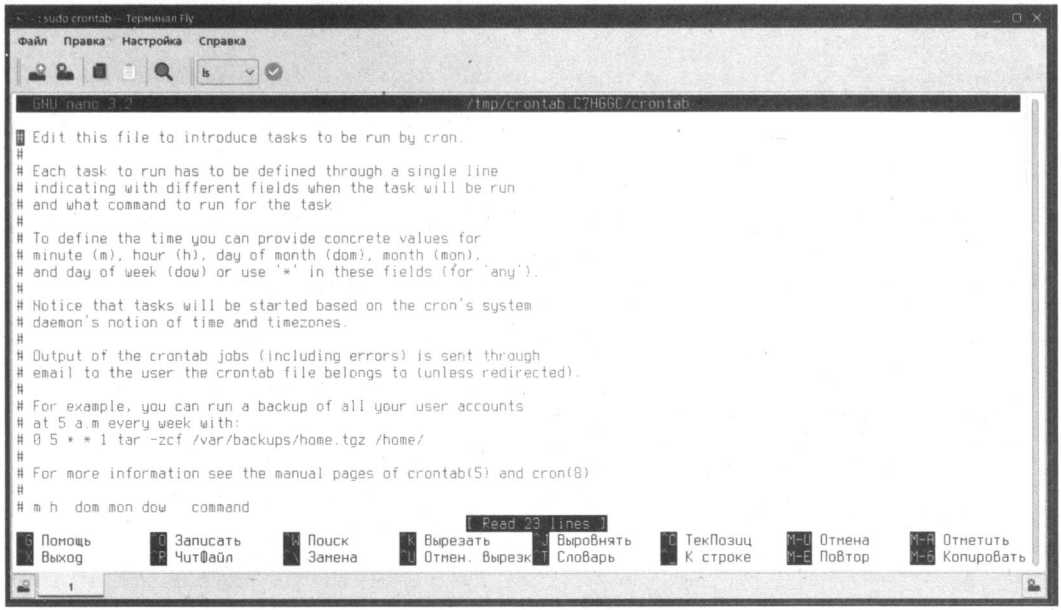

Рис. 5.9. Текстовый редактор nano

ЧАСТЬ II

Сеть и Интернет

ASTRA LINUX®

ГЛАВА 6

Настройка проводного соединения с Интернетом

- Локальная сеть с использованием технологии Gigabit Ethernet
- Настройка подключения по локальной сети
- Используем сторонние DNS

6.1. Локальная сеть с использованием технологии Gigabit Ethernet

6.1.1. Стандарты Gigabit Ethernet

Сетевых технологий существует много, но в этой книге мы займемся настройкой локальной сети, построенной по технологии Gigabit Ethernet. Зато рассмотрим мы ее полностью: от обжатия кабеля до конфигурирования сети.

> ***Примечание***
>
> Сегодня беспроводные сети практически вытеснили проводные варианты. А использование проводных часто сводится к подключению ближайших устройств уже готовыми патч-кордами к роутеру Wi-Fi или коммутатору, если он есть.

Основные характеристики стандарта Gigabit Ethernet:

- скорость передачи данных: 1000 Мбит/с;
- метод доступа к среде передачи данных: CSMA/CD;
- среда передачи данных: витая пара UTP/STP (категорий 5, 5e, 6, 7), оптоволоконный кабель;
- максимальное количество компьютеров: 1024;
- максимальная длина сети: 200 м (UTP/STP)

Здесь нужно понимать, что имеется в виду под длиной сети. Максимальная длина сегмента, т. е. участка кабеля между коммутатором (активным сетевым оборудованием) и конечным пользователем, — 100 метров. Если предположить, что к коммутатору подключены два пользователя и они находятся каждый на расстоянии 100 метров от коммутатора (в разные стороны), то получим как раз длину

в 200 метров. На практике расстояние от коммутатора до компьютера будет, разумеется, меньше. Хотя бы по причине того, что кабель по пути к пользователю проходит через различные конструкции и преграды (стены, мебель, перекрытия), и если по прямой между коммутатором и компьютером может быть расстояние три метра, то с учетом периметра помещения и расположения компьютера длина кабеля может составлять 14–15 метров.

Таблица 6.1 содержит характеристики различных стандартов сети Gigabit Ethernet.

Таблица 6.1. *Характеристики стандартов Gigabit Ethernet*

Стандарт	Тип	Максимальная длина сегмента, м.	Тип кабеля
IEEE 802.3z	1000Base-CX	25	UTP/STP, кат. 5, 5e, 6
	1000Base-LX	одномод — 5000 многомод — 550	оптоволокно
	1000Base-SX	550	оптоволокно
IEEE 802.3ab	1000Base-T	100	UTP/STP, кат. 5, 5e, 6, 7
TIA 854	1000Base-TX	100	UTP/STP, кат. 6,7
IEEE 802.3ah	1000Base-LX10	10 000	оптоволокно
	1000Base-BX10	10 000	оптоволокно
IEEE 802.3ap	1000Base-KX	1	для объединительной платы
Проприетарный, нет IEEE-стандарта	1000BASE EX	40 000	оптоволокно
Проприетарный, нет IEEE-стандарта	1000Base-ZX	70 000	оптоволокно

Первые версии сети Ethernet работали со скоростью до 10 Мбит/с, затем появились стандарты Fast Ethernet, гарантирующие передачу данных на скорости до 100 Мбит/с. Эти стандарты все еще актуальны, и в продаже можно найти много, пусть и не очень свежих, но очень дешевых устройств, поддерживающих Fast Ethernet.

Стандарты Gigabit Ethernet также не очень свежие. Так, самый древний из них — IEEE 802.3z — появился в 1998 году, а самый свежий — IEEE 802.3ap — в 2007-м. А по-настоящему популярной эта технология стала только в последние несколько лет. Сегодня никого не удивишь дома гигабитной сетью — и роутеры, и сетевые адаптеры поддерживают Gigabit Ethernet и при этом весьма доступны по цене. В качестве магистральных используются стандарты 10 Gigabit Ethernet (10GbE) и 100 Gigabit Ethernet (100GbE), которые обеспечивают передачу данных со скоростью 10 и 100 Гбит/с. Эти стандарты тоже не прямо вчера были приняты. Например, IEEE 802.3ba (100GBase-*) был принят в 2010 году. Однако нужно понимать, что с момента выпуска стандарта до его воплощения в виде реального оборудования на полках в магазинах могут пройти годы.

6.1.2. Нужна ли сейчас проводная сеть?

Все еще нужна. На предприятии, где количество компьютеров составляет несколько десятков, а к ним еще столько же смартфонов сотрудников, беспроводная сеть перестает «вытягивать». Поэтому серверы, а также хотя бы ближайшие к «компьютерной комнате» (она же серверная) компьютеры стараются подключать кабелем. Это когда речь идет о современных офисах. А если предприятию лет 10 или еще больше, то можно с уверенностью сказать, что там повсеместно проводная сеть, а беспроводная обслуживает только смартфоны сотрудников и клиентов.

Дома такая же картина. Лет 7–10 назад в вашей беспроводной сети было несколько клиентов — ваши компьютеры (т. к. вам не хотелось портить ремонт и прокладывать витую пару) и, возможно, телевизор (если он был у вас ультрасовременным на то время). Даже смартфонами вы дома реже пользовались, т. к. удобнее было использовать компьютер или ноутбук. Сейчас же, когда к Wi-Fi подключается каждый чайник, ситуация в корне меняется. В семье из четырех человек легко может быть около 20 клиентов Wi-Fi: компьютеры, смартфоны, планшеты, IP-камеры, телевизоры, игровая приставка и прочая бытовая техника с Wi-Fi (так называемая «умная»). А если вы еще додумались развернуть систему «умного дома» без выделенного шлюза (из-за дешевизны), где каждое такое умное устройство использует Wi-Fi, то количество беспроводных клиентов может приближаться к трем десяткам. При этом вы заметите, что имеющийся роутер «не вытягивает» и пойдете покупать новый. А затем поймете, что новый хоть и работает лучше, но иногда происходят неприятные моменты, особенно при просмотре онлайн-ТВ или в компьютерных играх. Если скорость упадет даже в два раза при загрузке файла, ничего страшного не случится — он просто будет дольше загружаться. А вот ежели такое же произойдет при просмотре видео, то начнутся торможения, отставание картинки от звука и закончится все этот тем, что вы на своем 4К-телевизоре будете смотреть видео в лучшем случае в формате 480p. Конечно, вы попробуете побороться за скорость Wi-Fi, но потом вам придет в голову самое простое и самое дешевое решение: подключить ваш телевизор кабелем к роутеру. Вуаля! Скорость больше падать не будет, и никаких проблем с просмотром 4К-видео у вас не возникнет. Следующей на очередь к кабельному подключению встанет игровая приставка, если она есть. А затем вам захочется проводом подключить стоящий рядом компьютер, чтобы, когда вы будете скачивать большие файлы, это не отражалось на остальной беспроводной сети.

Именно поэтому лучше покупать роутер с четырьмя (или больше, если такой найдете) портами Ethernet. А когда войдете во вкус, купите дешевенький гигабитный коммутатор на 8 портов и подключите его к роутеру, а остальные устройства — к этому коммутатору. Вот так и появится у вас дома проводная сеть. И поверьте — ради 4К-картинки и отсутствия задержек в играх вам будет наплевать на змеящиеся по квартире провода Ethernet, тем более что есть всевозможные решения вроде кабель-каналов или плинтусов с кабель-каналами, позволяющими если не скрыть полностью, то хотя бы замаскировать провода.

6.1.3. Оборудование для проводной сети

Нужно убедиться, что компьютеры, предназначенные для соединения в сеть, оснащены *сетевыми адаптерами*, поддерживающими технологию Gigabit Ethernet. Как правило, сейчас сетевые адаптеры интегрированы в материнскую плату, и устанавливать их отдельно необходимости нет. На всех современных и не очень (а таковыми считаю компьютеры не старше 2010 года выпуска) уже есть гигабитные порты.

В крайнем случае, если у вас оказался старый компьютер, на «борту» которого адаптер Fast Ethernet, вам придется приобрести сетевой адаптер отдельно (рис. 6.1). Установка сетевого адаптера проблем не вызывает — просто вставьте его в свободный разъем шины PCI (все адаптеры Fast Ethernet выполнены в виде плат расширения именно для шины PCI). Существуют также сетевые адаптеры, подключаемые к компьютеру по USB (рис. 6.2), — стоят они не сильно дороже PCI-адаптеров, и нет никаких оснований подозревать, что при работе в Linux с ними возникнут какие-либо проблемы.

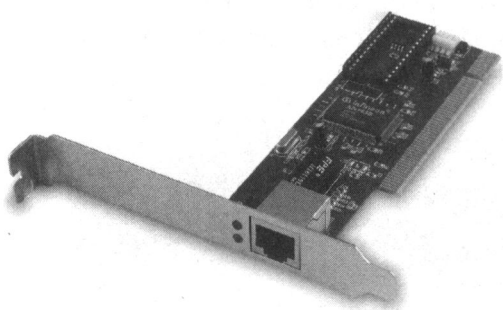

Рис. 6.1. Сетевой PCI-адаптер Fast Ethernet

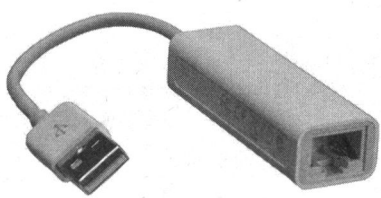

Рис. 6.2. Сетевой адаптер USB

Ясно, что устанавливать сетевой PCI-адаптер полагается при выключенном компьютере — шина PCI пока еще не поддерживает «горячую замену». Собрав и включив компьютер, подключите к сетевому адаптеру *коннектор* (специальный наконечник) сетевого кабеля.

Сетевые кабели различной длины (от 0,5 до 5 м), оснащенные коннекторами (патч-корды), можно приобрести в магазинах компьютерной техники, а можно и сделать самим, нарезав кабель на нужные отрезки и закрепив (обжав) коннекторы на их концах.

Обжимать самому или купить готовый? Все зависит от того, сколько устройств вам нужно подключить. Готовы ли вы купить хороший инструмент для обжимки кабеля? Плохой не стоит потраченных денег и долго не прослужит, а на хороший нужно потратиться.

Если вы — администратор сети предприятия, и вам нужно регулярно иметь дело с проводной сетью, лучше купить хороший инструмент для себя любимого. Если же вы домашний пользователь и вам нужно подключить несколько устройств по кабелю, проще и дешевле купить уже готовый патч-корд нужной длины. Можно

также попросить изготовить его в любом компьютерном магазине, где продается витая пара, — вам без проблем помогут. Там же вы можете заказать патч-корд на основе STP-кабеля — он лучше подходит для наружных работ, чем обычный UTP-кабель, больше пригодный для использования внутри помещения. А обычные патч-корды (уже готовые) в магазинах продаются в основном на основе UTP-кабеля. Так что если нужен STP-кабель, то или делать самому, или заказывать.

> ***Обжатие кабеля***
>
> Обжать кабель — значит особым образом закрепить на его концах специальные наконечники-коннекторы (см. далее).

Для создания сети Gigabit Ethernet вам потребуются следующие устройства:

- сетевые адаптеры — о них мы только что поговорили;
- коммутатор (switch) — его можно купить в любом компьютерном магазине. Дизайном и количеством портов коммутаторы могут отличаться друг от друга.

  На рис. 6.3 показан 24-портовый коммутатор, более подходящий для корпоративной сети (и внешним видом, и возможностью помещения в специальную стойку), нежели для дома. А для домашней сети можно найти и более симпатичное устройство;

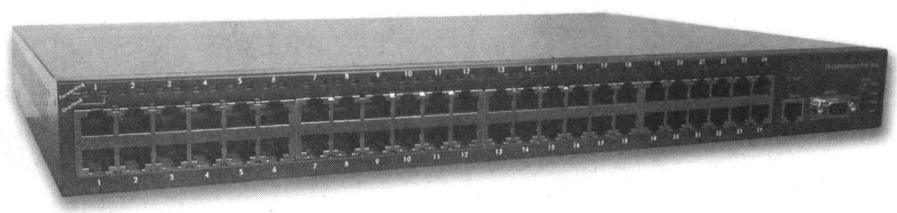

Рис. 6.3. 24-портовый коммутатор

- сетевой кабель (витая пара 5-й категории или лучше) — приобретайте именно такой тип кабеля и такой длины, чтобы нормально хватило для соединения каждого компьютера сети с коммутатором;
- коннекторы RJ-45 — таких коннекторов вам понадобится в два раза больше, чем компьютеров, поскольку каждый отрезок кабеля нужно обжать с двух концов. Но я рекомендую купить еще несколько лишних штук — если вы будете обжимать кабель впервые, думаю, без неудачных проб не обойдется. Не пожалейте пару копеек, а то придется сбегать в магазин еще раз;
- инструмент (специальные обжимные щипцы) для обжатия коннекторов витой пары — хороший инструмент стоит относительно дорого (примерно как коммутатор), а плохой, как я уже отметил ранее, лучше не покупать[1]. Если не хотите выкладываться, одолжите такие щипцы у кого-нибудь на пару дней.

[1] Помню, как-то дешевый инструмент обжимал не все жилы, и некоторые из них приходилось дожимать плоской отверткой. Удовольствие еще то, особенно когда нужно обжать более 30 коннекторов.

КАТЕГОРИЯ ВИТОЙ ПАРЫ ИМЕЕТ ЗНАЧЕНИЕ...

Сегодня гигабитными портами никого не удивишь, но если вы хотите, чтобы ваша сеть работала на скорости 1 Гбит/с, используйте витую пару категории 6 или выше. Теория гласит, что если обжаты все 4 пары, то даже при использовании витой пары категории 5е можно получить скорость 1 Гбит/с. Но это в теории. На практике же, имея в сети маршрутизатор с гигабитными портами, гигабитный коммутатор, все компьютеры с гигабитными адаптерами и кабель категории 5е, видим следующее: индикатор 1000 Mbps на коммутаторе не светится, и сеть работает в режиме 100 Мбит/с, обеспечивая передачу файлов со скоростью 11–12 Мбайт/с, что вполне нормально для 100 Мбит/с. То есть дешевый китайский кабель понизил скорость работы сети в 10 раз... А вот при переходе на витую пару категории 6 та же сеть заработала на скорости 1000 Мбит/с.

Теперь приступим к самому процессу обжатия. Внутри кабеля идут четыре витые пары проводов (всего восемь), причем у каждого провода своя цветовая маркировка. Суть процесса обжатия заключается в том, чтобы подключить каждый из проводов к нужному контакту коннектора (табл. 6.2). Для этого сначала надо вставить провода в коннектор таким образом, чтобы каждый провод зашел на всю глубину по направляющим соответствующего контакта (зачищать провода не обязательно — за вас это сделает инструмент), затем коннектор со вставленными проводами осторожно (чтобы провода из него не выпали) помещается в специальное гнездо обжимных щипцов, и их рукоятки сильно сжимаются. Используя данные табл. 6.2, вы без проблем сможете обжать кабель.

Таблица 6.2. *Обжим витой пары*

Контакт	Цвет провода	Контакт	Цвет провода
1	Бело-оранжевый	5	Бело-синий
2	Оранжевый	6	Зеленый
3	Бело-зеленый	7	Бело-коричневый
4	Синий	8	Коричневый

Осталось один конец обжатого отрезка кабеля своим коннектором подключить к коммутатору (концентратору), а второй — к сетевому адаптеру компьютера. Если вы неправильно (или несильно) обожмете кабель, то ваша сеть работать не будет или же станет работать на скорости 1 Гбит/с.

Проверить, правильно ли вы обжали кабель, очень просто — обратите внимание на коммутатор: возле каждого его порта имеются по два индикатора. Если горят оба — все нормально. Если же горит только один из них, то этот порт работает на более низкой скорости. А если вообще не горит ни один из индикаторов, вам нужно повторить попытку — отрезать плохо обжатые коннекторы и обжать концы кабеля новыми коннекторами заново.

6.1.4. PowerLine — сеть через розетку

PowerLine — весьма интересная технология, о которой часто забывают или попросту не знают. Впервые с ней я познакомился более 10 лет назад, но тогда адаптеры

PowerLine было трудно раздобыть, а те, что удавалось найти, стоили все деньги мира. Да и сейчас пара таких адаптеров обойдется вам примерно в 6 тыс. рублей или более (а вам нужна именно пара), что дороговато для подобного рода гаджетов.

С помощью PowerLine вы можете передавать данные через электропроводку. Да, через обычную проводку 220 В. Выглядит все это так:

1. Вы покупаете пару адаптеров PowerLine.
2. Один адаптер вставляете в розетку 220 В рядом с роутером и патч-кордом подключаете его к роутеру.
3. Второй адаптер вы временно подключаете в розетку 220 В рядом и нажимаете на обеих адаптерах кнопку Pair. Это нужно для сопряжения адаптеров, которые должны работать в паре.
4. Затем извлекаете из розетки второй адаптер и подключаете его в розетку рядом с рабочим местом — например, со стационарным компьютером, телевизором, игровой приставкой и т. п.
5. Затем патч-кордом соединяете этот адаптер и устройство, которому необходима сеть. Все, вы получили проводную сеть без необходимости прокладывать Ethernet-кабель.

Эта технология — своеобразный костыль. Вы должны понимать, что полноценную сеть на несколько десятков устройств вы таким образом не построите. Но для подключения того самого устройства, которое нельзя подключить иначе, это выход.

Почему эта технология не получила популярности? Пожалуй, из-за ее цены: один адаптер стоит порядка 3 тыс. рублей и дороже, а пара соответственно свыше 6 тыс. рублей. Представим типичный вариант использования. Есть двухэтажное здание, роутер — на первом этаже, на втором этаже — детская комната, в которой есть игровая консоль без поддержки Wi-Fi. Вот ее и можно подключить подобным способом. Однако стоит ли оно того? Если можно купить самый дешевый повторитель Wi-Fi вроде TP-Link RE200, цена которого те же самые 3 тыс. рублей, установить его рядом с приставкой и подключить ее патчкордом к повторителю. Во-первых, это дешевле. Во-вторых, какой бы слабый ни был усилитель (RE200 можно ругать, любить и не любить, но свое дело он делает), сеть с ним будет работать лучше, чем без него. То есть за 3 тыс. руб. вы не только решаете поставленную задачу, но и улучшаете качество Wi-Fi на втором этаже. Кстати, *в главе 7* описан практический опыт использования именно этого Mesh-усилителя.

Таким образом, прибегать к PowerLine нужно только в том случае, если получить стабильное соединение другими способами не получается. Например, на большом расстоянии от повторителя (уже от повторителя, а не роутера) или при наличии большого количества преград на пути канала от повторителя, не позволяющих получить стабильное соединение на удаленном компьютере. Представим, что у вас в доме есть мертвая зона. Вы устанавливаете повторитель сигнала где-то посередине между роутером и мертвой зоной. Так, чтобы сам повторитель находился в зоне относительного уверенного приема сигнала Wi-Fi, — если он не получит сигнал, то и не сможет его повторить. А далее вы обнаруживаете, что в вашей мертвой зоне

сигнал Wi-Fi все равно недостаточно стабильный. Можно, конечно, подключить удаленное устройство кабелем. Но опять-таки — придется прокладывать кабель, портить ремонт и т. д. Именно тогда на помощь и приходят адаптеры PowerLine. С одной стороны, 6 тыс. руб. — дороговато. С другой, хороший усилитель сигнала вроде TP-Link RE450 стоит дороже, а RE505X — еще дороже. А если и он не поможет (а узнаете вы об этом только после покупки), то вам придется-таки прокладывать кабель... или же купить два адаптера PowerLine.

Расстояние, на которое распространяется раздача PowerLine-сети, ограничивается только вашей электропроводкой. Поэтому сеть у вас однозначно будет. Качество и скорость этой сети будут зависеть от качества проводки и удаленности от роутера (заявленная скорость передачи сети PowerLine — до 1000 Мбит/с). Причем удаленности в метрах проводки, а не «по воздуху». В принципе тот же TP-Link обещает скорость до 600 Мбит/с между двумя адаптерами. Пусть этой скорости вы с PowerLine на практике не достигнете, но 50–100 Мбит/с у вас быть должно. А 100 Мбит/с — это скорость того же FastEthernet, который еще недавно был стандартом, и новые роутеры с портами 100 Мбит/с (а не 1000 Мбит/с) до сих пор продаются. Так что это вполне хороший результат, учитывая, что вам не придется прокладывать кабель и портить ремонт. И даже если вы получите скорость на уровне 50 Мбит/с — это все же лучше, чем ничего. В любом случае не ждите от PowerLine высоких скоростей — это средство на случай, если другие способы не работают.

Вот некоторые особенности этой технологии:

- обеспечивает функционирование в пределах одного счетчика. Передать сеть в квартиру соседа таким образом не получится;
- совместима только с однофазной сетью, что не позволяет использовать эту технологию в частных домах, в которые заведено три фазы;
- нельзя подключать адаптеры через сетевые фильтры;
- включение мощных бытовых приборов вроде стиральной машины, электродуховки, электрокотла может создавать помехи и снижать скорость сигнала.
- Использовать PowerLine или нет — решать вам.

6.2. Настройка подключения по локальной сети

Подключение к Интернету по локальной сети включает и тот частный случай, когда компьютер подключен кабелем непосредственно к маршрутизатору (роутеру), — это ведь тоже локальная сеть, разве что подключение к роутеру минует другое сетевое оборудование, не установленное у вас за ненадобностью.

В большинстве случаев на домашнем маршрутизаторе, как правило, запущен DHCP-сервер, основной задачей которого является автоматическое назначение клиенту IP-адреса и других сетевых настроек. Другими словами, при наличии в сети DHCP-сервера вам вообще ничего не придется настраивать — просто подключите сетевой кабель к компьютеру.

Протокол DHCP

DHCP (Dynamic Host Configuration Protocol) — сетевой протокол, позволяющий сетевым устройствам автоматически получать IP-адрес и другие параметры, необходимые для работы в сети TCP/IP.

Итак, для подключения по локальной сети с использованием DHCP-сервера вам нужно выполнить следующие действия:

1. Подключить сетевой кабель к компьютеру (компьютер выключать для этого необязательно).
2. Дождаться уведомления о том, что соединение установлено (рис. 6.4).
3. Запустить браузер и начать пользоваться Интернетом.

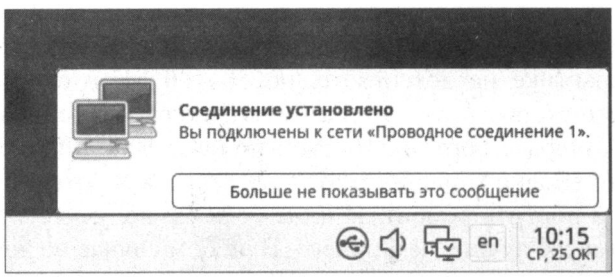

Рис. 6.4. Проводное соединение установлено

Если же DHCP-сервер в сети не работает, то вам следует установить параметры протокола IP: IP-адрес, маску сети и IP-адрес шлюза. Все эти параметры вы можете узнать у администратора вашей сети. Но, повторюсь, в 99% случаев вам это делать не придется.

Тем не менее установить параметры IP можно так:

1. Запустите **Панель управления** (рис. 6.5).
2. Перейдите в раздел **Сеть**.
3. Запустите конфигуратор **Сетевые соединения**.
4. Щелкните на соединении **Проводное соединение 1** двойным щелчком.
5. Перейдите на вкладку **Параметры IPv4** (рис. 6.6).
6. Из списка **Метод** выберите значение **Вручную**.
7. Нажмите кнопку **Добавить**.
8. Укажите IP-адрес, маску сети и шлюз, если они не были указаны там ранее.
9. Укажите серверы DNS. Если в вашей сети нет сервера DNS, можно использовать сервер DNS 8.8.8.8, но об этом позже (см. *разд. 6.3*).
10. Нажмите кнопку **Сохранить.**
11. Откройте браузер и попробуйте обратиться к любому сайту.

    Если ничего не получилось, проверьте указанные параметры — возможно, вы допустили ошибку. Найдя ошибку, исправьте ее, если все равно сеть не заработала, обратитесь к ее администратору за помощью.

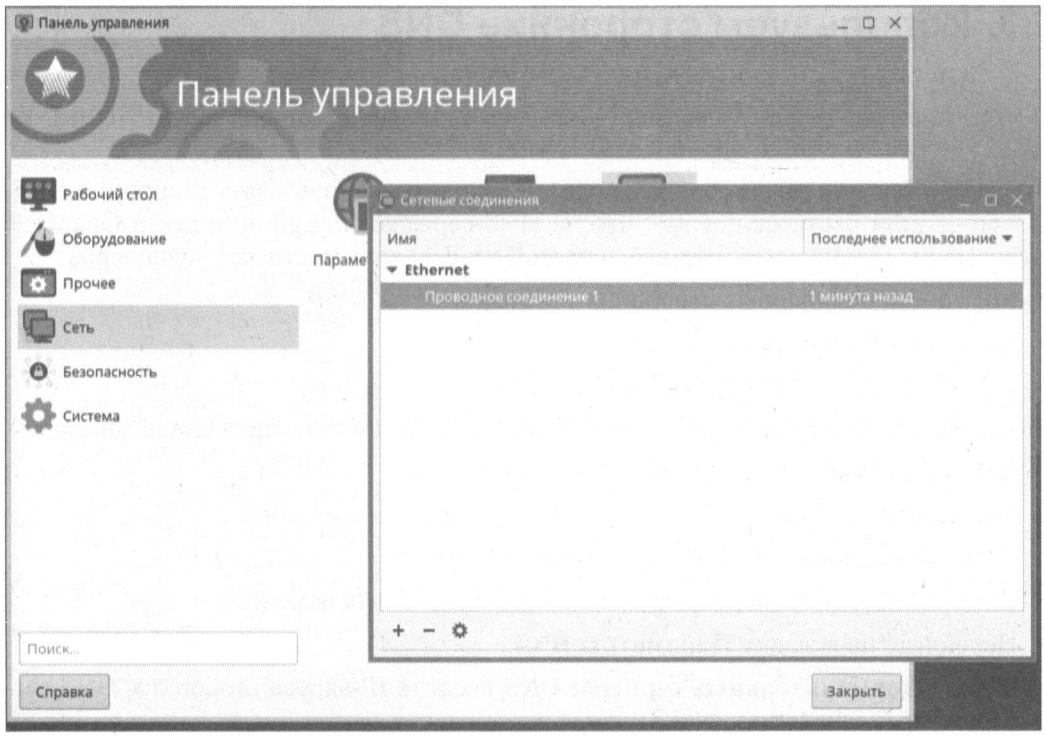

Рис. 6.5. Панель управления | Сеть | Сетевые соединения

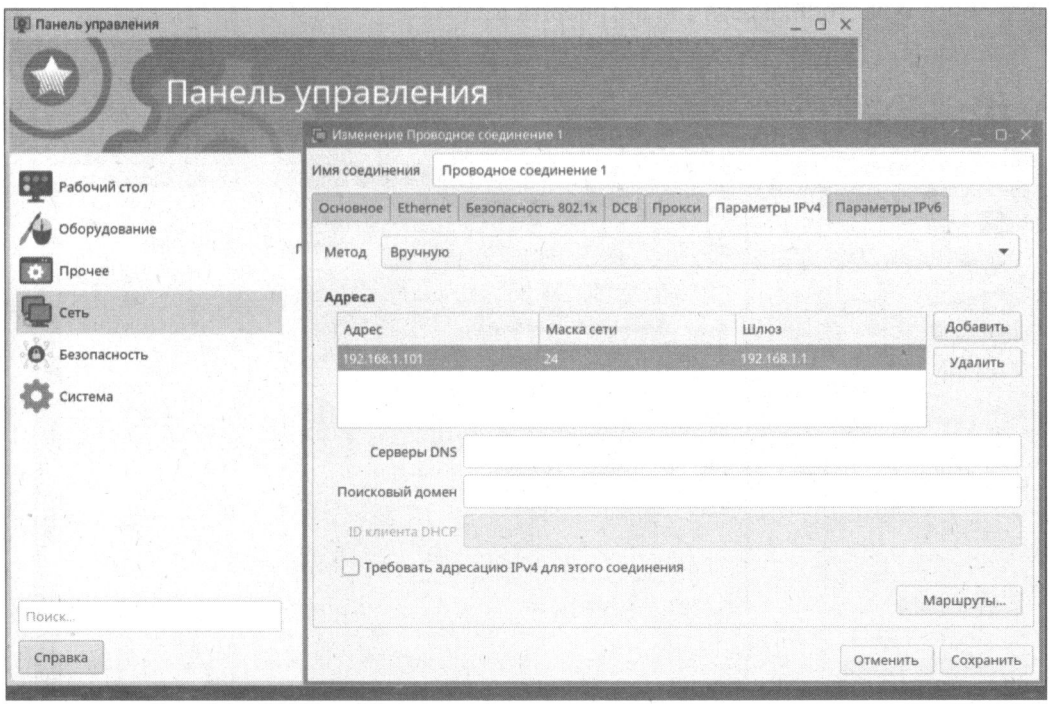

Рис. 6.6. Вкладка **Параметры IPv4**: редактирование параметров соединения

6.3. Используем сторонние DNS

DNS-сервер

DNS-сервер — это специальный компьютер в Интернете, который хранит или кеширует IP-адреса сайтов и выдает их браузеру по запросу.

Иногда по тем или иным соображениям необходимо использовать сторонний DNS-сервер — хотя бы даже потому, что на DNS-серверах Google и проекта OpenDNS информация обновляется быстрее, чем на DNS-серверах местных провайдеров.

Вот IP-адреса популярных сторонних DNS-серверов:

- 8.8.8.8 и 8.8.4.4 (Google);
- 208.67.222.222 и 208.67.220.220 (OpenDNS).

Чтобы указать IP-адреса сторонних DNS-серверов, выполните следующие действия:

1. Откройте **Панель управления** (рис. 6.7).
2. Перейдите в раздел **Сеть**.
3. Запустите конфигуратор **Сетевые соединения**.
4. Выберите соединение, параметры которого вы хотите изменить.
5. Перейдите на вкладку **Параметры IPv4**.
6. В поле **Дополнительные серверы DNS** введите IP-адреса сторонних DNS-серверов через запятую (рис. 6.7).
7. Нажмите кнопку **Сохранить**.

Подробнее о настройке сети мы поговорим в *главе 8*.

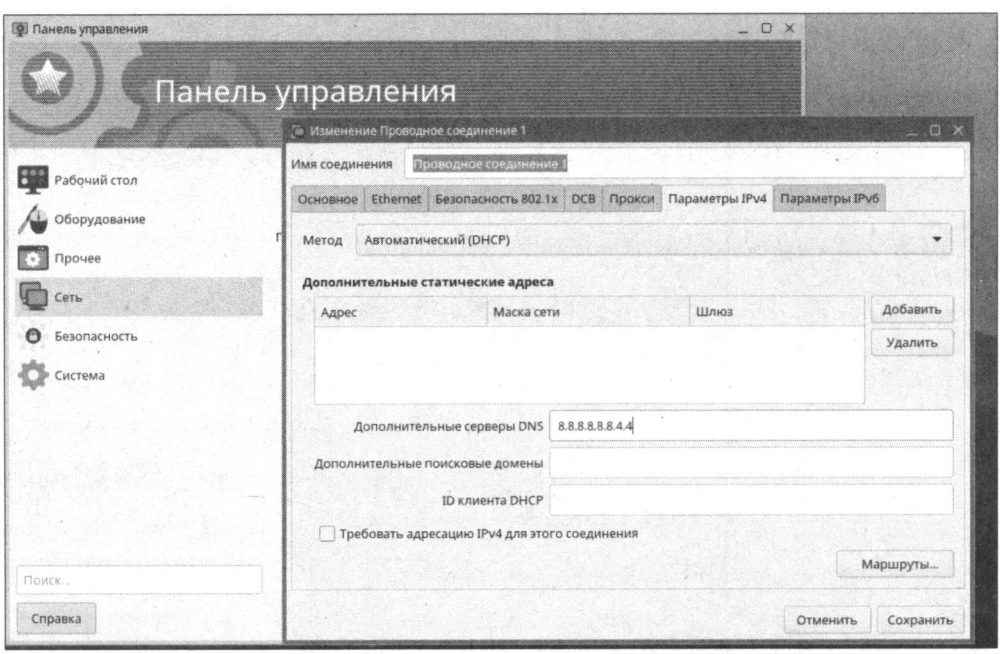

Рис. 6.7. Использование IP-адресов сторонних DNS-серверов

ASTRA LINUX®

ГЛАВА 7

Настройка беспроводного соединения с Интернетом

- Выбор роутера
- Усиление сигнала. Mesh-системы
- Некоторые советы по настройке роутера
- Подключение по Wi-Fi в Astra Linux

7.1. Выбор роутера

Настройка самого соединения Wi-Fi, как правило, не вызывает каких-либо проблем и осуществляется за пару десятков секунд: выбрать сеть и ввести пароль. Глава, содержащая описание лишь этого процесса, была бы очень скучной и малополезной. Но я попытаюсь внести в нее немного разнообразия... Посвящаю эту главу тем, кто покупал беспроводной роутер лет 5–10 назад и при этом не особо задумывался над выбором устройства.

С тех пор много что изменилось, и от правильного выбора устройства будет зависеть ваше общее впечатление от работы сети. Будете ли вы получать удовольствие от ее работы или же вам придется постоянно перезагружать роутер?

Что же изменилось за последнее время? Скажем так, лет 10 назад домашняя сеть состояла всего из нескольких устройств — например, ноутбука и/или компьютера и пары смартфонов, да и то ими не всегда для общения с сетью пользовались, — бороздить просторы Интернета удобнее было с помощью ноутбука или стационарного компьютера. Сейчас же количество устройств выросло в 5–10 раз. Если раньше у вас было 3–4 устройства, то сейчас — 15–40. Многие производители все чаще переходят на выпуск «умных устройств», и по Wi-Fi сейчас можно управлять телевизором, кондиционером, пылесосом, камерой наблюдения и даже различной мелкой бытовой техникой вроде чайников. А если вы еще надумали организовать у себя «умный дом», то количество устройств будет зашкаливать — появятся розетки, выключатели, вентиляторы, всевозможные реле и датчики. И все это использует Wi-Fi.

Возьмите и ради интереса запишите устройства, которые вы планируете подключать по Wi-Fi, а также их количество:

- ноутбуки и компьютеры;
- смартфоны и планшеты;
- смарт-телевизоры;
- ТВ-приставки;
- игровые консоли;
- «умные розетки», выключатели, датчики и прочие устройства «умного дома»;
- устройства сигнализации;
- IP-камеры и видеогистраторы;
- «умная бытовая техника», управляемая по Wi-Fi: пылесосы, чайники, мультиварки и пр.;
- «умная климатическая техника», управляемая по Wi-Fi: вентиляторы, вытяжки, кондиционеры, увлажнители и осушители воздуха, рекуператоры и т. д.

Даже в семье из 3–4 человек будет как минимум 3–4 смартфона, 1–2 компьютера, 1–2 планшета, телевизор (а может, даже несколько), ТВ-приставка, игровая консоль, сигнализация. В среднем получается около 10 устройств, и это без учета бытовой техники, различных устройств «умного дома» и камер наблюдения. Да что там говорить, прямо сейчас в моей домашней сети активны 15 устройств, и это еще не все включено (рис. 7.1).

Старые девайсы вроде TP-Link 840N уже не справляются с такой нагрузкой. И самое интересное, что они и подобные им устройства до сих пор продаются в магазинах. Но хоть убейте меня, не может справляться с современной нагрузкой роутер стоимостью до 1,5 тыс. рублей. Если у вас такой есть, не выбрасывайте — он может послужить подменным вариантом, если с основным что-то случится. Но для постоянного использования он не подходит. Тем более подобные роутеры не стоит покупать в 2024 году. Не беспокойтесь, дальше я подскажу, какие можно покупать, и причем за достойные варианты вам не придется выложить всю свою «подушку безопасности».

Итак, с количеством устройств, а также с их разнообразием мы разобрались. Теперь посмотрим, что поменялось с технической стороны. Во-первых, все современные роутеры стали гигабитными. Это означает, что порты локальной сети (LAN) на роутере поддерживают скорость в 1 Гбит/с. Вы можете возразить, мол, зачем мне скорость портов 1 Гбит/с, если скорость доступа к Интернету ограничена провайдером в 100 Мбит/с? А причин несколько:

- посмотрите, может, у вашего провайдера есть более дорогой тариф с более высокой скоростью? Недавно я узнал, что у моего провайдера есть тариф со скоростью 200 Мбит/с. Подключив смарт-телевизор кабелем к гигабитному роутеру, обнаруживаешь, что смотреть 4К-видео с такой скоростью — одно удовольствие;
- скорость работы внутренней сети также имеет значение. К роутеру можно подключить, например, сетевое хранилище с загруженными фильмами. Фильмы

можно скачать и на скорости 20 Мбит/с, причем всё равно, сколько на это будет потрачено времени — пусть загружаются хоть всю ночь, зато на следующий день вы будете наслаждаться видео в самом высоком качестве.

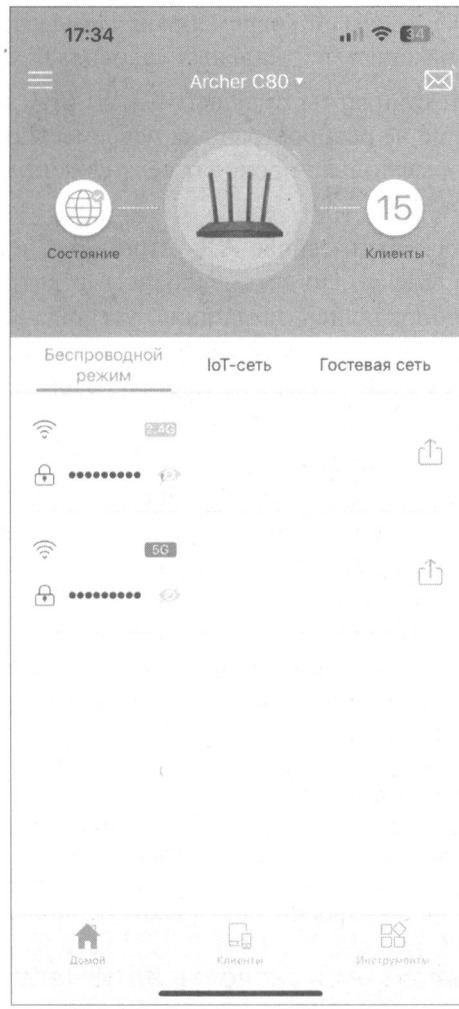

Рис. 7.1. Количество активных клиентов

Во-вторых, появились новые стандарты Wi-Fi. Собственно Wi-Fi — это коммерческое название типа связи. А самое главное — это используемый стандарт Wi-Fi. На текущий момент актуальными являются следующие стандарты:

- IEEE 802.11n (Wi-Fi4) — морально устарел, хотя, как уже отмечалось ранее, такие устройства все еще продаются. Покупать роутер, поддерживающий только этот стандарт, в 2024 году — это выброшенные на ветер деньги. Максимальная скорость на канал у него — 150 Мбит/с. Если он обладает двумя антеннами, то максимальная общая скорость составит 300 Мбит/с. Но все это, как вы понимаете, в теории;

- IEEE 802.11ac (Wi-Fi5) — все еще современный стандарт, хотя активно вытесняется более новым стандартом 802.11ax. Максимальная скорость — 433 Мбит/с на канал. Такие устройства уже можно покупать в 2024 году, особенно если вы ограничены в средствах и не готовы много заплатить за роутер;
- IEEE 802.11ax (Wi-Fi6) — самый современный стандарт, обеспечивающий скорость до 1200 Мбит/с на канал (в идеальных условиях).

Разрабатываются также и стандарты будущего — Wi-Fi6e и Wi-Fi7 (802.11be), но устройства на них пока еще не распространены повсеместно. В общем, если забыть о новых стандартах, которые только проходят сертификацию, то выбирать придется между Wi-Fi5 (AC) и Wi-Fi6 (AX).

С одной стороны, можно купить сейчас AX-устройство, и его возможностей вам хватит на несколько лет вперед. Но есть и нюансы, о которых вы должны знать. Если у вас давно не обновлялись домашние устройства Wi-Fi, то новейший AX-роутер перейдет на более медленный стандарт AC или даже на 802.11n. Другими словами, вы можете не получить выгоду от Wi-Fi6, если ваши устройства не поддерживают этот стандарт. Такова суровая реальность.

Второй нюанс связан со скоростью работы AX-роутера на частоте 2,4 ГГц. Дело в том, что современные роутеры стали двухдиапазонными, т. е. могут работать не только на частоте 2,4 ГГц, как старые 802.11n, но и на частоте 5 ГГц. Весь потенциал современных AX-роутеров раскрывается как раз на частоте 5 ГГц. Многие AX-модели работают на частоте 2,4 ГГц даже медленнее, чем AC-роутеры. Другими словами: если ваши устройства не поддерживают 5 ГГц, лучше купите AC-роутер. Он также будет двухдиапазонным, но на частоте 2,4 ГГц вы можете получить скорость беспроводной сети выше, чем у AX-роутера. Здесь нужно читать характеристики подключаемых устройств.

Wi-Fi6 и потребление энергии

Мобильные устройства вроде смартфонов потребляют меньше энергии при использовании Wi-Fi6 — примерно на 30%. Если вы строите «умный дом», использующий технологию Wi-Fi, купите AX-роутер и подбирайте датчики «умного дома», которые поддерживают Wi-Fi6, — так вы продлите срок работы их батареек.

Скорость работы роутера и скорость Интернета

Все правильно, скорость работы Интернета зависит от вашего тарифа и способа подключения к Интернету. Если у вас тариф на 50 Мбит/с, то Интернет не заработает быстрее, даже если вы купите самый дорогой AX-роутер. Интернет будет работать стабильнее, будут меньше задержки, будет лучше покрытие — все, что может сделать роутер, он сделает. Но без повышения тарифа не обойтись. Проверить текущую скорость Интернет-соединения можно на сайте **www.speedtest.net**.

2,4 ГГц — самая распространенная частота Wi-Fi. Однако из-за популярности этого диапазона он часто засорен. Ведь на этой частоте работает не только ваш роутер, но и роутер ваших соседей рядом, соседей выше и соседей ниже. Достоинство у этого диапазона тоже есть — чем ниже частота, тем лучше сигналы проходят сквозь различные препятствия, в том числе окна, двери, стены. Этот диапазон все еще востребован, особенно в частных домах, где нужно покрывать большие площади.

5 ГГц — более мощная частота. Скорость работы на этой частоте будет выше, ниже станет уровень интерференции (наложения сигналов), поскольку на этой частоте работает не так много устройств, но и покрытие станет меньше (причем иногда настолько, что сигнал может не проходить сквозь несущую стену).

Если у вас небольшая квартира или квартира-студия, вы можете попробовать вообще отключить диапазон 2,4 ГГц и работать только на частоте 5 ГГц. Большая площадь покрытия вам там не нужна, поэтому вы получите отсутствие интерференции и высокую скорость работы. Но это только в случае, если все ваши устройства поддерживают 5 ГГц.

Во всех остальных случаях лучше использовать обе частоты. Тем более что современные роутеры поддерживают технологию SmartConnect (рис. 7.2), при использовании которой ваша сеть будет доступна под одним именем для диапазонов и 2,4 ГГц, и 5 ГГц, а устройства Wi-Fi станут сами выбирать нужную им частоту. Это довольно удобно, поскольку если бы роутер не поддерживал такую технологию, то вам приходилось бы постоянно выбирать между двумя разными сетями: 2,4 и 5 ГГц. А так вам можно не задумываться над постоянной сменой сети. Представьте, что у вас двухэтажный дом, и на первом этаже работает лучше сеть 5 ГГц, но на втором ее покрытия уже нет, и вам, когда вы поднимаетесь на второй этаж, надо вручную переключаться на сеть, которая работает на частоте 2,4 ГГц,. Согласитесь, не очень удобно. Не говоря уже о том, что разговор через мессенджер вроде Telegram будет обрываться, когда покрытие 5 ГГц «закончится».

В-третьих, появились новые вспомогательные технологии. Одна из них — SmartConnect, с которой мы уже познакомились. Две следующих: квадратурная модуля-

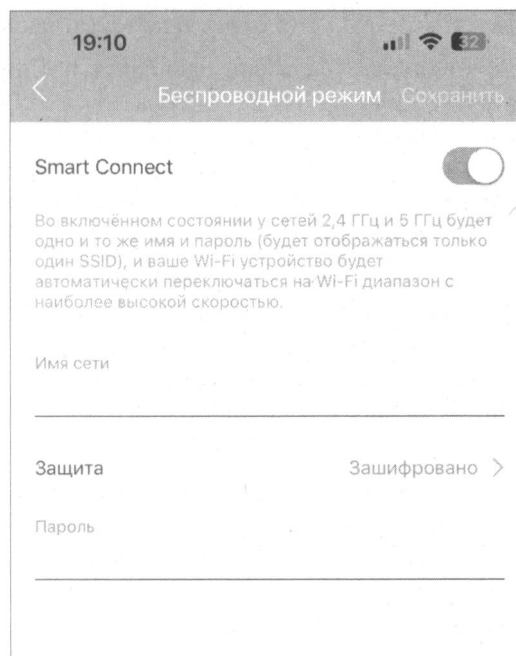

Рис. 7.2. Настройки SmartConnect

ция 1024-QAM (Quadrature Amplitude Modulation) и частотное мультиплексирование OFDMA (Orthogonal Frequency Division Multiple Access). Суть первой в том, что данные по сети передаются пакетами, и новый тип модуляции — благодаря динамическому изменению фазы и амплитуды радиоволн — вмещает больше информации в одном пакете. А технология OFDMA позволяет увеличить емкость сети — передавать данные одновременно восьми устройствам без потери скорости. Обе технологии присущи Wi-Fi6.

Также нельзя не упомянуть технологию MU-MIMO (Multi-User Multiple Input Multiple Output). Обычные маршрутизаторы, не поддерживающие MU-MIMO, работают по технологии SU-MIMO (Single User MIMO), т. е. в один момент времени данные может отправить только одно устройство. Технология MU-MIMO позволяет одновременную передачу данных несколькими устройствами. Точное количество зависит от реализации самого роутера. Роутеры от TP-Link поддерживают, как правило, три и более потока данных. Наглядную демонстрацию работы этой технологии вы можете увидеть на сайте TP-Link по адресу: **https://www.tp-link.com/ru/MU-MIMO/**. Как правило, MU-MIMO поддерживают роутеры стандарта Wi-Fi5 (AC), поскольку AX-роутеры используют OFDMA.

Технология Beamforming позволяет поддерживать высокую скорость передачи в местах, где распространение сигнала затруднено. В ее основе лежит специальная обработка сигнала, которая позволяет устройству «почувствовать», в каком направлении возникают потери сигнала, и скорректировать работу роутера соответствующим образом. Если у вас роутер поддерживает эту технологию, вы можете сами легко посмотреть, как она работает Для этого отойдите в самую дальнюю комнату и запустите тест скорости (speedtest.net). Затем запустите его второй и третий раз. Вы увидите, как, начиная со второго раза, скорость станет выше.

Еще одна новинка — технология Mesh. Она позволяет расширить зону покрытия сети, используя объединение сигнала Wi-Fi от нескольких точек доступа бесшовным способом, и пригодится при покрытии больших площадей — например, если у вас большой дом и двор или же вам нужно покрыть офисное помещение на несколько этажей. Вы получите одну сеть на все помещение, и при этом она будет обеспечивать стабильное подключение (о технологии Mesh далее расказано более подробно).

Теперь о выборе роутера. Наконец-то мы перешли к самой интересной части этой главы. Я понимаю, что все упирается в бюджет, и, если позволяют финансы, вы можете купить себе тот же TP-Link AX11000 или Keenetic Giant (я не ставлю их в один ряд, а просто сделал выборку по верху рынка) и в 99% случаев будете довольны.

А вот если вы предпочитаете тратить деньги рационально, тогда читаем дальше. Но прежде чем рассматривать конкретные модели, надо сначала обсудить необходимость наличия в них тех или иных отдельных функций. Например, некоторые роутеры оснащены 4G-модулем, что может быть полезно, если основной канал недоступен, — тогда Интернет можно будет «раздавать» по 4G. Лично для меня эта функциональность не является необходимой, поэтому я не стану выделять наличие 4G-модуля как преимущество. Если же вам она нужна, то вы, понятное дело, будете

смотреть на такие модели. Далее — наличие USB-порта. Это тоже та функциональность, которая не всем нужна. При этом наличие USB-порта еще не означает, что роутер будет делать то, что вы от него ожидаете, поскольку на некоторых моделях USB-порт используется только для обновления прошивки роутера. В то же время на ряде моделей его можно задействовать для подключения USB-модема (т. е. вы можете добавить функциональность 4G, если она вам понадобится) и внешних накопителей. Такие модели могут выступать в качестве домашнего медиасервера — к USB-порту роутера вы сможете подключить накопитель, содержимое которого будет доступно устройствам сети.

Также сугубо индивидуальным требованием является количество LAN-портов. Например, у моего роутера четыре LAN-порта: к одному подключен телевизор, ко второму — одна из IP-камер наблюдения, к третьему — сигнализация и один порт остается свободным. В принципе, если вы ошибетесь с количеством LAN-портов — ничего страшного. Можно купить копеечный коммутатор, подключить его Ethernet-кабелем к свободному LAN-порту, и вы получите минимум 4 дополнительных порта (если у коммутатора 5 портов, то один будет занят подключением к роутеру).

Итак, переходим к конкретным моделям. Если вы не хотите тратить много денег, купите TP-Link Archer C80 — это настоящий народный роутер (рис. 7.3). Да, он поддерживает только стандарт 802.11ac, т. е. Wi-Fi5, но у него 4 антенны, поддержка двух диапазонов частот, поддержка технологий Beamforming, MU-MIMO, Mesh, IPTV и SmartConnect (это вообще редкость для AC-роутеров). За те же деньги можно купить Archer A8, но он не поддерживает Beamforming и Mesh и имеет на одну антенну меньше.

TP-Link Archer C80 поддерживает стандарт AC1900, т. е. скорость работы Wi-Fi (суммарно по всем каналам) составляет 1900 Мбит/с. Это очень достойный вариант

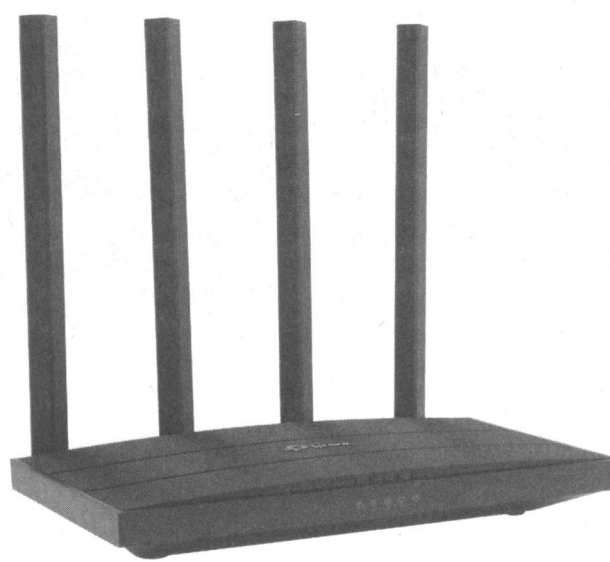

Рис. 7.3. Отличный вариант AC-роутера

для Wi-Fi5 — не все роутеры Wi-Fi6 могут работать на такой скорости. Скорость работы Wi-Fi на частоте 2,4 ГГц у него будет выше, чем у других AX-роутеров, — 600 Мбит/с против 573 Мбит/с у AX73 (на частоте 2,4 ГГц). Оставшиеся 1300 Мбит/с суммарно остаются для частоты 5 ГГц.

Также у C80 все порты, в том числе и WAN, по которому подключается Интернет, — гигабитные (1 Гбит/с). Если через год или два у вас появится тариф со скоростью более 100 Мбит/с, вам не придется менять роутер, чтобы им воспользоваться. Так что C80 — это отличный выбор за свои деньги.

Если вы хотите купить самый дешевый AX-роутер, но лишь бы он был AX, не спешите этого делать. Такую покупку я категорически не рекомендую. Сейчас вы можете и не раскрыть весь потенциал этого устройства, поскольку не все ваши устройства поддерживают Wi-Fi6 (а может так случиться, что у вас вообще пока нет ни одного устройства Wi-Fi6), а через пару лет станут доступными роутеры Wi-Fi7. Так что, если вы все же хотите что-то взять сейчас, то самый минимум из AX-вариантов — это TP-Link Archer AX1800 (рис. 7.4). Характеристики его ничем не примечательные: суммарно 1800 Мбит/с, гигабитные порты, есть Beamforming, Mesh. Это приемлемый начальный вариант роутера Wi-Fi6 — всё, что дешевле, брать не стоит.

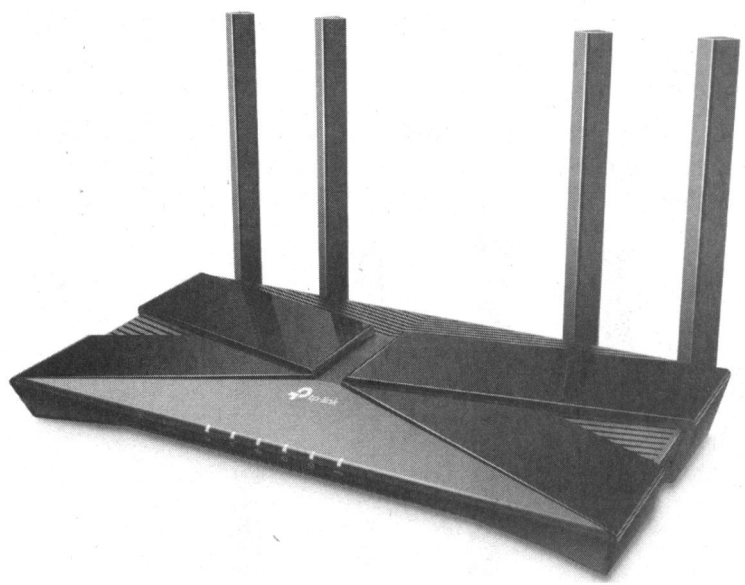

Рис. 7.4. Роутер Tp-Link Archer AX1800

Рекомендуемый AX-вариант — Archer AX73 (рис. 7.5). Большинству пользователей лучшего и не нужно: шесть антенн, скорость Wi-Fi до 4800 Мбит/с, USB-порт, позволяющий подключать 4G-модем (не во всех версиях прошивки, но это можно исправить) или внешний накопитель, встроенный VPN-клиент (вы можете защитить сразу все свои устройства, работающие через Wi-Fi).

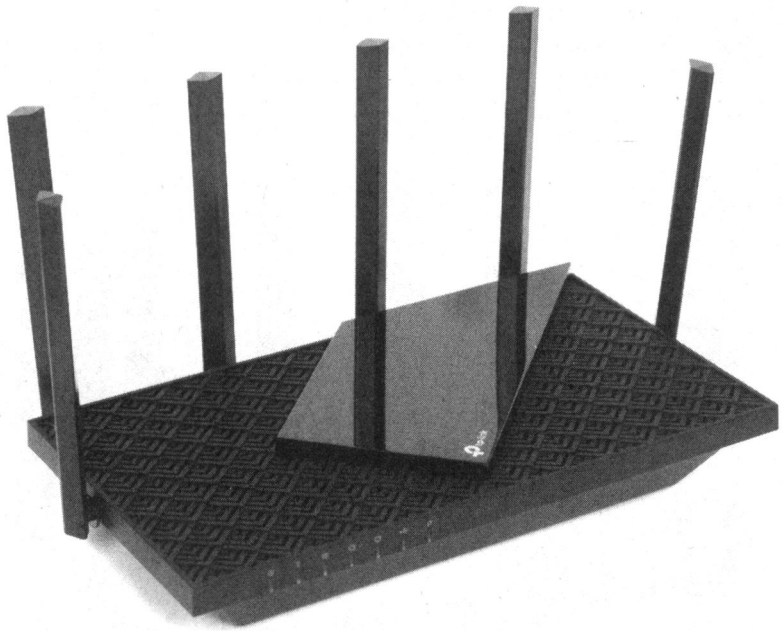

Рис. 7.5. Archer AX73

Еще о поддержке VPN

С одной стороны, в спецификациях к AX73 сказано, что у него есть поддержка VPN (а у того же C80 такой опции нет). Но маркетинг беспощаден. По сути, поддержка VPN есть у любого роутера, который поддерживает протоколы PPTP/L2TP, — даже у бюджетных моделей. Находите провайдера, предоставляющего тоннель по протоколу PPTP (считайте — это то же самое, что и VPN), и у все ваши устройства будут защищены одним роутером без необходимости устанавливать VPN на каждое из них. А то, о чем пишут маркетологи на коробках, — это поддержка реализации OpenVPN (попросту говоря, в роутер встроен OpenVPN-клиент).

Archer AX73 — очень достойный вариант, который справится с довольно сложными задачами и будет радовать вас не один год. Недостаток у него — цена, он стоит ровно в два раза дороже, чем AX1800, и в 2,5 раза дороже, чем C80. Если можете себе позволить (точнее, если не жалко денег, поскольку все-таки мы сейчас обсуждаем не покупку «Формулы-1», а всего лишь роутера) — покупайте.

Чтобы никто меня не критиковал за рекламу TP-Link, обратите внимание на Xiaomi Mi Router AX6000. По цене — примерно та же, что и у AX73 (может, чуть дороже, но не критично), зато вы получаете настоящий комбайн: скорость LAN-портов 2,5 Гбит/с, скорость Wi-Fi — 5378 Мбит/с (суммарная по двум диапазонам), 7 антенн, 512 Мбайт ОЗУ, поддержка до 248 устройств одновременно (тот же AX73 поддерживает до 200 устройств). Просто замечательный вариант от китайского производителя (рис. 7.6). Недостаток у него только один — не очень удобная панель управления, а иногда и вовсе не переведенная даже на английский язык. Благо Google Chrome умеет переводить с китайского, да и можно на AX6000 установить глобальную версию прошивки, «говорящую» на английском языке. К сожалению,

у него нет VPN-клиента, а за такие деньги хотелось бы, чтобы он был. Так что если выбирать между AX6000 и AX73, я рекомендую последний. Один только аргумент в пользу AX6000 — у вас действительно много устройств (порядка 200), и вам важна скорость Wi-Fi, которая у Xiaomi выше.

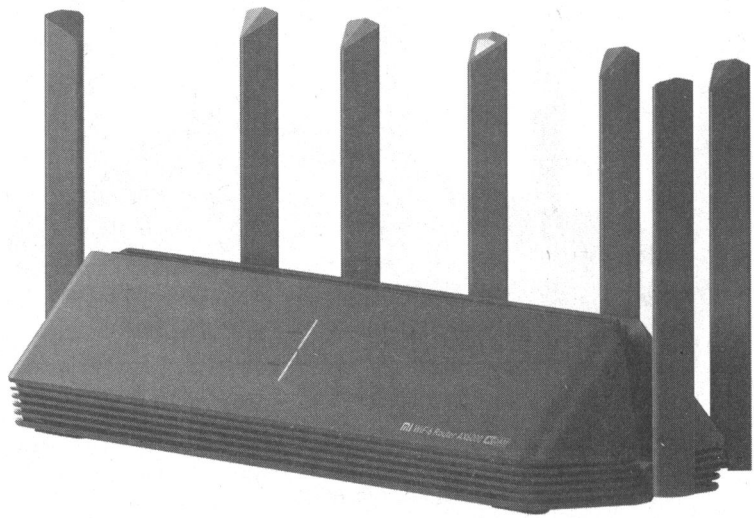

Рис. 7.6. Xiaomi Mi Router AX6000

Более дорогие маршрутизаторы вам и не нужны. Конечно, можно купить что-то вроде Archer AX11000 или Xiaomi Mi Router AX9000, но не все пользователи готовы отдать столько за роутер.

Что же касается TP-Link — они выпускают проверенные временем устройства по вполне народным ценам. Аналогичное по характеристикам устройство от Keenetic/Zyxel будет стоить дороже. Забота о пользователях тоже на высоте — для того же C80, хотя это не самая новая модель, до сих пор продолжают выпускаться обновления прошивки, которые устанавливаются автоматически, — вам не нужно ничего делать, в отличие от других брендов, где приходится самостоятельно скачать на флешку прошивку и установить ее.

К тому же мне очень нравится фирменное приложение Tether, позволяющее удаленно управлять всеми вашими роутерами от TP-Link. Вы можете добавить в свой аккаунт все свои роутеры (домашний, рабочий и т. д.) и управлять ними со смартфона.

Впрочем, и у Keenetic есть моя любимая модель, а именно — Keenetic Giant (рис. 7.7 — в отличие от остальных моделей, здесь представлен вид сзади этого замечательного роутера). Особенности Giant:

- съемные антенны и возможность работы без антенн — не всем нужна большая площадь покрытия, поэтому антенны можно просто снять, если покрытие без антенн вас устраивает;
- 8 гигабитных портов RJ-45;

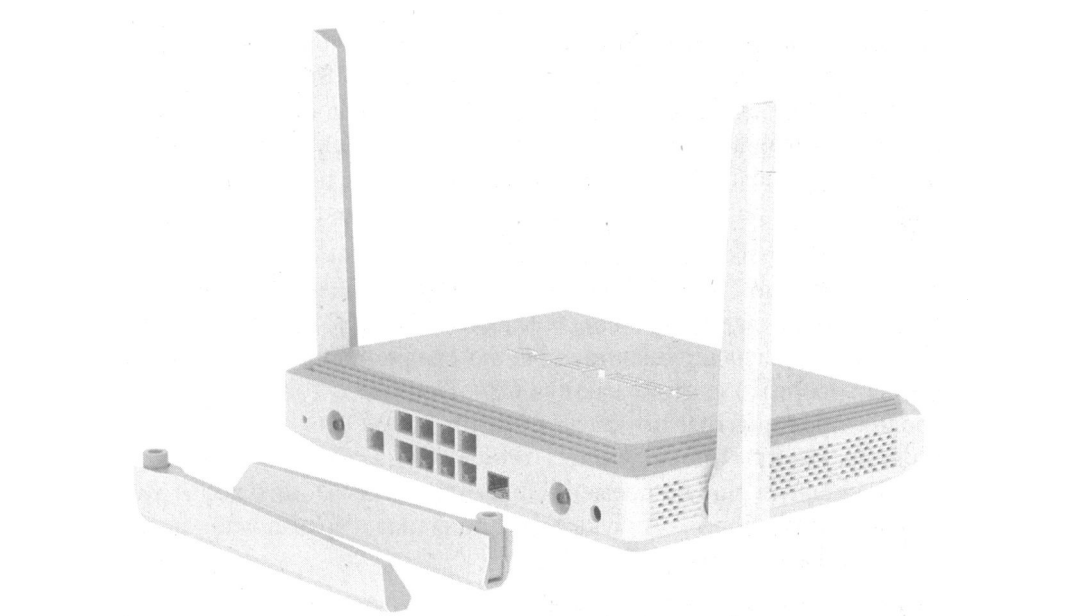

Рис. 7.7. Keenetic Giant

- один порт комбинированный — RJ-45/SFP. Он может использоваться как Ethernet, а может — как SFP, для подключения оптического кабеля;
- два USB-порта, один из них — 3.0 (чтение с диска до 90 Мбайт/с).

Разумеется, все остальные функции Keenetic: Mesh-сети, MU-MIMO, OpenVPN и прочее, что соответствуют топовым моделям от Keenetic, — присутствуют.

Нравится он мне количеством Ethernet-портов и наличием SFP-порта. Представьте следующую конфигурацию: стационарный компьютер, сигнализация, контроллер «умного дома», две IP-камеры, смарт-телевизор или ТВ-бокс, игровая приставка. Как раз восемь портов.

У него два недостатка. Первый — это роутер стандарта Wi-Fi5 (к сожалению, пока модель Wi-Fi6 с 8 портами не вышла, — тогда бы я рекомендовал ее). Но, поверьте, на сегодняшний день Wi-Fi5 более чем достаточно, особенно, когда на вход подается соединение со скоростью не более 200 Мбит/с (счастливчики гигабитных соединений тихо улыбаются).

Второй — цена. Даже с максимальными скидками, учитывая, что модель не новая и не Wi-Fi6, Keenetic Giant стоит дороже, чем тот же AX73. Если к AX73 добавить недорогой репитер (хотя бы RE300 — его последние версии получили поддержку Mesh) и не очень дорогой коммутатор на 5–8 портов, то получится конфигурация, даже превосходящая Giant. И при этом она будет стоить дешевле. Но здесь больше вопрос дизайна и личных предпочтений по части брендов. Кому нравится Keenetic, никогда не посмотрит на более дешевые Tp-Link. А если судить с точки зрения дизайна, то Giant выглядит лаконичнее и как-то более благородно, а со снятыми антеннами — вообще красавчик. Tp-Link, при всем уважении к этому бренду, в плане дизайна проигрывает, плюс в описанной только что «сборной» конфигурации у вас

появятся дополнительные «коробочки» — это коммутатор (минус, кстати, одна розетка — его нужно запитать) и USB-хаб (у Giant два USB-порта, поэтому, чтобы уравнять шансы, могу сделать предположение, что вы добавите к Keenetic дополнительный хаб). Выглядеть будет вся эта конфигурация не очень. Разве что вы спрячете все это за висящий на стене телевизор или в распределительный шкаф. Хотя сам роутер в распределительный шкаф помещать не рекомендуется.

Идем дальше. Маршрутизаторы MikroTik — тоже достойный выбор, но не для всех, поскольку это довольно специфические устройства. Линейка WRT от Linksys уже неактуальна, а другие варианты, вроде Linksys MAX-STREAM AC1750, на фоне моделей от TP-Link выглядят бледно, даже по сравнению с тем же C80 (про AX73 можно даже не говорить), а стоят как два C80. Про Cisco тоже молчу — это оборудование уровня предприятия, но когда речь идет о SOHO-сетях, на первое место выходит один аргумент — цена. А здесь он явно не в пользу Cisco. На продукцию D-Link у меня аллергия еще со времен первого роутера. Понимаю, что много воды утекло, но фантомные боли — штука мало понятная. Хотя по ценовому диапазону — это устройства, близкие к TP-Link.

Отдельного разговора заслуживает китайский производитель Mercusys. На удивление — неплохие роутеры по весьма доступной цене. Сравните аналогичные решения от того же Tp-Link, и вы поймете разницу. Так, топовый MR90X AX6000 с LAN-портами 2,5 Гбит/с обойдется дешевле 11 тыс. рублей. У него восемь антенн, Wi-Fi6, два WAN-порта (т. е. можно добавить резервное соединение). Всего четыре RJ45, один WAN, один комбинированный WAN/LAN и два LAN-порта. Очень мощный аппарат за очень скромные деньги. Почему так дешево? Да потому что у него нет никаких плюшек вроде USB-портов, поддержки Mesh-сетей и пр. Да, USB-порта нет даже в топовой MR90X. Производитель жертвует всеми этими «вкусностями» ради цены. Единственное, что у него есть (из разряда того, что есть даже на бюджетном C80) — удаленное управление с помощью мобильного приложения от производителя и MU-MIMO 4x4 (одновременно могут передавать 4 устройства). Добавьте сюда 4-ядерный процессор с частотой 1,6 ГГц, и вы поймете, что перед вами довольно серьезная железка.

На этом всё... Так что выбирайте, что вам надо. Повторюсь, роутер AX6000 за такие деньги — это очень дешево. И если нужен просто быстрый маршрутизатор — неплохой вариант. С остальными вендорами, увы, не работал — нельзя объять необъятное (хотя знаю, что есть неплохие варианты от ASUS).

Теперь подытожим. Если не жалко денег — покупайте TP-Link Archer AX73/ Keenetic Giant, и вы останетесь ими довольны. Если жалко и не принципиален Wi-Fi6 — оптимальным вариантом, как по цене, так и по характеристикам, будет Archer C80.

7.2. Усиление сигнала. Mesh-системы

В больших помещениях, скорее всего, сила сигнала в отдаленных комнатах будет ниже, чем в комнатах, находящихся ближе к роутеру. Это вполне нормальное явление. Тем не менее даже если сигнал не самый сильный, скорость соединения

может быть все еще приемлемой. В этих случаях такими потерями сигнала можно пренебречь. А вот если сила сигнала падает до такой степени, что пользоваться Интернетом в отдаленных комнатах становится некомфортно, тогда нужно задуматься об усилении сигнала.

Есть несколько способов улучшить сигнал:

- покупка нового, более мощного роутера.

  Если вы давно не меняли свой роутер, купите современную и не самую дешевую модель — в большинстве случаев, особенно если речь идет о помещении площадью до 100 кв. м, это решит вашу проблему. Например, если роутер установлен в центре первого этажа площадью 50–60 кв. м, то в самой дальней комнате второго этажа останется вполне приемлемая скорость соединения, так что вы сможете смотреть фильмы онлайн в хорошем качестве. Здесь не нужно ничего выдумывать. Более того, даже на небольшом расстоянии от здания вы получите вполне нормальный сигнал. Конечно, все зависит от интерференции и материала стен, но здесь не та площадь, чтобы слишком тратиться на покупку Mesh-системы;

- подключение еще одного роутера к уже существующему.

  Пригодится, если у вас уже есть роутер, который вы не планируете менять, и остался старый роутер, который был установлен до этого. Например, вы установили новый роутер и обнаружили, что качество сигнала все еще не такое, как бы вам того хотелось. В этом случае вы можете подключить второй роутер в режиме WDS-моста для расширения существующей сети Wi-Fi. В Интернете есть множество подробных руководств на этот счет, и вы сможете использовать одно из них для настройки такой системы. Скажу только, что работает эта система не всегда стабильно. Я пробовал подобную связку из двух роутеров TP-Link 840N — сила сигнала была достаточно мощной, но в дальних помещениях скорость передачи данных по Wi-Fi все равно оставляла желать лучшего. Конечно, второй роутер был подключен к основному по Wi-Fi. Если бы я подключил его с помощью Ethernet-кабеля, было бы лучше, но тогда бы пришлось портить стены и прокладывать кабель, а этого не хотелось. Такую систему можно использовать в качестве временного решения, особенно если имеется второй роутер. А если его нет, то для расширения сети Wi-Fi покупать именно роутер не стоит, поскольку есть специальные решения — повторители, или расширители диапазона (range extender);

- повторитель (ретранслятор) — специальное устройство, позволяющее расширить покрытие вашей сети.

  Представляет он собой небольшую коробочку с антеннами Wi-Fi, включаемую в розетку 220 В. В идеале хорошо бы подключить ее к роутеру по Ethernet — после этого вы можете разместить ее в любом месте помещения в пределах длины кабеля (а она может составлять до 100 метров). Если же подключать ее кабелем не хочется, тогда установите повторитель как можно ближе к роутеру. Не к месту возможного подключения клиента Wi-Fi, а на таком расстоянии от роутера,

чтобы между повторителем и роутером оставался высокий сигнал Wi-Fi. Многие повторители оснащаются индикаторами, позволяющими узнать силу сигнала;

- Mesh-системы — относительно новое решение, позволяющее создать бесшовную сеть Wi-Fi в большом помещении.

  Внешне Mesh-система представляет собой несколько довольно симпатичных модулей — стартовый набор, например, состоит из двух или трех таких элементов. Один из них основной — он содержит WAN-порт. Остальные модули могут, как и ретрансляторы, подключаться к основному кабелем или по Wi-Fi. По сути, это и есть ретрансляторы, но для конечных пользователей. Все уже настроено «из коробки», и всё, что вам нужно, — это расположить модули в нужных местах вашего дома. В среднем один модуль позволяет покрыть до 150 кв. м, а три модуля расширяют зону покрытия до 400 кв. м. Точные спецификации устройств различаются в зависимости от конкретной модели.

  О технологии Mesh, во всяком случае от компании Tp-Link, нужно знать следующее:

  - далеко не все роутеры от Tp-Link поддерживают Mesh. Список моделей, поддерживающих бесшовные сети, представлен по адресу **https://www.tp-link.com/us/easymesh/product-list/**. Хорошая новость состоит в том, что Tp-Link регулярно обновляет прошивки своих устройств (во всяком случае это касается роутеров среднего и высшего ценовых диапазонов) и если у вас актуальная модель роутера, но она еще не поддерживает Mesh, то может так случиться, что она получит поддержку в будущем;
  - если у вас есть два роутера с поддержкой Mesh, вы можете создать на их базе бесшовную Mesh-сеть, хотя еще год назад эта возможность отсутствовала и для расширения сети предлагалось использовать только репитер;
  - не все модели репитеров поддерживают Mesh. Убедитесь, что вы покупаете репитер с поддержкой Mesh. Если вы купите обычный репитер (или репитер не от Tp-Link), у вас будет две сети: одна основная и вторая — созданная репитером, что не очень удобно;
  - в процессе работы Mesh-роутер создает две невидимые сети SSID (их можно обнаружить только при сканировании сетей Wi-Fi специальными приложениями вроде Wi-Fi Scanner) — не волнуйтесь, так и должно быть. Когда вы подключите повторитель с поддержкой Mesh, эти сети будут использоваться как раз повторителем, и вы увидите, что у вас уже четыре сети с одним и тем же именем: две будут работать на частоте 2,4 ГГц, а две — на 5 ГГц;
  - Mesh-система вроде Deco (см. далее) несовместима с Mesh-сетью, создаваемой роутером Tp-Link. Вы можете использовать эти устройства вместе, но Deco будет работать отдельно в качестве точки доступа и подключаться к роутеру. Возможности Deco в качестве маршрутизации сильно ограничены, поэтому такая конфигурация также имеет право быть. Но у вас, по сути, будут две отдельные сети: одна — от роутера, вторая — от Deco.
  - цены на системы Deco не сбалансированы, и конфигурация на три модуля стоит почему-то дороже, чем две конфигурации по два модуля. В одной сети

может быть до 32 модулей Deco, поэтому, если у вас большой дом, можете смело покупать два набора по два модуля — так вы сможете более уверенно покрыть всю его площадь;

- все модули системы Deco взаимозаменяемые, и каждый из них может выступать как в качестве главного модуля, так и устройства-спутника.

Теперь давайте подытожим. Если у вас небольшой дом, скажем на 100–130 кв. м, достаточно купить хороший роутер — что-то вроде TP-Link AX73 (или, в крайнем случае, C80, который прекрасно справляется с такой же площадью, но у вас будет стандарт Wi-Fi5), и этого будет достаточно. Если в каком-то из направлений беспроводной сигнал окажется недостаточно сильным, всегда можно добавить повторитель. Это самый бюджетный вариант, который будет также нормально работать в домах до 200 кв. м.

При выборе повторителя желательно, но не обязательно покупать продукцию того же вендора, что и роутер. Наоборот, на всем известном сайте можно купить обладающие влагозащищенными корпусами изделия китайских инженеров, которые вы сможете установить снаружи здания. Это будет идеальное решение — вынести повторитель за пределы здания, если вдали от дома у вас находится беседка, и вы хотите, чтобы и там было нормальное покрытие Wi-Fi. Понятно, что полную влагозащиту гарантировать никто не может, но под свесом крыши или под козырьком с этим устройством ничего не случится, зато вы минуете минимум одну стену — наружную. Единственное, на что нужно здесь обращать внимание, — это на стандарт Wi-Fi, который поддерживает повторитель: если у вас роутер Wi-Fi6, а вы купили повторитель Wi-Fi5, согласитесь, получится не очень хорошо.

Хороший повторитель не может стоить дешево. Он стоит дешевле, чем хороший роутер, но может стоить дороже, чем среднестатистический роутер вроде C80, — так что не удивляйтесь, почему так дорого. Даже если сейчас у вас роутер Wi-Fi5, купите повторитель Wi-Fi6 — в скором времени роутер вы поменяете, зато повторитель менять не придется. Тем более что разница в цене между ними не очень большая. Например, если сравнивать модели RE405 (Wi-Fi5) и RE505x — разница может быть чуть более 1000 рублей, поэтому лучше взять модель с Wi-Fi6.

На рис. 7.8 показан повторитель TP-Link RE505x — оптимальный вариант, поддерживающий стандарты 802.11n, 802.11ac и 802.11ax.

Повторители подключаются к роутеру только по Wi-Fi. Если устройство подключается к роутеру по кабелю, то это уже точка доступа и немного другая функциональность. На текущий момент точки доступа (по крайней мере, у того же TP-Link) не поддерживают Mesh, поэтому Wi-Fi расширить-то получится, но у вас будут две сети: одна — от роутера, вторая — от точки доступа. Даже если вы назовете их одинаково, у вас не будет бесшовного перехода между ними, и когда сигнал с от точки доступа станет сильнее, чем от роутера, устройство отключится от сети роутера и подключится к сети точки доступа, т. е. произойдет разрыв соединения.

Расскажу о своем опыте с роутером C80 и повторителем сигнала TP-Link RE200 V5 (в версии V5 есть поддержка Mesh-сетей). Предыстория такова: на втором этаже дома периодически телевизор не хотел подключаться к Wi-Fi, и приходилось его

перезагружать (вытаскивать вилку, находящуюся за ТВ, висящем на стене, — не очень-то удобно), а если это не помогало — перезагружать роутер, что вообще неудобно. Самое интересное, что замеры скорости возле телевизора на смартфоне показывали стабильный сигнал (4 деления из 4) и скорость на уровне 80 Мбит/с (по данным сайта speedtest.net).

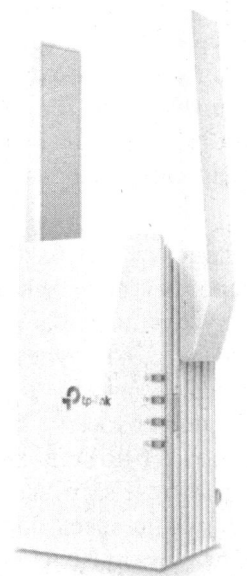

Рис. 7.8. Повторитель TP-Link RE505x

В соседней комнате, правда, ситуация была хуже, и компьютер показывал 25–50 Мбит/с. Сеть работала на 2,4 ГГц, а на втором этаже сети 5 ГГц не было из-за отсутствия уверенного сигнала. Хотя если отключить SmartConnect, можно было принудительно подключиться к сети 5 ГГц.

То есть сигнал был, но не так уж чтобы очень хороший. Я решил попробовать усилитель, который установил рядом с телевизором, а сам телевизор «посадил на шнурок» — т. е. подключил к нему патч-кордом. Все бы хорошо, но проблему это не решило. Причина была в прошивке самого телевизора, и помогло назначение статического IP-адреса для проводного соединения. Но это другая история. Что же касается Wi-Fi, то благодаря самому простому повторителю RE200 у меня в доме на двух этажах получилась бесшовная Mesh-сеть. На втором этаже устройства стали подключаться по 5 ГГц, а скорость, показываемая сайтом speedtest.net, на компьютере, находящемся в дальней комнате, составила 90 Мбит/с. Учитывая, что на «вход» подавалось 100 Мбит/с, — вполне себе хороший результат. Так что усилитель работает, и конкретно в этом случае нет необходимости в более дорогих его собратьях вроде RE450 или RE505x.

В итоге из роутера C80 и простейшего повторителя RE200 получилась Mesh-сеть, покрывшая более 100 кв. м площади, два этажа и часть двора всего за 7 тыс. рублей. Теперь посмотрите, во сколько вам обойдется подобная сеть на базе Keenetic. Можете подсчитать сами.

Если же вы не хотите ничего искать в Интернете, а у вас большое помещение, и вы хотите установить нечто, и чтобы все работало «из коробки», — обратите внимание на Mesh-системы. На рис. 7.9 показана Mesh-система Mercusys Halo H30G на три модуля.

Рис. 7.9. Mercusys Halo H30G

Эта система позволяет развеять миф о том, что Mesh-системы стоят дорого. Да, решения от того же TP-Link, да еще и на три модуля, стоят дороже. А вот эта система обойдется вам не дороже 10 тыс. рублей. Да, это Wi-Fi5. Да, скорость 1300 Мбит/с (суммарная), но есть Beamforming, MU-MIMO, IPTV, двухдиапазонный Wi-Fi (2,4 и 5 ГГЦ), порты LAN 1 Гбит/с. Есть и еще более дешевая система — H30 (без G) — то же самое, но ниже скорость Wi-Fi (1200 Мбит/с). Если вы не очень требовательный пользователь, не планируете на каждом из телевизоров смотреть видео в 8K онлайн, а вам просто нужно обеспечить стабильное покрытие Wi-Fi по всему дому и при этом не потратить кучу денег — это идеальное решение. Конечно, в линейке Mercusys есть и более производительные модели вроде Halo H80X с поддержкой Wi-Fi6 и скоростью до 3000 Мбит/с. Однако и цена там будет ровно в два раза дороже.

Линейка подобных систем от TP-Link называется Deco. Но цена вас не порадует. На рис. 7.10 изображена Mesh-система Deco S4. По своими характеристикам она похожа на Mercusys Halo H30 (даже не H30G) — скорость Wi-Fi5 на уровне 1200 Мбит/с, а цена при этом в полтора раза выше. Единственное, чем она лучше, — так это дизайном. Но дизайн штука такая — через 3 дня после покупки уже перестаешь обращать на него внимание.

Чуть более производительная Deco S7 обойдется даже дороже, чем Halo H80X, но при этом вы получите Wi-Fi5 AC1900, т. е. 1900 Мбит/с — на уровне роутера C80 за 4200 рублей. Нормальная же по характеристикам Mesh-система (Wi-Fi6, AX3000) Deco PX50 стоит почти 70 тыс. рублей. Не думаю, что это рациональное

Рис. 7.10. TP-Link Deco S4

решение. Можно уже тогда купить AX73 и пару повторителей RE505X. Бюджет обойдется почти в половину дешевле, зато вы получите полноценный роутер.

Преимущества Mesh-систем — дизайн и простота развертывания. Все-таки не каждый современный роутер можно назвать красавцем, а повторители — те еще уродцы. А красивые бочонки или кубики легко впишутся в любой дизайн и не будут привлекать к себе много внимания. Да и простота развертывания на высоте — вам нужно только включить систему и настроить основной модуль (грубо говоря, указать название сети Wi-Fi и придумать пароль). Но на этом преимущества заканчиваются и начинаются недостатки. Недостатков тоже два. Первый — это цена. Здесь каждый решает сам, может ли он позволить себе Mesh-систему или же будет создавать зоопарк из роутеров и повторителей. Второй — админка основного модуля довольно урезанная, и вы не найдете привычных функций, которые были у вас на обычном роутере. Например, на устройствах от TP-Link есть функция блокирования по MAC-адресу — вы задаете MAC-адреса устройств, которым разрешено подключаться к вашей сети. Также есть функция родительского контроля, когда вы определяете расписание доступа к Интернету для каждого детского устройства (также есть функция блокировки нежелательных сайтов и т. п.). Всего этого на Mesh-системе у вас не будет. Количество LAN-портов также ограничено. Как правило, у вас будет один WAN-порт и один-два LAN-порта. На основном модуле могут быть два LAN-порта (как вы уже догадались, в идеале они будут заняты двумя другими модулями Mesh-системы). На дополнительных — будет по одному LAN-порту, который может использоваться как для подключения каких-либо устройств, так и для последовательного соединения модулей. Конечно, можно купить коммутатор рублей за 500–700 и решить этот вопрос. Ну а как же дизайн? А никак: возле вашей красивой Mesh-системы, за дизайн которой вы переплатили половину, будет стоять угловатый «кирпич» за 700 рублей. Это примерно как в гараж рядом с самым современным мерседесом поставить что-то вроде «буханки».

7.3. Некоторые советы по настройке роутера

Здесь я не стану описывать весь процесс от начала и до конца, поскольку он будет зависеть от производителя маршрутизатора и даже от его модели. Нет смысла рассматривать настройку на примере моего роутера, если вы, скорее всего, купите себе другой. Однако есть несколько советов:

- Обязательно включите гостевую сеть.

  Когда к вам придут гости, вы сможете предоставить им данные для гостевого подключения. Гостевая сеть не дает доступа к вашим локальным ресурсам (т. е., подключившись к вашей сети, никто не сможет просмотреть ваши фото на медиасервере, который вы для своего же удобства настроили без пароля), а только предоставляет доступ к Интернету. К тому же вам не придется сообщать гостям пароль от вашей сети.

- Включать ли SmartConnect?

  Как уже было отмечено, эта функция позволяет устройствам самостоятельно выбирать диапазон подключения к Wi-Fi: 2,4 или 5 ГГц. Если ее выключить, то у вас будут две разные сети: одна — на 2,4 ГГц, вторая — на 5 ГГц. Попробуйте сначала включить SmartConnect и посмотрите, к каким диапазонам подключаются ваши устройства. Мой ноутбук сначала подключался по 5 ГГц, потом стал подключаться почему-то по 2,4 ГГц, хотя в конфигурации ничего не менялось (как и расположение роутера/ноутбука). Что случилось — непонятно. В конечном итоге я выключил эту функцию и теперь сам выбираю, к какой сети мне подключаться.

- Если есть возможность подключать кабелем отдельные устройства — подключайте.

  Если, например, ваш роутер находится рядом с телевизором, подключите телевизор к роутеру кабелем. На качестве картинки это не отразится, но так вы снизите нагрузку на всю сеть и улучшите качество картинки (чем выше сила сигнала, тем выше скорость, тем более качественный контент можно смотреть) на телевизоре, который находится в удаленной от роутера комнате.

7.4. Подключение по Wi-Fi в Astra Linux

В большинстве случаев ваш компьютер будет подключаться к Интернету по Wi-Fi. Беспроводные сети гораздо проще разворачивать, поэтому они приобрели огромную популярность, и кабельное подключение в последнее время используется только в крупных компаниях, да и то при условии, что сеть там прокладывалась давно.

Беспроводная сеть в Astra Linux настраивается так же просто, как и проводное соединение:

1. Включите ваш беспроводной адаптер. Нет смысла продолжать настройку, если ваш адаптер Wi-Fi выключен. На ноутбуках есть, как правило, специальная ком-

бинация клавиш, включающая беспроводной адаптер, или же физический переключатель на корпусе ноутбука. В случае со стационарным компьютером вам нужно физически подключить в беспроводной адаптер USB-порт.

2. Подождите некоторое время, особенно если вы впервые подключили адаптер Wi-Fi к компьютеру.
3. Щелкните на значке Network Manager в области уведомлений — вы увидите список доступных сетей. На рис. 7.11 показано, что доступно две сети, но беспроводное соединение пока не установлено.
4. Выберите нужную вам сеть. Если значок сети украшает замок, значит, для подключения к сети нужно будет ввести пароль (рис. 7.12).
5. В случае успешного подключения вы увидите соответствующее уведомление в области уведомлений. Можете еще раз щелкнуть на значке Network Manager, чтобы убедиться, что соединение действительно установлено (рис. 7.13).

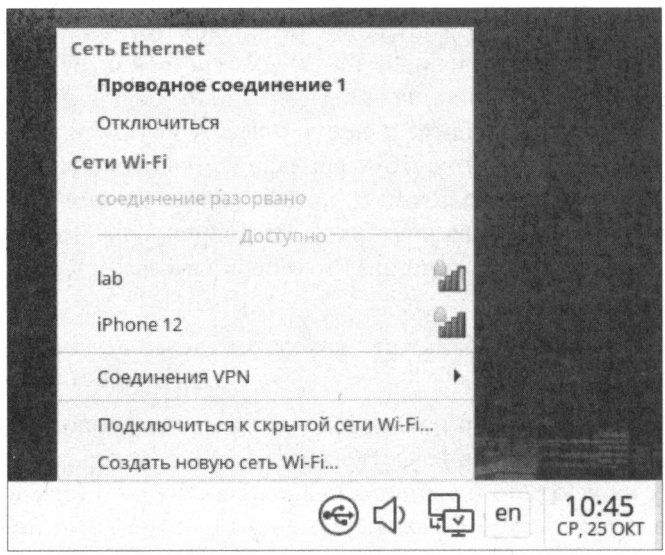

Рис. 7.11. Список доступных сетей

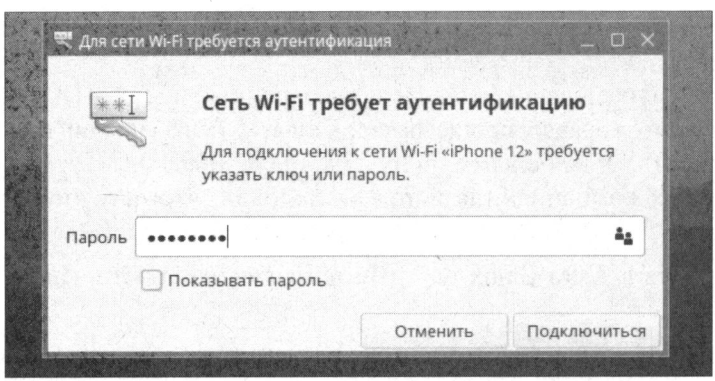

Рис. 7.12. Ввод пароля

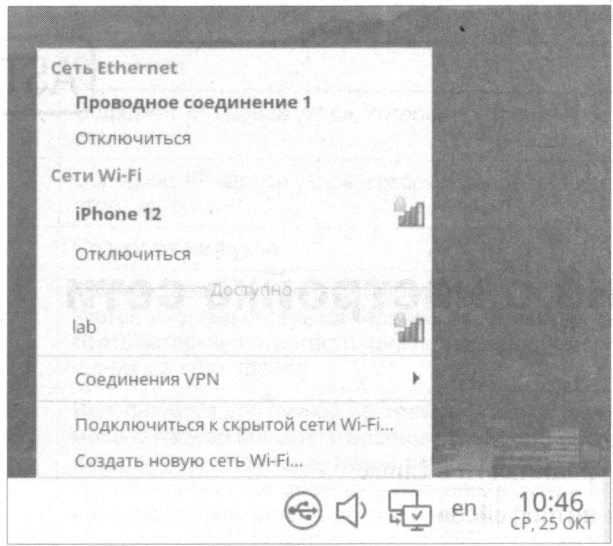

Рис. 7.13. Соединение установлено

Скрытые сети

Некоторые сети не транслируют свой SSID, поэтому вы не увидите их в списке сетей. Такие сети называются *скрытыми*. Для подключения к скрытой сети нужно выбрать команду **Подключиться к скрытой сети Wi-Fi** (см. рис. 7.12). Далее вам нужно будет ввести SSID (вы должны его знать), а потом — пароль сети.

Наверное, большинству пользователей интересно только одно: где хранится пароль от сети Wi-Fi? Перейдите в каталог /etc/NetworkManager/system-connections. В нем находятся файлы конфигурации для каждого из ваших соединений. Имя файла соответствует имени соединения, а в случае с Wi-Fi — имени сети. Откройте файл соединения и найдите параметр `psk` — это и будет пароль от Wi-Fi в незашифрованном виде.

ASTRA LINUX®

ГЛАВА 8

Подробно о настройке сети

- Файлы конфигурации сети в Linux
- Имена сетевых интерфейсов
- Статические маршруты
- Включение IPv4-переадресации

8.1. Файлы конфигурации сети в Linux

Графические конфигураторы — это просто замечательно, но ведь эта книга посвящена Linux, а в Linux принято все настраивать вручную. Вы не станете настоящим Linux-пользователем, если хотя бы не будете знать, в каких файлах находятся настройки, сделанные графическим конфигуратором (табл. 8.1).

Таблица 8.1. *Общие файлы конфигурации сети в Linux*

Файл	Описание
/etc/aliases	База данных почтовых псевдонимов. Формат этого файла очень прост: *псевдоним пользователь*
/etc/aliases.db	Системой на самом деле используется не файл /etc/aliases, а файл /etc/aliases.db, который создается программой newaliases по содержимому файла /etc/aliases. Поэтому после редактирования этого файла не забудьте выполнить от имени root команду *newaliases*
/etc/hosts.conf	Содержит параметры разрешения доменных имен. Например, директива `order hosts,bind` означает, что сначала поиск IP-адреса по доменному имени будет произведен в файле /etc/hosts, а затем лишь будет произведено обращение к DNS-серверу, заданному в файле /etc/resolv.conf. Директива `multi on` означает, что одному доменному имени могут соответствовать несколько IP-адресов
/etc/hosts	В этом файле можно прописать IP-адреса и имена узлов локальной сети, но обычно здесь указывается только IP-адрес узла localhost (127.0.0.1), потому что сейчас даже в небольшой локальной сети устанавливается собственный DNS-сервер

Таблица 8.1 *(окончание)*

Файл	Описание
/etc/hosts.allow	Содержит IP-адреса узлов, которым разрешен доступ к сервисам этого узла
/etc/hosts.deny	Содержит IP-адреса узлов, которым запрещен доступ к сервисам этого узла
/etc/hostname	Содержит имя узла
/etc/motd	Файл задает сообщение дня (Message of the day). Этот файл используется многими сетевыми сервисами (например, серверами FTP и SSH), которые при регистрации пользователя могут выводить сообщение из этого файла
/etc/network/interfaces	Используется для ручной настройки сетевых интерфейсов (не с помощью NetworkManager). Вообще-то принято настраивать сетевые интерфейсы с помощью NetworkManager, но некоторые администраторы предпочитают отключать NetworkManager и настраивать сетевые интерфейсы вручную — по старинке
/etc/resolv.conf	Задает IP-адреса серверов DNS. Формат файла прост: `nameserver IP-адрес` Всего можно указать четыре DNS-сервера. При использовании NetworkManager этот файл не используется, и настройки DNS хранятся файле соединения в каталоге /etc/NetworkManager/system-connections
/etc/services	База данных сервисов, задающая соответствие символьного имени сервиса (например, pop3) и номера порта (110/tcp, tcp — это наименование протокола)
/etc/NetworkManager/system-connections/	В дистрибутивах, использующих NetworkManager, в этом каталоге хранятся настройки соединений — в отдельных файлах по одному для каждого соединения. При этом название файла соответствует названию соединения, введенному при настройке

Наверное, большинству пользователей интересно только одно: где хранится пароль от сети Wi-Fi? В *главе 7* об этом уже упоминалось, но повтрение — мать учения... Перейдите в каталог /etc/NetworkManager/system-connections. В нем хранятся файлы конфигурации для каждого из ваших соединений. Имя файла соответствует имени соединения, а в случае с Wi-Fi — имени сети. Откройте файл соединения и найдите параметр `psk` — это и будет пароль от Wi-Fi в незашифрованном виде.

8.2. Имена сетевых интерфейсов

Все течет, все меняется. В мире компьютеров обычно все меняется в лучшую, более простую и понятную сторону. Взять хотя бы настройку сети с помощью конфигураторов. Раньше у каждого дистрибутива был свой конфигуратор сети, и не один — это и понятно: свои файлы конфигурации, сервисы настройки сети и т. д.

Сейчас же благодаря тому, что все (или практически все) современные дистрибутивы перешли на единый сервис настройки сети NetworkManager (и Astra Linux — не

исключение), конфигуратор сети у всех стал один — nm-connection-editor, и выглядит он примерно одинаково во всех дистрибутивах. Это как универсальное зарядное USB-устройство, которым можно зарядить любой современный смартфон любого производителя.

Классическое название интерфейса Ethernet — eth*X*, где *X* — это число от 0. Первому Ethernet-интерфейсу соответствует имя eth0, второму — eth1 и т. д. Вряд ли у вас будет много сетевых интерфейсов. Обычно даже на серверах, устанавливаемых в стойку, всего два интерфейса: eth0 и eth1.

Но у некоторых дистрибутивов имена интерфейсов могут отличаться от классических — например, вместо eth0 может использоваться имя ens33 или enp3so. Такие имена появились в связи с переходом современных дистрибутивов на udev (систему управления устройствами для новых версий ядра Linux).

Впрочем, определенная система в назначении новых имен также имеется, только она не столь очевидная. Сетевые адаптеры, встроенные в материнскую плату (`ID_NET_NAME_ONBOARD`), носят теперь название типа eno1. Если же сетевая карта подключена к PCI Express (`ID_NET_NAME_SLOT`), то она будет нести название типа ens33. Имена устройств, содержащие физическое/географическое расположение коннектора (`ID_NET_NAME_PATH`), будут выглядеть как enp2s0. А самые сложные и «страшные» имена у сетевых адаптеров, определяемых через MAC-адрес, — enx66e7b1es34dd.

В Astra Linux таких имен вы не найдете — за это отдельное спасибо разработчикам, которые не поленились добавить параметр ядра по умолчанию net.ifnames = 0, который отключает такие странные имена сетевых интерфейсов. На рис. 8.1 приведен вывод команды `sudo ifconfig`. Здесь мы видим три сетевых интерфейса: lo, eth0 и wlan0.

Интерфейс lo — это интерфейс обратной петли, который соответствует локальному компьютеру. Обычно ему назначается IP-адрес 127.0.0.1, и при обращении к этому IP-адресу вы обращаетесь к локальному компьютеру. Вообще-то, этому интерфейсу соответствует любой IP-адрес из подсети 127.*, поэтому 127.0.2.2 — это тоже будет локальный компьютер.

Интерфейс eth0 — это первая сетевая плата Ethernet, как уже было отмечено ранее. А интерфейс wlan0 — это беспроводной сетевой адаптер Wi-Fi. Обратите внимание: этому интерфейсу не назначен IP-адрес — потому что мы еще не подключались к сети Wi-Fi.

Кроме упомянутых имен, вы можете столкнуться со следующими именами сетевых интерфейсов:

- pppX — соединение по протоколу PPP/PPPoE (PPP over Ethernet)/PPTP;
- plip — сетевой интерфейс PLIP (Parallel Line IP). Очень древний интерфейс, но вдруг вы с ним столкнетесь...
- ax — сетевой интерфейс радио AX.25;
- tr — интерфейс Token Ring.

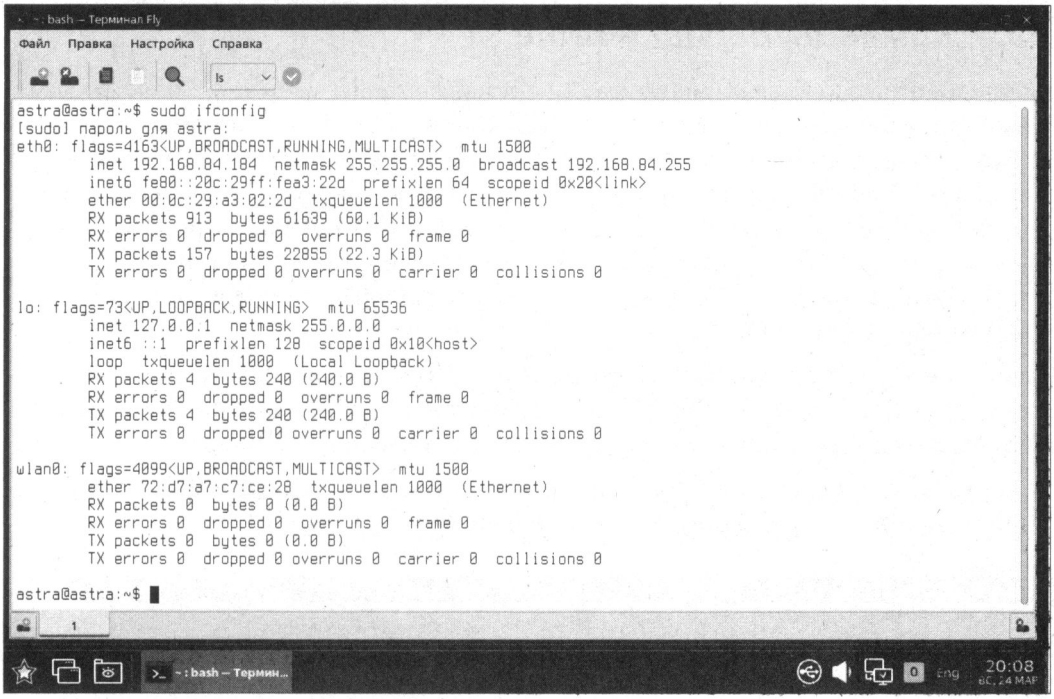

Рис. 8.1. Вывод команды `sudo ifconfig`

8.3. Статические маршруты

В TCP/IP-сетях информация о маршрутах имеет вид *правил* — например, чтобы добраться до сети А, нужно отправить пакеты через компьютер Д. Ничего удивительного и необычного — примерно так же выглядит и информация о маршрутах на дороге: чтобы доехать до города А, нужно проехать через город Д. Кроме набора маршрутов, есть также и стандартный маршрут — по нему отправляют пакеты, предназначенные для отправки в сеть, маршрут к которой явно не указан. Компьютер, на который отправляются такие пакеты, называется *шлюзом по умолчанию* (default gateway). Получив пакет, шлюз решает, что с ним сделать: или отправить дальше, если ему известен маршрут в сеть получателя пакета, или же уничтожить пакет, как будто бы его никогда и не было. В общем, что сделать с пакетом — это личное дело шлюза по умолчанию, все зависит от его набора правил маршрутизации. А наше дело маленькое — отправить пакет на шлюз по умолчанию.

Данные о маршрутах хранятся в таблице маршрутизации ядра Linux. Каждая запись этой таблицы содержит несколько параметров: адрес сети назначения, сетевую маску и т. д. Если пакет не удалось отправить ни по одному маршруту (в том числе и по стандартному), отправителю пакета передается ICMP-сообщение «сеть недоступна» (network unreachable).

8.3.1. Команды *netstat* и *route*

Для просмотра таблицы маршрутизации служат команды `netstat -r` и `netstat -rn`. Можно также по старинке воспользоваться командой `route` без параметров. Разница между командами `netstat -r` и `netstat -rn` заключается в том, что параметр `-rn` запрещает поиск доменных имен в DNS, поэтому все адреса будут представлены в числовом виде (подобно команде `route` без параметров). А вот разница между выводом `netstat` и `route` заключается в представлении маршрута по умолчанию (`netstat` выводит адрес 0.0.0.0, а `route` — метку default) и в названии полей самой таблицы маршрутизации.

Какой командой пользоваться — дело вкуса. Раньше я использовал `route` и для просмотра, и для редактирования таблицы маршрутизации. Теперь для просмотра таблицы я отдаю команду `netstat -rn`, а для ее изменения — команду `route`.

Посмотрите на рис. 8.2. На нем показан вывод команд `netstat` и `route`. Также видно, что `route` невозможно запустить без привилегий root.

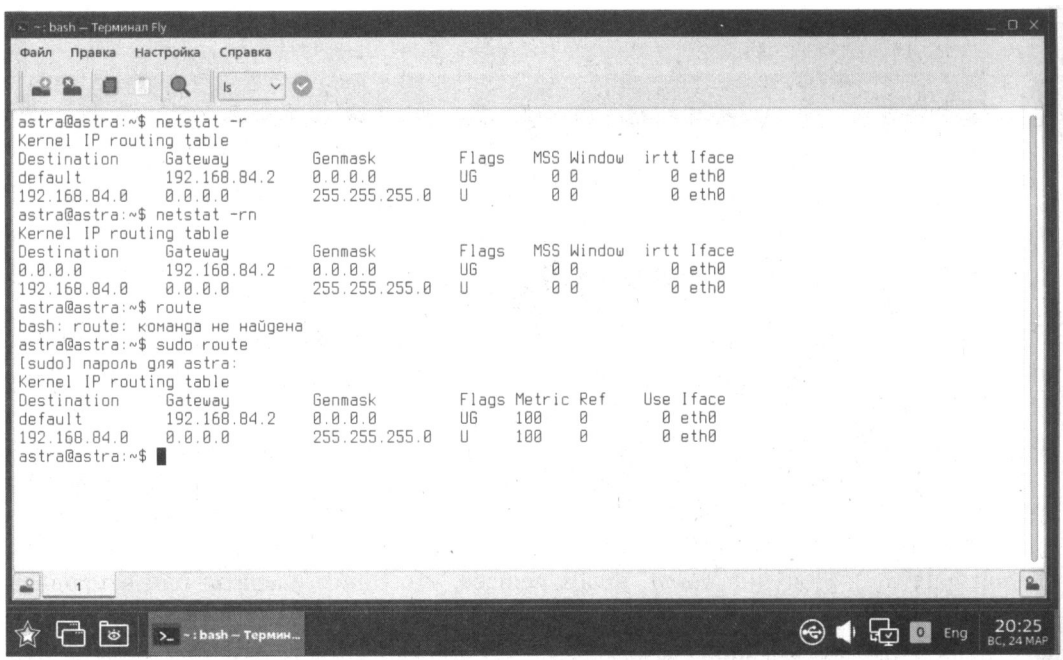

Рис. 8.2. Команды `netstat` и `route`

Поля таблицы маршрутизации объясняются в табл. 8.2.

Таблица 8.2. *Поля таблицы маршрутизации*

Поле	Описание
Destination	Адрес сети назначения
Gateway	Шлюз по умолчанию

Таблица 8.2 *(окончание)*

Поле	Описание
Genmask	Маска сети назначения
Flags	Поле **Flags** содержит флаги маршрута: • U — маршрут активен; • H — маршрут относится не к сети, а к хосту; • G — эта машина является шлюзом, поэтому при обращении к ней нужно заменить MAC-адрес машины получателя на MAC-адрес шлюза (если MAC-адрес получателя почему-то известен); • D — динамический маршрут, установлен демоном маршрутизации; • M — маршрут, модифицированный демоном маршрутизации; • C — запись кеширована; • ! — запрещенный маршрут
Metric	Метрика маршрута, т. е. расстояние к цели в хопах (переходах). Один хоп (переход) означает один маршрутизатор
Ref	Количество ссылок на маршрут. Не учитывается ядром Linux, но в других операционных системах — например, в FreeBSD, вы можете столкнуться с этим полем
Use	Содержит количество пакетов, прошедших по этому маршруту
Iface	Используемый интерфейс
MSS	Максимальный размер сегмента (Maximum Segment Size) для TCP-соединений по этому маршруту
Window	Размер окна по умолчанию для TCP-соединений по этому маршруту
Irtt	Протокол TCP гарантирует надежную доставку данных между компьютерами. Такая гарантия обеспечивается повторной отправкой пакетов, если они были потеряны. При этом ведется счетчик времени: сколько нужно ждать, пока пакет дойдет до назначения и придет подтверждение о получении пакета. Если время вышло, а подтверждение-таки не было получено, то пакет отправляется еще раз. Это время и называется round-trip time (время «путешествия туда-обратно»). Параметр **Irtt** — это начальное время rtt. В большинстве случаев подходит значение по умолчанию, но для некоторых медленных сетей, — например, для сетей пакетного радио, значение по умолчанию слишком короткое, что вызывает ненужные повторы. Параметр **Irtt** можно увеличить командой `route`. По умолчанию его значение 0

Добавить маршрут в таблицу маршрутизации можно статически (с помощью команды `route`), динамически или комбинированно (так, статические маршруты добавляются при запуске системы, а динамические — по мере работы системы). Статические маршруты добавляются, как правило, командой `route`, запущенной из сценария инициализации системы. Например, следующая команда задает шлюз по умолчанию для интерфейса eth0:

```
# route add default gw 192.168.181.2 eth0
```

Неприятно, но после перезагрузки системы добавленная нами запись исчезнет из таблицы маршрутизации. Подробнее об этом — в следующем разделе.

8.3.2. Изменение таблицы маршрутизации ядра

Маршрутизация осуществляется на сетевом уровне модели OSI. Когда маршрутизатор получает пакет, предназначенный для другого узла, IP-адрес получателя пакета сравнивается с записями в таблице маршрутизации. Если есть хотя бы частичное совпадение с каким-то маршрутом из таблицы, пакет отправляется по IP-адресу шлюза, связанного с этим маршрутом.

Если совпадений не найдено (т. е. вообще нет маршрута, по которому можно было бы отправить пакет), пакет отправляется на шлюз по умолчанию, если таковой задан в таблице маршрутизации. Как уже отмечалось ранее, если шлюза по умолчанию нет, отправителю пакета посылается ICMP-сообщение «сеть недоступна» (network unreachable).

Команда `route` за один вызов может добавить или удалить только один маршрут. Другими словами, вы не можете сразу добавить или удалить несколько маршрутов. Формат вызова `route` следующий:

```
# route [операция] [тип] адресат gw шлюз [метрика] [dev интерфейс]
```

- параметр *операция* может принимать два значения: `add` (добавить маршрут) и `del` (удалить маршрут);
- параметр *тип* необязательный — он задает тип маршрута: `-net` (маршрут к сети), `-host` (маршрут к узлу) или `default` (маршрут по умолчанию);
- параметр *адресат* содержит адрес сети (если задается маршрут к сети), адрес узла (при добавлении маршрута к сети) или вообще не указывается (если задается маршрут по умолчанию);
- параметр *шлюз* задает IP-адрес (или доменное имя) шлюза;
- последние два параметра: *метрика* и *dev* необязательны:
  - параметр *метрика* задает максимальное число переходов (через маршрутизаторы) на пути к адресату (в Linux, в отличие от других ОС, этот параметр необязательный);
  - параметр `dev` имеет смысл указывать, если в системе установлено несколько сетевых интерфейсов и требуется указать, через какой именно сетевой интерфейс нужно отправить пакеты по заданному маршруту.

Команда удаления маршрута выглядит так:

```
# route del адрес
```

В других UNIX-подобных системах имеется параметр `-f`, удаляющий все маршруты (`route -f`), но в Linux такого параметра нет. Следовательно, для очистки всей таблицы маршрутизации вам придется ввести серию команд `route del`.

Изменять таблицу маршрутизации нужно, только зарегистрировавшись на компьютере локально. При удаленной регистрации (например, по SSH) легко удалить ошибочно маршрут, по которому вы вошли в систему. О последствиях такого действия, думаю, можно не говорить.

Примеры использования команды `route`:

```
route add -net 192.76.16.0 netmask 255.255.255.0 dev eth0
```

Добавляет маршрут к сети `192.76.16.0` (сеть класса C, о чем свидетельствует сетевая маска, заданная параметром `netmask`) через устройство `eth0`. Шлюз не указан, просто все пакеты, адресованные сети `192.76.16.0`, будут отправлены на интерфейс `eth0`.

```
route add -net 192.16.16.0 netmask 255.255.255.0 gw 192.76.16.1
```

Добавляет маршрут к сети `192.16.16.0` через маршрутизатор `192.76.16.1`. Сетевой интерфейс указывать не обязательно, но можно и указать при особом желании.

```
route add default gw gate1
```

Добавляет маршрут по умолчанию. Все пакеты будут отправлены компьютеру с именем `gate1`. Обратите внимание: мы указываем доменное имя узла вместо IP-адреса.

```
route add -net 10.1.0.0 netmask 255.0.0.0 reject
```

Добавляет запрещающий маршрут. Отправка пакетов по этому маршруту (в сеть `10.1.0.0`) запрещена.

8.3.3. Сохранение статических маршрутов

Итак, мы добавили необходимые маршруты, пропинговали удаленные узлы — все работает. Теперь нужно сохранить установленные маршруты, чтобы они были доступны при следующей загрузке системы.

Перейдите в каталог /etc/network/if-up.d и создайте файл routes:

```
sudo mcedit /etc/network/if-up.d/routes
```

В листинге 8.1 приведен пример этого сценария. В его начале проводится проверка интерфейса — если она не нужна, тогда просто закомментируйте эту строку. После нее идет список команд `route`, которые вы вводили для настройки вашей таблицы маршрутизации.

Листинг 8.1. Файл /etc/network/if-up.d/routes

```
#!/bin/bash
# Проверка имени интерфейса
[ "$IFACE" != "eth0" ] || exit 0
# Дополнительные маршруты
route add -net 192.168.32.0/24 gw 192.168.1.1 dev eth0
```

Теперь нужно сделать этот скрипт исполняемым:

```
sudo chmod +x /etc/network/if-up.d/routes
```

8.4. Включение IPv4-переадресации

Основное предназначение шлюза (маршрутизатора) — это пересылка (forwarding) пакетов. Чтобы включить пересылку пакетов протокола IPv4 (IPv4 forwarding), нужно записать значение 1 в файл /proc/sys/net/ipv4/ip_forward:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Но включить пересылку мало — нужно еще сохранить это значение, иначе при перезагрузке будет восстановлено значение по умолчанию (0). Для этого следует в файл /etc/sysctl.conf добавить строку:

```
net.ipv4.ip_forward=0
```

ASTRA LINUX®

ГЛАВА 9

Настройка брандмауэра

- Что такое брандмауэр?
- Базовая настройка
- Создание правил для сервисов
- Разрешаем IP-адреса
- Запрещаем IP-адреса и службы
- Графическая оболочка Astra Linux

9.1. Что такое брандмауэр?

Брандмауэр (он же firewall, бастион, межсетевой экран) — это системное приложение, предназначенное для защиты внутренней сети (или даже одного компьютера, напрямую подключенного к Интернету) от вторжения извне. С помощью брандмауэра вы можете контролировать доступ пользователей Интернета к узлам вашей внутренней сети, а также управлять доступом локальных пользователей к ресурсам Интернета — например, вы можете запретить им посещать определенные узлы с целью экономии трафика.

Прежде чем перейти к настройке межсетевого экрана, определимся с терминологией и, в частности, с понятием *шлюз*. Шлюзом называется компьютер, предоставляющий компьютерам локальной сети доступ к Интернету. Шлюз выполняет как бы маршрутизацию пакетов. Но не нужно путать шлюз с обычным маршрутизатором. Маршрутизатор осуществляет простую пересылку пакетов, поэтому его можно использовать для соединения сетей одного типа — например, локальной и локальной, глобальной и глобальной. А шлюз служит для соединения сетей разных типов — например, локальной и глобальной, как в нашем случае. Конечно, сейчас можно встретить маршрутизаторы с функцией шлюза, но это уже, скорее, аппаратные шлюзы, чем простые маршрутизаторы. Тем не менее часто термины «маршрутизатор» и «шлюз» употребляются как синонимы, хотя это не совсем так.

Сложность в соединении сетей разных типов заключается в различной адресации. Как мы знаем, в локальной сети обычно используются локальные адреса, которые

недопустимы в Интернете, — например: 192.169.*.* (сеть класса C), 10.*.*.* (сеть класса A) и 172.16.*.*–172.31.*.* (класс B). Поэтому шлюз должен выполнить *преобразование сетевого адреса* (NAT, Network Address Translation). Суть такого преобразования в следующем. Предположим, у нас есть шлюз и локальная сеть с адресами 192.169.*.*. Реальный IP-адрес (который можно использовать в Интернете) есть только у шлюза, пусть это 193.254.219.1. У всех остальных компьютеров — локальные адреса, поэтому при всем своем желании они не могут обратиться к интернет-узлам.

Основная задача брандмауэра — это фильтрация пакетов, которые проходят через сетевой интерфейс. При поступлении пакета брандмауэр анализирует его и затем принимает решение: принять пакет (ACCEPT) или избавиться от него (DROP). Брандмауэр может выполнять и более сложные действия, но часто при его настройке ограничиваются именно этими двумя.

Прежде чем брандмауэр примет решение относительно пакета, пакет должен пройти по цепочке правил. Каждое правило состоит из условия и действия (цели). Если пакет соответствует условию правила, то выполняется указанное в правиле действие. Если пакет не соответствует условию правила, он передается следующему правилу. Если же пакет не соответствует ни одному из правил цепочки, выполняется действие по умолчанию.

Когда вы имеете дело с домашним компьютером, возможно, вам и вовсе не придется настраивать брандмауэр. В домашней сети, как правило, брандмауэр установлен на домашнем роутере, и его настройка выполняется посредством встроенной панели управления. Роутер обеспечивает NAT, а также защиту внутренней сети от подключений извне. Правила его просты (если не задано иначе) — разрешаются все исходящие соединения и блокируются все входящие.

> ***Панель управления домашнего роутера***
>
> О том, как войти в панель управления домашнего роутера, написано на нижней крышке устройства, — там вы найдете адрес, логин и пароль по умолчанию.

Именно поэтому, если вы подключаетесь к Интернету через домашний роутер, вам не нужно настраивать брандмауэр. Однако могут быть и исключения. Например, вам не нужен Wi-Fi, а ваш домашний компьютер подключается к оборудованию провайдера напрямую, минуя роутер. Например, по Ethernet, или же у вас PON-терминал (оптическое соединение), и компьютер подключается напрямую к нему. В этом случае вы не знаете, как настроен компьютер провайдера, и, скорее всего (что логично), он не блокирует никакие входящие соединения (т. к. это не входит в функции/обязанности провайдера). Следовательно, любой желающий сможет подключиться к вашему компьютеру. А это не всегда хорошо, поэтому нужно настроить правила брандмауэра и активировать его, чтобы быть в безопасности.

А вот в сети предприятия, как правило, имеется шлюз, предоставляющий всем остальным компьютерам доступ к Интернету, как было описано ранее. Аналогичный шлюз есть и в сети интернет-провайдера — он предоставляет доступ абонентам провайдера к Интернету.

Вторая функция брандмауэра (кроме NAT) — это защита компьютера от несанкционированного доступа, чтобы никто извне не смог обратиться к компьютерам вашей сети.

В Linux в качестве брандмауэра используются два решения. Первое — iptables, второе — UFW. Брандмауэр iptables — достаточно сложное решение и больше подходит для сетей предприятий. Для защиты домашнего компьютера вполне сгодится и UFW (Uncomplicated Firewall, или «простой брандмауэр»), который и будет рассмотрен далее.

> ***Брандмауэр UFW***
>
> Брандмауэр UFW установлен в Linux по умолчанию, поэтому вам не нужно предпринимать какие-либо действия по его установке

9.2. Базовая настройка

Сначала посмотрим состояние брандмауэра:

```
sudo ufw status verbose
```

По умолчанию фильтр пакетов выключен, поэтому вы получите сообщение (рис. 9.1):

```
Status: inactive
```

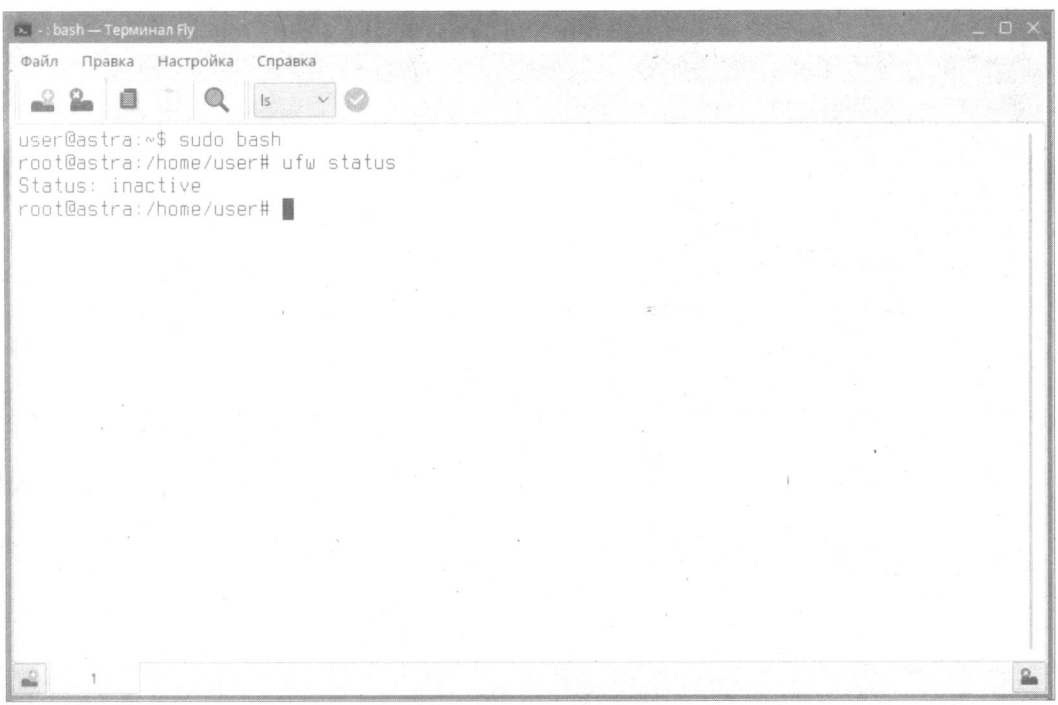

Рис. 9.1. Состояние UFW по умолчанию — выключен

Будучи включен, брандмауэр по умолчанию запрещает все входящие соединения и разрешает все исходящие. Такая политика идеальна с точки зрения безопасности компьютера (дальше вы поймете почему) — ведь если кто-то (и вы в том числе) захочет к нему подключиться, у него это не получится. В то же время приложения на сервере смогут создавать исходящие соединения.

Рассмотрим две команды:

```
ufw default deny incoming
ufw default allow outgoing
```

Эти два правила как раз и задают политику по умолчанию: запрещаются все входящие соединения и разрешаются все исходящие.

Итак, все входящие соединения запрещены. Чтобы до сервера можно было «достучаться» по определенному порту, его нужно сначала открыть. UFW хорош тем, что вам даже не нужно помнить номер порта, — надо знать только название сервиса. Например, вот так можно разрешить подключение по SSH:

```
ufw allow ssh
```

При этом UFW сам создаст правило для порта 22 — именно этот порт используется для SSH. Брандмауэр знает порты и имена всех распространенных служб (HTTP, SSH, FTP, SFTP и пр.).

Однако если вы перенастроили SSH на нестандартный порт из соображений той же безопасности, следует явно указать номер порта:

```
ufw allow 3333
```

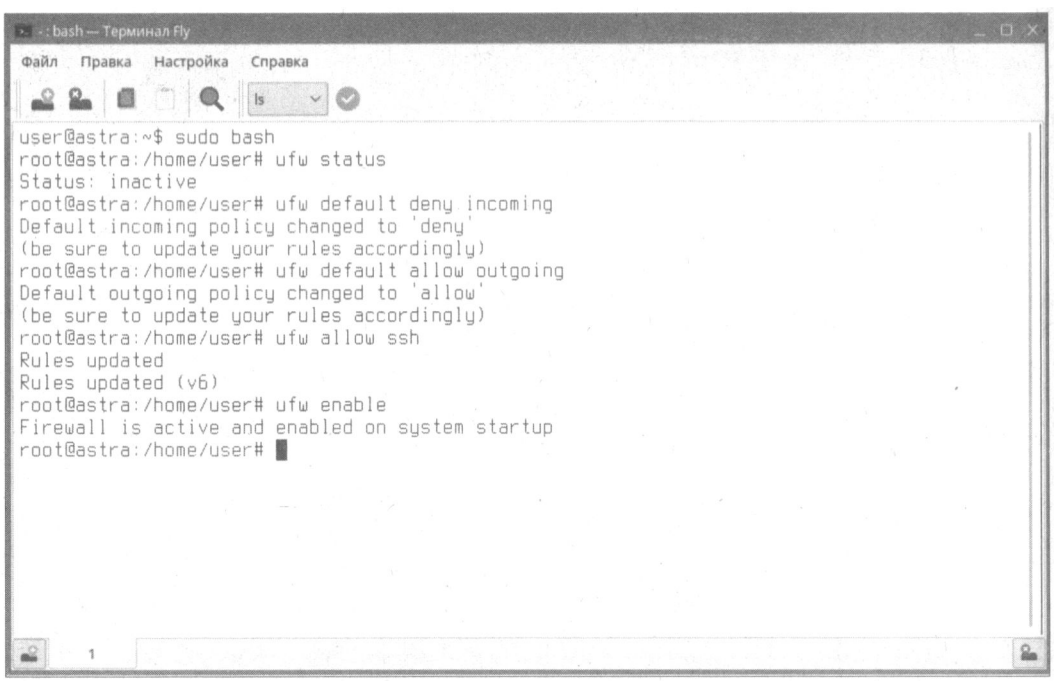

Рис. 9.2. Базовая настройка UFW

После разрешения SSH (это главное, чтобы сейчас брандмауэр нам не разорвал соединение) можно включить UFW командой:

```
ufw enable
```

Посмотрите на рис. 9.2. Разберемся, что здесь произошло. Сначала мы разрешили SSH, на что получили ответ, что правила обновлены:

```
Rules updated
```

Затем включили брандмауэр и получили сообщение, что он активен и будет запускаться при загрузке системы.

На этом базовая настройка выполнена: SSH успешно работает, и мы можем приступить к дальнейшей настройке фильтра пакетов.

9.3. Создание правил для сервисов

Теперь надо разрешить работу других приложений. Как правило, следует разрешить службы HTTP (веб-сервер), FTP (если этот сервис вам нужен) и постараться не забыть о HTTPS (что очень важно в последнее время):

```
ufw allow http
ufw allow https
ufw allow ftp
```

Сделать то же самое можно было бы и по номерам портов:

```
ufw allow 80
ufw allow 443
ufw allow 21
```

При желании можно разрешить целый диапазон портов, указав при этом транспортный протокол (UDP или TCP):

```
sudo ufw allow 2000:2200/tcp
sudo ufw allow 4000:4400/udp
```

9.4. Разрешаем IP-адреса

Брандмауэр UFW позволяет разрешить определенному IP-адресу доступ ко всем портам сервера, например:

```
ufw allow from 46.229.220.16
```

Если нужно разрешить конкретному IP-адресу доступ только к определенному порту, то делается это так:

```
ufw allow from 46.229.220.16 to any port 22
```

Здесь мы разрешаем не все подключения по SSH, а только с IP-адреса 46.229.220.16.

Разрешить доступ целого диапазона IP-адресов (например, когда у админа динамический IP) можно так:

```
ufw allow from 123.45.67.89/24 to any port 22
```

9.5. Запрещаем IP-адреса и службы

Запретить доступ с определенного IP-адреса можно аналогично:

```
ufw deny from 123.45.67.89
```

При желании можно запретить все подключения к определенной службе:

```
ufw deny ftp
```

9.6. Удаление/сброс правил

Сбросить все правила можно командой:

```
ufw reset
```

Но убедитесь, что на момент ввода этой команды вы отключили брандмауэр, иначе вы потеряете доступ по SSH.

Удалить конкретное правило можно по номеру. Сначала введите следующую команду, чтобы узнать номер правила:

```
ufw status numbered
```

А уже затем удаляйте его:

```
sudo ufw delete <номер правила>.
```

Как видите, в настройке брандмауэра нет ничего сложного.

9.7. Графическая оболочка Astra Linux

В Astra Linux для управления правилами брандмауэра вы также можете использовать графическую оболочку, запустить которую можно с помощью панели управления. Откройте панель управления, перейдите в раздел **Сеть** и запустите утилиту **Настройка межсетевого экрана**.

Использовать оболочку просто. Рассмотрим пример включения брандмауэра и добавления правила, разрешающего входящее SSH-соединение:

1. Переведите переключатель **Статус** в положение включено (рис. 9.3). По умолчанию все исходящие подключения разрешены, а все входящие — запрещены.
2. Перейдите на вкладку **Правила**. По умолчанию список правил пуст (рис. 9.4). Нажмите кнопку **+**.
3. В открывшемся окне перейдите на вкладку **Обычные**, введите название правила, выберите политику **Разрешить**, направление **В**, протокол **TCP**, порт **22**. Нажмите кнопку **Добавить** (рис. 9.5).
4. В список правил будут добавлены созданные вами правила — это одно и то же правило, но для протоколов IPv4 и IPv6 (рис. 9.6).

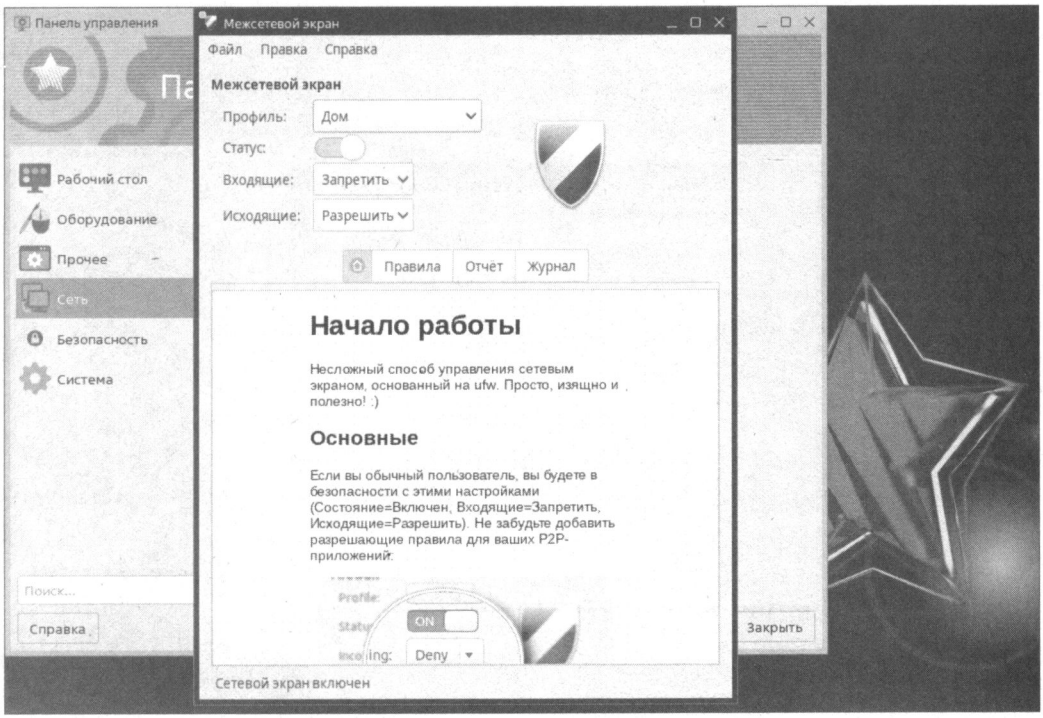

Рис. 9.3. Включение брандмауэра

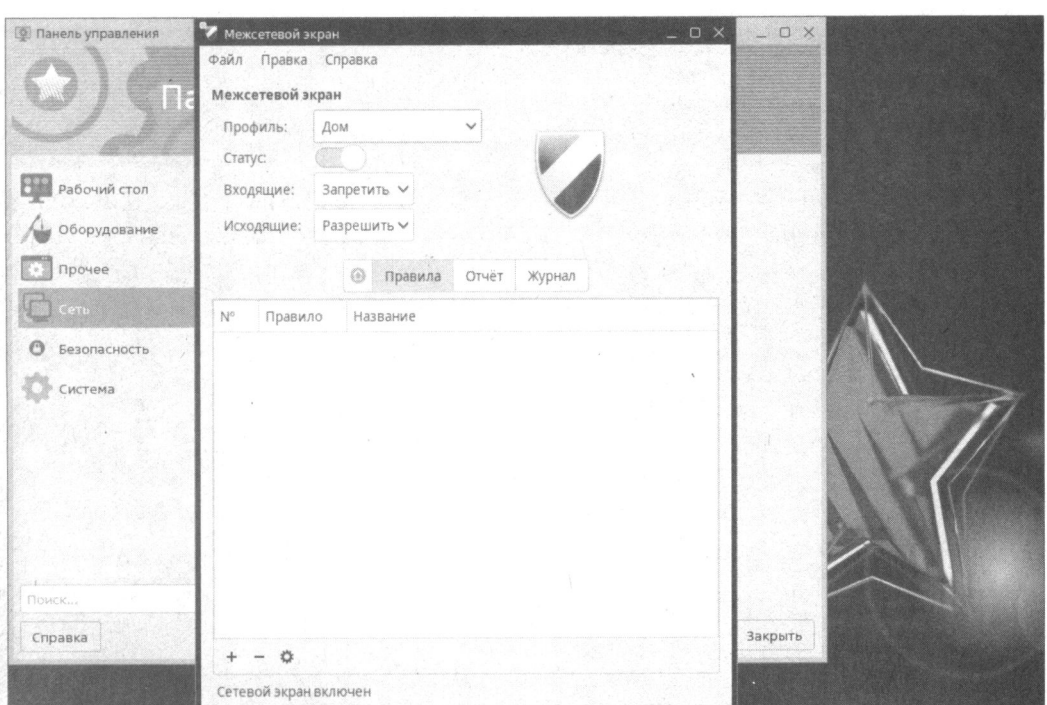

Рис. 9.4. Пустой список правил

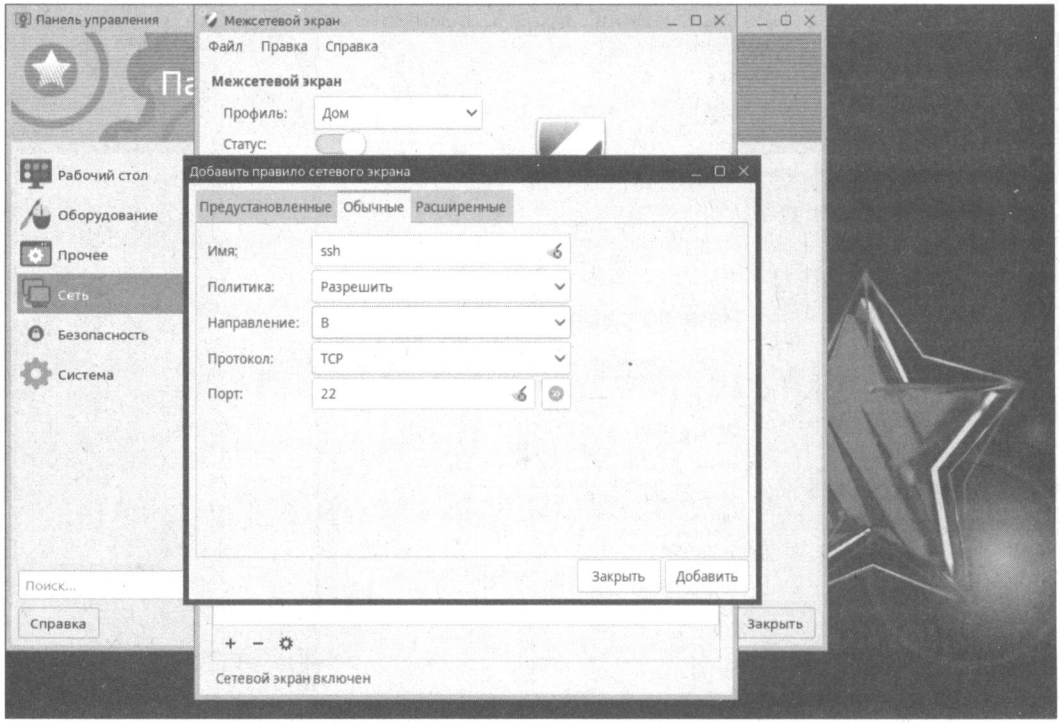

Рис. 9.5. Добавление правила

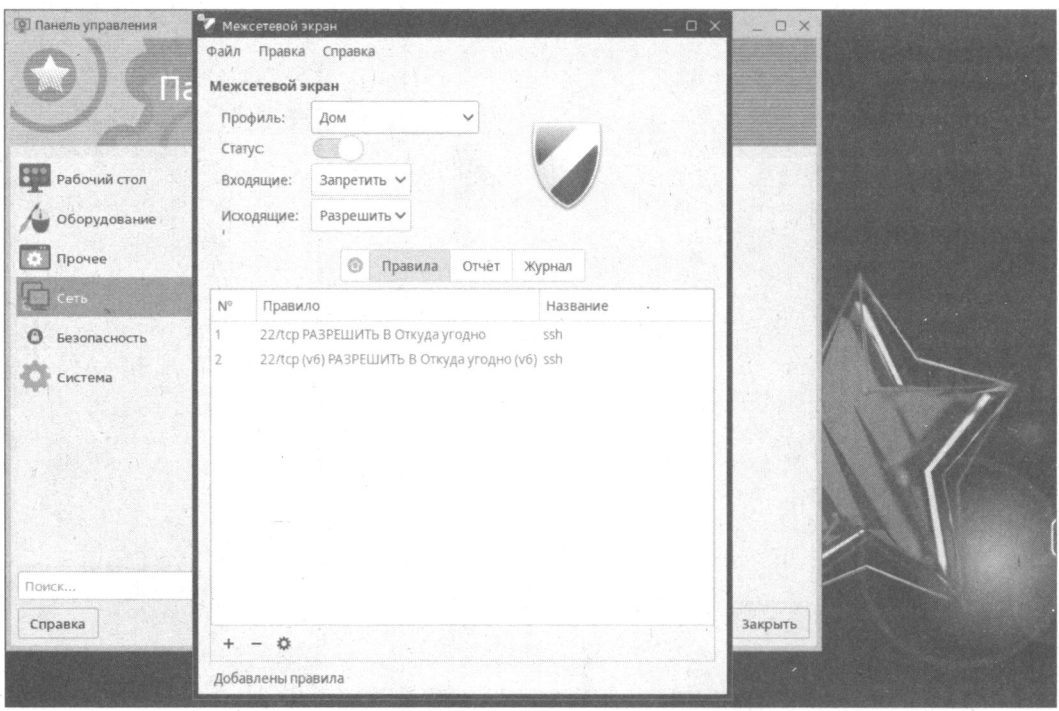

Рис. 9.6. Правила добавлены

ASTRA LINUX®

ГЛАВА 10

Диагностика сети

- Команда `ping`
- Команда `traceroute`
- Команда `ifconfig`
- Команда `ip`
- Команда `nmcli`
- Проблемы с разрешением доменных имен
- Сканирование портов

10.1. Команда *ping*

Неизменным инструментом-помощником администратора является всем известная команда `ping`. О ней не знает разве что самый начинающий пользователь. Первое, что нужно делать при диагностике сети, — это пропинговать либо интересующий узел, к которому у вас не получается подключиться, либо всем известный узел, который должен быть доступен всегда. На рис. 10.1 показан результат «пинга» Google. Мы видим, что ответ получен, и значит, с нашим соединением все в порядке.

Принцип действия команды прост. Она отправляет на заданный узел ICMP-датаграмму[1] `ECHO_REQUEST` и ожидает от него команду `ECHO_RESPONSE`. Единственное существенное отличие Linux-версии от версии для Windows в том, что «пинг» здесь идет бесконечный (а не только четыре попытки), и для завершения работы программы нужно нажать комбинацию клавиш <Ctrl>+<C> либо сразу указать с помощью параметра `-c` количество пакетов `ECHO_REQUEST`.

Для каждого пакета выводится значение time — это время, за которое ICMP-запрос дошел до адресата и вернулся обратно. В нашем случае в среднем время путешест-

[1] ICMP (Internet Control Message Protocol) — сетевой протокол, входящий в стек протоколов TCP/IP. В основном ICMP используется для диагностирования сети и помогает определить, достигают ли адресата пакеты, передаваемые по сети.

вия каждого пакета составило 23 мс, и это вполне нормально. А вот если время будет выше 200 мс (0,2 с) — это уже не очень хорошо. Запомните, что высокое значение time (высокая задержка) — это не показатель медленного Интернета, а следствие нестабильной работы сети. У мобильных соединений 3G/LTE показатель ping, как правило, выше, чем для кабельных. А для беспроводной сети высокий ping может означать плохое качество сигнала, которое можно повысить разными способами:

- перенести компьютер ближе к роутеру хотя бы на время, чтобы убедиться, что высокая задержка является следствием плохого сигнала. Улучшится сигнал — уменьшится задержка. Если же задержка не уменьшилась, значит, проблема не в сигнале;
- установить ретранслятор или Mesh-систему;
- подключить компьютер к роутеру кабелем;
- сменить канал, на котором работает роутер. Интерференция сигналов многих соседей-пользователей Wi-Fi, работающих на одном и том же канале, также может быть причиной высокой задержки;
- сменить диапазон частот. Если позволяет роутер и беспроводной адаптер, попробуйте подключиться к сети 5 ГГц.

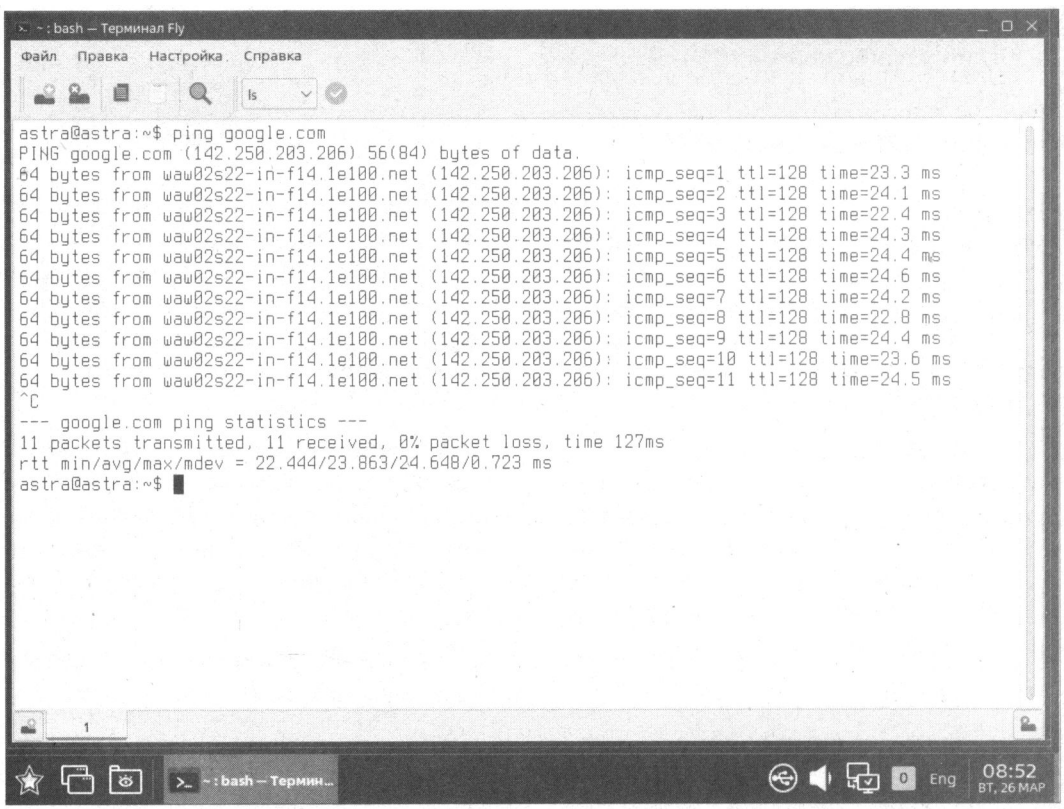

Рис. 10.1. «Пинг» сайта **google.com**

Алгоритм диагностики с помощью `ping` выглядит так:

1. Пропингуйте интересующий вас узел. Если ответ получен, и время каждого запроса не превышает 200 мс, значит, все в порядке.
2. Если ответ не получен, попробуйте пропинговать общеизвестный узел вроде google.com. Если с этим узлом все в порядке, значит, нужно разбираться, в чем проблема с нужным вам узлом. Возможно, он просто выключен. Если же вы знаете, что он включен, тогда проблема может быть в брандмауэре (на вашем компьютере, на маршрутизаторе, на сервере провайдера или же на целевом узле — вполне вероятно, что он не хочет принимать запросы с вашего IP-адреса). Возможно, также, что причина в «упавшем» маршруте. Допустим, маршрут пакетов к нужному вам узлу проходит через узел X, который в текущий момент недоступен. В этом случае нужно изменить маршрут. Самый простой вариант — использовать какой-нибудь прокси-сервер.
3. Если ответ получен, но время каждого пакета превышает 200 мс, это свидетельствует о проблеме с сетью. Проблема может быть как с вашей локальной сетью — например, плохо работающей беспроводной сетью, так и с одним из узлов по пути следования пакетов.

10.2. Команда *traceroute*

Когда вы устанавливаете соединение с каким-либо узлом, пакеты не отправляются непосредственно на этот узел. Они посылаются вашему шлюзу по умолчанию (как правило, это ваш роутер или компьютер, который выполняет его функции), который передает эти пакеты маршрутизатору провайдера, а тот, в свою очередь, передает пакеты дальше, и так повторяется, пока пакет не достигнет получателя. Непосредственная передача пакетов между вашим и целевым узлом возможна только в локальной сети, когда оба компьютера находятся в одной подсети, и для отправки пакетов не нужно, чтобы они выходили за ее пределы.

Очень полезной для диагностики сети является команда `traceroute`. Она показывает маршрут прохождения пакетов к интересующему вас узлу, а также время прохождения пакета через каждый узел.

С помощью `traceroute` вы можете:

- определить точку сбоя в сети (найти узел, который не хочет пропускать ваши пакеты);
- обнаружить задержку и потери пакетов в сети;
- получить информацию о маршруте следования пакетов;
- выявить проблему с сетевыми настройками или маршрутизатором.

Посмотрите на рис. 10.2 — на нем показан маршрут прохождения пакетов к узлу example.com. Сразу можно сказать, что с маршрутом все хорошо, о чем свидетельствуют низкие задержки на протяжении всего маршрута. Чем выше задержка, тем больше времени на этом узле пробыл ваш пакет. Здесь видно, что наш шлюз имеет

IP-адрес 192.168.0.1, и обращение к нему происходит практически мгновенно, — с локальной сетью точно все в порядке. Далее видна небольшая особенность провайдера, и почему так у него устроено — никто не знает. Всего же на пути следования пакета имеется 30 хопов (узлов).

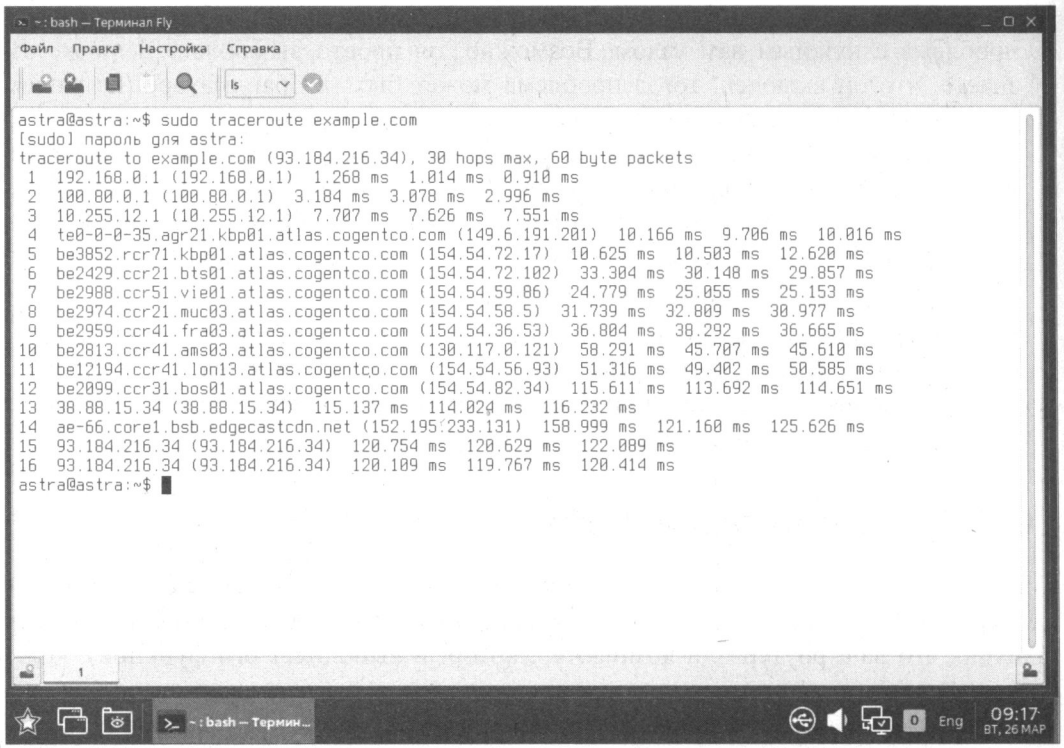

Рис. 10.2. Использование команды `traceroute`

Не всегда все проходит так гладко. На пути прохождения пакетов могут быть высокие задержки. Если это в вашей локальной сети, значит, нужно искать причину. Если на стороне провайдера, надо жаловаться в службу поддержки, а вот если задержка вызывается узлами уже после провайдера, вы мало на что можете повлиять. Разве что попытаться использовать прокси, чтобы изменить маршрут следования пакетов — вдруг это поможет.

Также вы можете увидеть сообщения вроде network unreachable или host unreachable. Первое свидетельствует о проблемах с настройкой сети, а второе — о проблемах с целевым узлом. Убедитесь, что вы правильно указываете его имя.

Отсутствие ответа от какого-то узла на пути следования пакетов означает, что он либо сломался/выключен, либо блокирует соединение от вас. Здесь, пока не свяжешься с администратором удаленного узла (что не всегда возможно), не поймешь, в чем причина. По крайней мере, `traceroute` поможет выяснить, с каким узлом возникла проблема.

10.3. Команда *ifconfig*

С помощью команды `ifconfig` (рис. 10.3) можно узнать IP-адрес вашего сетевого интерфейса, а также статистику использования пакетов, — например, сколько пакетов/байтов отправлено (TX packets), а также сколько получено (RX packets).

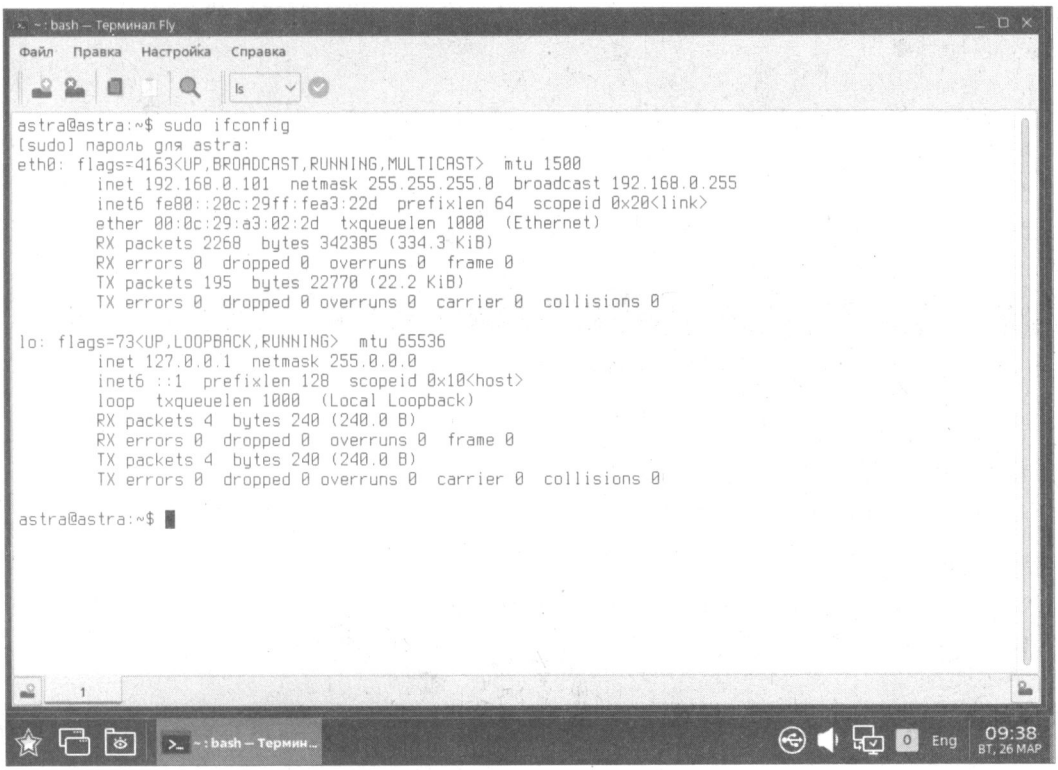

Рис. 10.3. Работа команды `ifconfig`

Если IP-адрес не назначен, значит, есть проблемы с настройками сети. Возможно, соединение не установлено (это характерно для соединений Wi-Fi). Возможно, что-то с роутером, и DHCP-сервер не назначает компьютеру IP-адрес. Так что настройку сети лучше в этом случае проверить.

10.4. Команда *ip*

Команда `ifconfig` сейчас считается устаревшей, и на смену ей пришла команда `ip`, объединяющая в себе несколько других команд. У нее много параметров, но чаще всего мы будем использовать параметры `link show` или `-br link show`. Второй набор параметров является сокращенным вариантом `link show` — команда показывает только статус интерфейса, его MAC-адрес и некоторые характеристики. Часто нам нужно узнать именно MAC-адрес, что очень удобно сделать командой: `ip -br link show` (рис. 10.4).

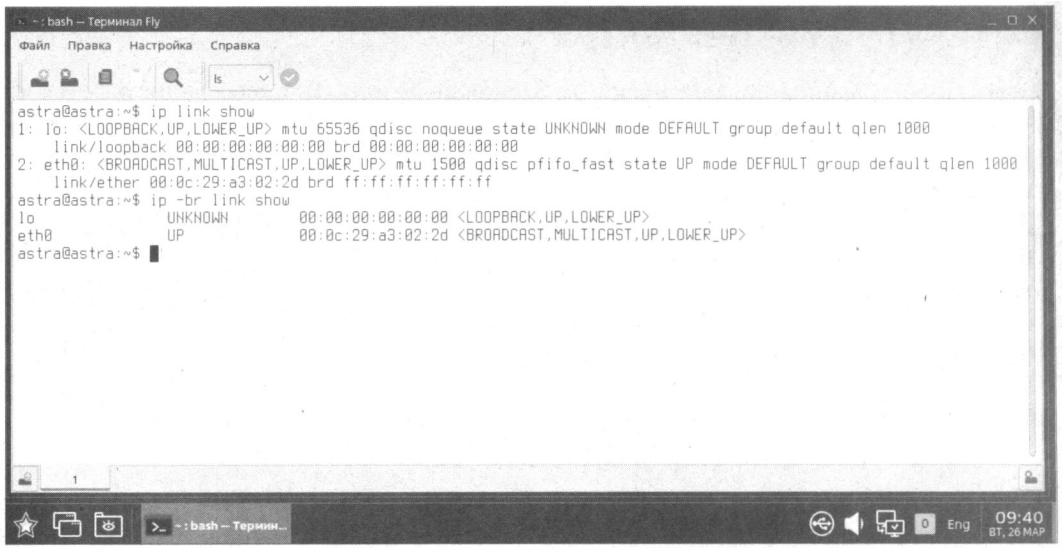

Рис. 10.4. Команда `ip -br link show`

10.5. Команда *nmcli*

С помощью команды `nmcli device status` можно вывести список соединений NetworkManager и соответствующие им сетевые устройства (рис. 10.5). Здесь видно, что соединению с названием **Wired connection 1** соответствует устройство **eth0**, а также выводится состояние этого соединения — **подключено**.

Рис. 10.5. Команда `nmcli device status`

10.6. Проблемы с разрешением доменных имен

Если вы можете пропинговать узел по его IP-адресу, но не можете по доменному имени, значит, налицо проблема с разрешением доменных имен. Первым делом введите команду:

```
sudo nslookup <имя узла>
```

Вы получите ответ от вашего сервера DNS. На рис. 10.6 показано, что доменное имя разрешил в IP-адрес узел с IP-адресом 192.168.0.1, который и является нашим DNS-сервером. Укажите в настройках соединения другой IP-адрес DNS-сервера. Вы можете использовать DNS-адрес провайдера или DNS от Google: 8.8.8.8 или 8.8.4.4.

После установки параметров соединения его нужно «выключить» и заново «поднять». Перезагружать компьютер по примеру Windows не нужно.

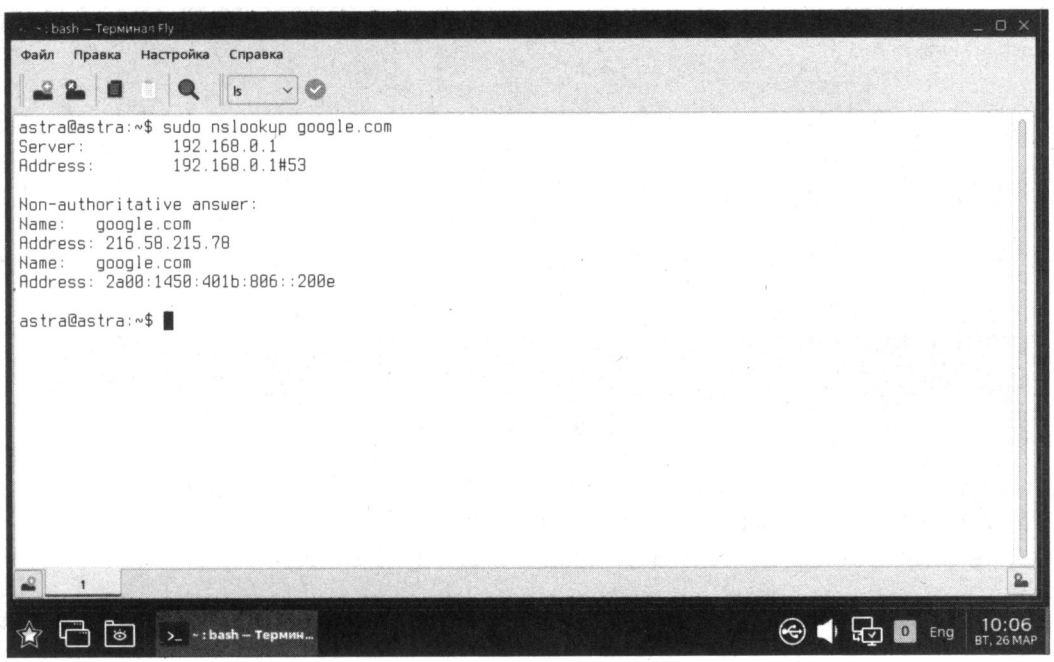

Рис. 10.6. Команда `nslookup`

10.7. Сканирование портов

Еще одна причина невозможности подключения к тому или иному сервису — закрытый порт. Например, вы можете пропинговать узел А, но не можете подключиться к нему как к веб-серверу. Это означает, что порт 80, который используется протоколом HTTP, закрыт. Узнать, открыт ли определенный порт на нужном вам узле, позволяет сканер портов `nmap`. Установите его:

```
sudo apt install nmap
```

Использовать `nmap` довольно просто:

```
nmap <имя или IP-адрес узла>
```

В ответ вы получите список открытых портов на указанном узле. На рис. 10.7 показано, что открыты порты 53, 80 и 443.

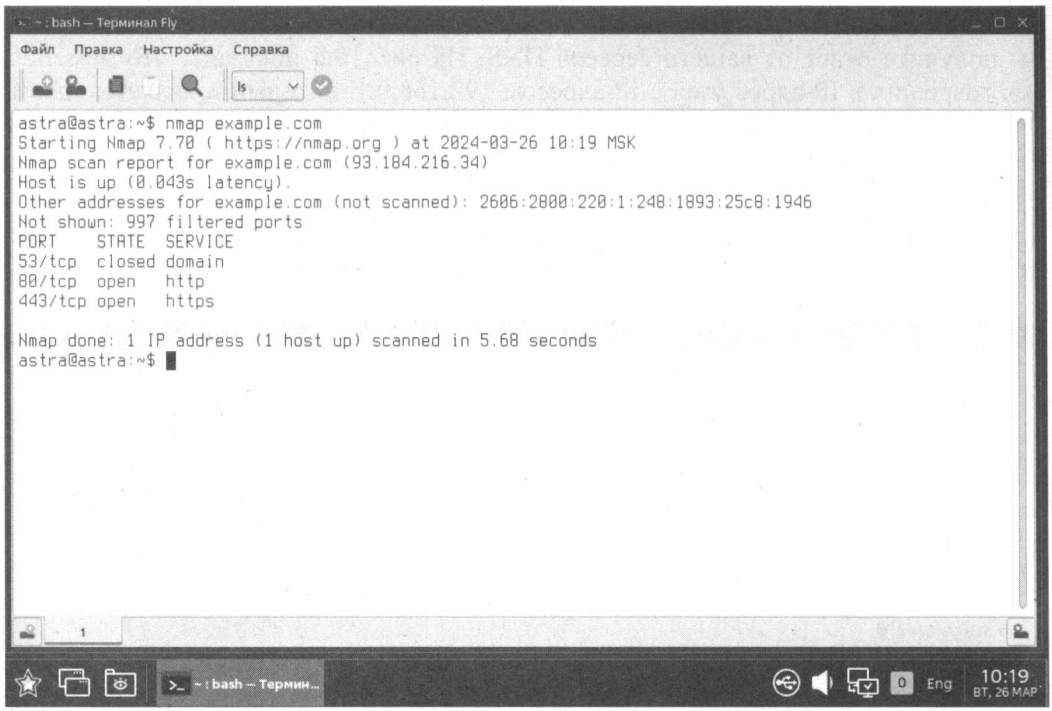

Рис. 10.7. Сканирование портов

У команды `nmap` очень много параметров, позволяющих задать диапазон сканирования портов, различные схемы сканирования и т. д. Обо всем этом вы сможете узнать на странице руководства, выполнив команду: `man nmap`.

ЧАСТЬ III

Приложения

ASTRA LINUX®

ГЛАВА 11

Установка пакетов

- Способы установки программ в Linux
- Репозитории пакетов
- Программы для управления пакетами

11.1. Способы установки программ в Linux

В Astra Linux доступны все способы установки программ, которые только могут быть в Linux, а именно: установка из пакетов, из исходного кода и с помощью снапов. Традиционный способ установки — установка из пакетов. Установка из исходного кода требует установки множества дополнительных компонентов: компилятора, утилиты для сборки, заголовочных файлов и прочих компонентов, которые обычному пользователю не нужны. Сегодня установка из исходников может применяться разве что энтузиастами или программистами, которые желают улучшить имеющиеся программы путем модификации их исходного кода.

Снапы — это относительно новый способ установки ПО, который будет рассмотрен в следующей главе. А в этой мы поговорим об установке пакетов.

Существуют два типа пакетов: RPM и DEB. Первые используются в RedHat-линейке дистрибутивов (Fedora, CentOS, SUSE и др.). Вторые — в Debian-линейке (Debian, Ubuntu, Astra Linux). Начиная с версии Debian 0.93, файл пакета *.deb представляет собой архив формата AR, содержащий саму программу, дополнительные ресурсы (например, страницы справочной системы man) и служебную информацию, включающую в себя информацию о зависимостях и конфликтах, а также инструкции, которые системе нужно выполнить при установке и удалении пакета.

Основное, что может представлять интерес для нас как для конечного пользователя, — это информация о зависимостях и конфликтах:

- *зависимости.*

  Программе для работы может требоваться какая-либо библиотека, без которой она не может запускаться, поскольку использует ее функции. Тогда в пакете указывается, что он *зависит* от другого пакета, содержащего эту библиотеку. При установке менеджер пакетов проверяет зависимости: если установлены не

все пакеты, от которых зависит устанавливаемый пакет, установка будет прервана — пока вы не установите все необходимое (некоторые менеджеры пакетов способны сами доустановить все необходимые по зависимостям пакеты). Впрочем, имеется возможность установки программы и без удовлетворения зависимостей (тогда информация о зависимостях будет просто проигнорирована), но в большинстве случаев установленная таким образом программа работать не станет;

- *конфликты.*

  Та или иная программа может в системе конфликтовать с другой программой. Например, программы sendmail и postfix являются серверами электронной почты — MTA-агентами (MTA, Mail Transfer Agent). Поскольку в системе может быть только один MTA-агент, установить можно или sendmail, или postfix, т. е. пакет sendmail конфликтует с пакетом postfix и наоборот.

Информация о зависимостях помогает понять, сколько дискового пространства понадобится для установки программы. Например, если вы используете в одном из дистрибутивов графическую среду KDE, попробуйте установить в нем GNOME-программу. И если сама программа может «весить» несколько килобайт, то из-за ее зависимости от среды GNOME менеджер пакетов предложит вам установить еще ряд дополнительных пакетов этой среды объемом в несколько гигабайт. Конфликты тоже представляют интерес, поскольку вы должны понимать, что нужная программа не будет установлена, пока вы не удалите несовместимые с ней программы. Конечно, можно установить пакет без удовлетворения конфликтов, но тогда могут перестать работать обе программы: и та, которую вы установили, и та, которая уже была установлена.

Некоторая информация о содержащейся в пакете программе, как правило, включена в само имя пакета. Сделано это исключительно для удобства — взглянув на название пакета, можно узнать версию программы и еще некоторую информацию о ней. Вот, например:

```
program-1.5-14.amd64.deb
```

Здесь `program` — название программы, `1.5` — ее версия, `14` — выпуск пакета, `amd64` — архитектура процессора, на которую рассчитана программа. Если программа независима от архитектуры, то указывается параметр `noarch`, — обычно так делают для документации и примеров конфигурационных файлов, т. е. для пакетов, содержащих информацию, которая не зависит от архитектуры.

11.2. Репозитории пакетов

Репозиторий — это хранилище пакетов. Репозиторий может быть *локальным* — например, каталогом на жестком диске или на любом другом носителе данных, или же *сетевым* — сервером в Интернете или в локальной сети, содержащим RPM- или DEB-пакеты. Для чего создаются репозитории? Для централизованного управления обновлением пакетов. Представьте, что у нас нет репозиториев. Тогда, чтобы узнать, вышла ли новая версия нужной вам программы, вам пришлось бы посещать

сайт ее разработчика или как минимум сайт разработчика дистрибутива Linux. А это не очень удобно. Один раз вы можете забыть проверить наличие обновлений, а потом вам вообще надоест это делать. Проще дождаться выхода новой версии дистрибутива и обновить все программы за один раз.

Так раньше и было. Вот вышла программа, ее включили в состав дистрибутива, но полностью не протестировали (да и невозможно предварительно протестировать все варианты использования любой новой программы). И в процессе эксплуатации программы выяснилось, что она работает неправильно, но только при некоторых условиях — например, с определенным форматом файла. Или же на платформе Linux был организован, например, веб-сервер. А через некоторое время оказалось, что в этой версии веб-сервера имеется «дыра», поэтому разработчики вскоре выпустили новую ее версию, эту «дыру» закрывающую. Пользователь же, установивший исходную программу веб-сервера, ничего не подозревая о том, что вышла новая ее версия, находился бы под угрозой взлома минимум полгода или даже год — до выхода следующей версии дистрибутива. А его сервер могли бы взломать уже на следующий день после обнаружения «дыры».

Но не тут-то было — разработчики Linux, заботясь о нас с вами, создали репозитории, с помощью которых можно быстро и удобно отслеживать обновления тех или иных пакетов. Причем это делает в автоматическом режиме сам менеджер пакетов, а вам остается лишь указать, какие обновления нужно загружать, а какие — нет. Практически все системы управления пакетами современных дистрибутивов поддерживают работу с репозиториями.

11.3. Программы для управления пакетами

В Astra Linux доступны четыре программы для управления пакетами:

- APT — традиционная программа для многих дистрибутивов, в том числе и не Debian-совместимых (в том же ALT Linux APT используется для управления RPM-пакетами);
- dpkg — классическая программа для установки пакетов, которая использовалась еще в самых первых версиях Debian. Она не умеет разрешать зависимости и используется, как правило, для установки одного пакета, скачанного извне (не из репозитория);
- Synaptic — графический менеджер пакетов, который является одной из самых удобных программ для их установки;
- aptitude — менеджер пакетов, предоставляющий псевдографический интерфейс для управления пакетами. Что-то вроде Synaptic, только работающего в консоли. Штука не очень удобная — в консоли как раз удобнее использовать apt, а если вам нужна оболочка — задействуйте Synaptic.

Программы dpkg и aptitude — это частные случаи. Первая подойдет, когда вы скачиваете пакет со стороннего источника и пытаетесь его установить в своей системе. То есть вам нужно самому скачать пакет, а затем ввести команду:

```
sudo dpkg -i <название пакета>
```

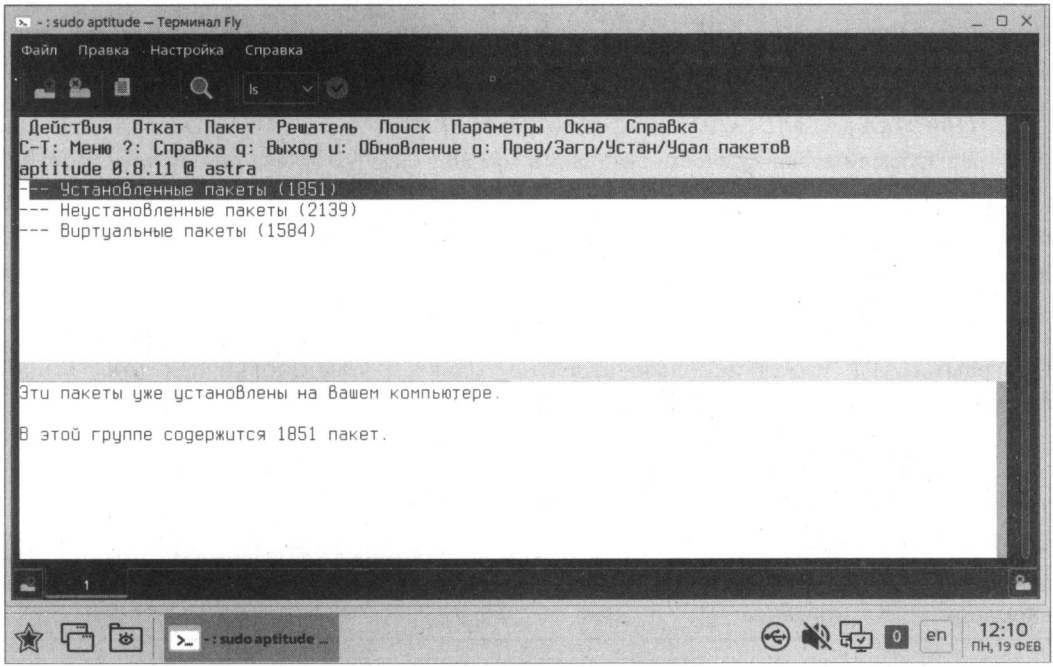

Рис. 11.1. Программа aptitude

Вторая — откровенно говоря, на любителя (рис. 11.1). Если она вам нравится — используйте.

11.3.1. Программа APT

Программа APT (Advanced Package Tool) — это программный интерфейс, используемый для управления инсталляцией и удалением программного обеспечения в Debian-совместимых дистрибутивах. Изначально программа была разработана разработчиками Debian.

Предположим, что мы устанавливаем некий пакет package.deb. Но в процессе установки обнаружилось, что для работы ему нужен пакет lib.deb, который в системе не установлен. Что ж, вы находите в Интернете недостающий пакет lib.deb, устанавливаете его, а затем заново устанавливаете пакет package.deb. Не очень удобно, правда?

Намного проще выполнить команду:

```
sudo apt install package
```

Программа APT просматривает файл /etc/apt/sources.list — в этом файле указаны источники (репозитории) DEB-пакетов, в качестве которых может выступать как внешний носитель, содержащий пакеты, так и сервер в Интернете. Программа находит указанный пакет, читает служебную информацию о нем, затем *разрешает зависимости* (т. е. устанавливает все другие пакеты, необходимые для работы программ устанавливаемого пакета), а затем устанавливает нужный нам пакет. Все за-

груженные программой APT и менеджером Synaptic (о нем — далее) пакеты записываются в каталог /var/cache/apt/archives.

> ***Очистка каталога/var/cache/apt/archives***
>
> Не забывайте периодически выполнять команду `sudo apt clean` для очистки этого каталога.

Чтобы просмотреть содержимое файла /etc/apt/sources.list, можно выполнить следующую команду:

```
sudo mcedit /etc/apt/sources.list
```

Программа APT может использоваться не только для установки пакетов. Общий формат вызова этой программы следующий:

```
apt [опции] команды [пакет]
```

Основные команды APT представлены в табл. 11.1.

Таблица 11.1. *Основные команды APT*

Команда	Описание
`update`	Синхронизирует файлы описаний пакетов (внутреннюю базу данных о пакетах) с источниками пакетов, которые указаны в файле /etc/apt/sources.list
`upgrade`	Обновляет указанный пакет. Может использоваться для обновления всех установленных пакетов, т. е. всей системы. При этом установка новых пакетов не производится, а загружаются и устанавливаются только новые версии уже установленных пакетов
`full-upgrade`	Обновляет дистрибутив. Для обновления всех пакетов рекомендуется использовать именно эту команду
`install`	Устанавливает один или несколько пакетов
`remove`	Удаляет один или несколько пакетов
`check`	Служит для поиска нарушенных зависимостей
`clean`	Используется для очистки локального хранилища полученных пакетов: перед установкой пакет загружается в локальное хранилище, а затем устанавливается оттуда. Эта команда может очистить хранилище для экономии дискового пространства
`list`	Показывает список пакетов на основе указанных имен
`show`	Показывает информацию о пакете
`search`	Производит поиск по описаниям пакетов
`autoremove`	Удаляет все неиспользуемые пакеты
`reinstall`	Переустанавливает пакеты

Рассмотрим несколько примеров. Представим, что мы хотим установить текстовый редактор. Произведем его поиск по слову `editor`:

```
sudo apt search editor | less
```

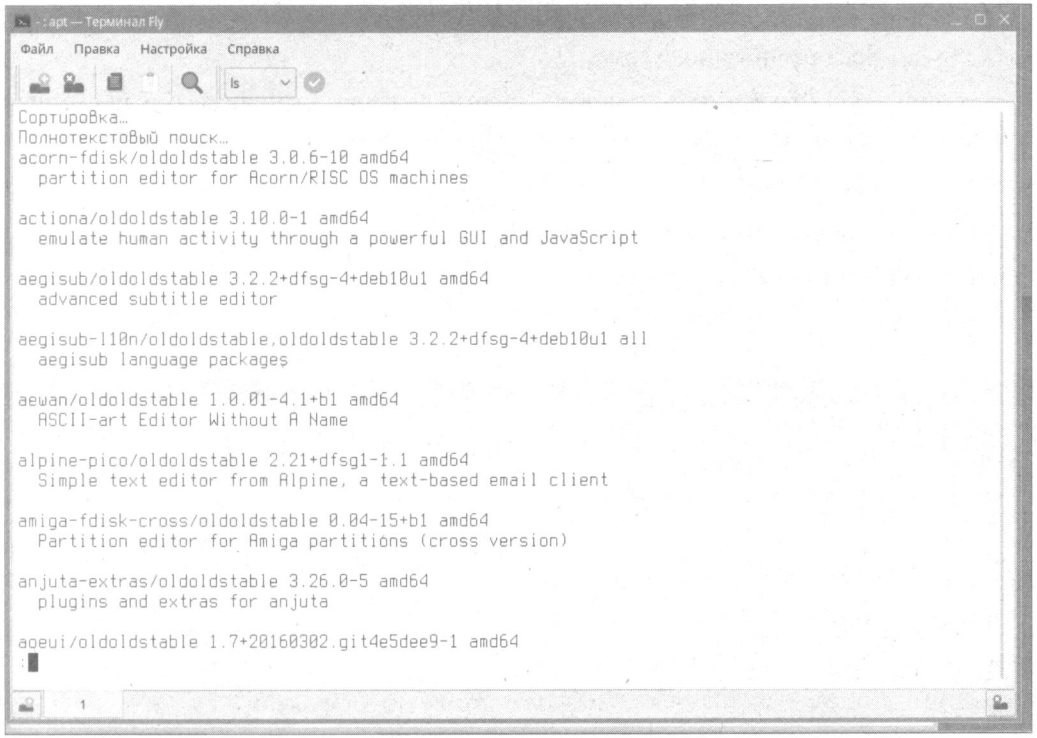

Рис. 11.2. Возможные варианты пакетов при поиске по слову `editor`

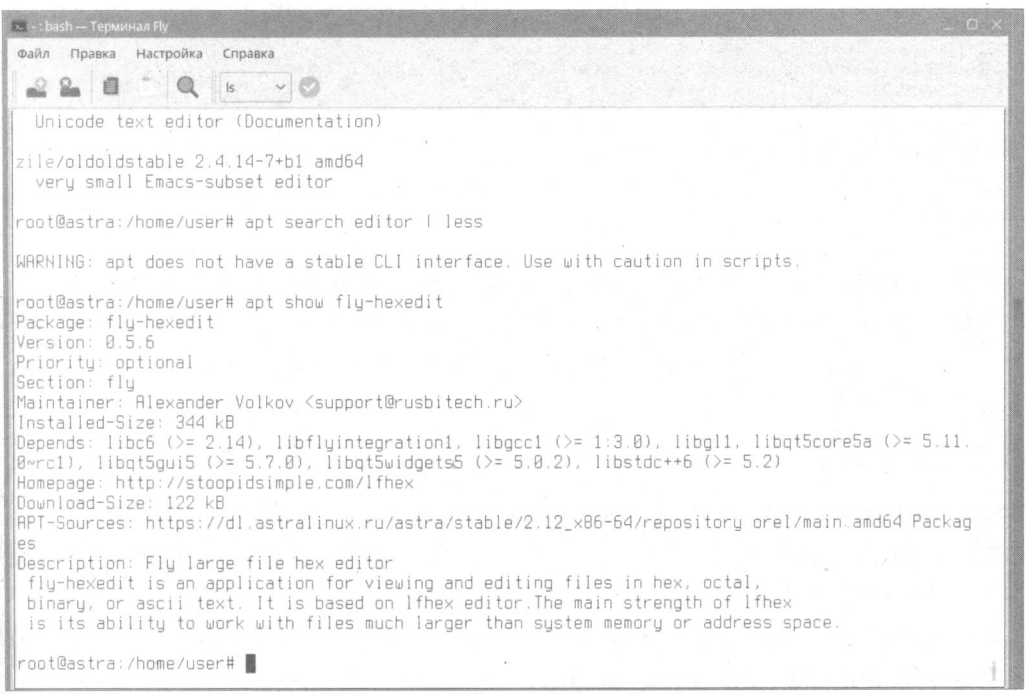

Рис. 11.3. Информация о пакете fly-hexedit

Вывод программы мы перенаправляем на команду `less` — вариантов по слову `editor` будет достаточно много (рис. 11.2).

Представим, что мы хотим установить шестнадцатеричный редактор и в списке нашли подходящий вариант — fly-hexedit. Просмотрим информацию о нем:

```
sudo apt show fly-hexedit
```

Эта команда выводит описание пакета (рис. 11.3).

Теперь можно установить пакет (рис. 11.4):

```
sudo apt install fly-hexedit
```

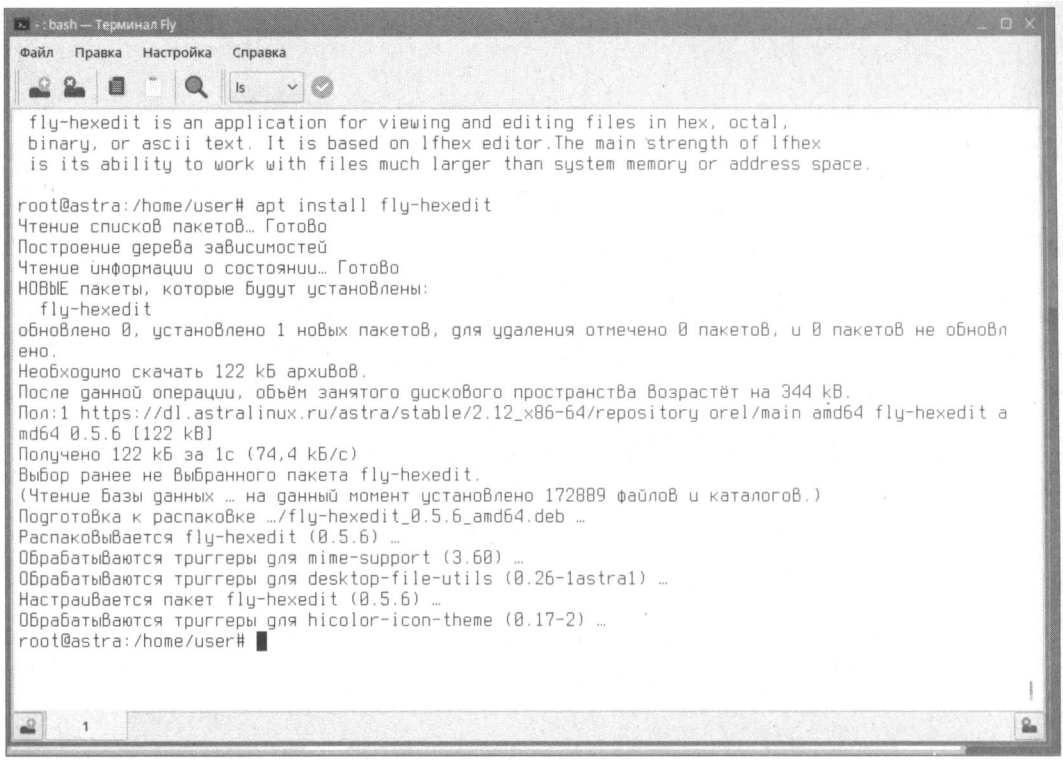

Рис. 11.4. Процесс установки пакета fly-hexedit

11.3.2. Графический менеджер пакетов Synaptic

Рассмотрим процесс установки программы с помощью менеджера Synaptic:

1. Запустите менеджер пакетов, выполнив команду **Меню | Системные | Менеджер пакетов Synaptic**.
2. Введите в открывшемся окне пароль администратора и нажмите кнопку **Да**.
3. Появится окно с кратким описанием программы. Чтобы больше не видеть его, нажмите кнопку **Закрыть**.

4. Нажмите кнопку **Поиск** и введите предполагаемое название искомого пакета.
5. Просмотрите результаты поиска. Если вас что-то заинтересовало, щелкните на названии пакета, чтобы прочитать его описание (рис. 11.5).

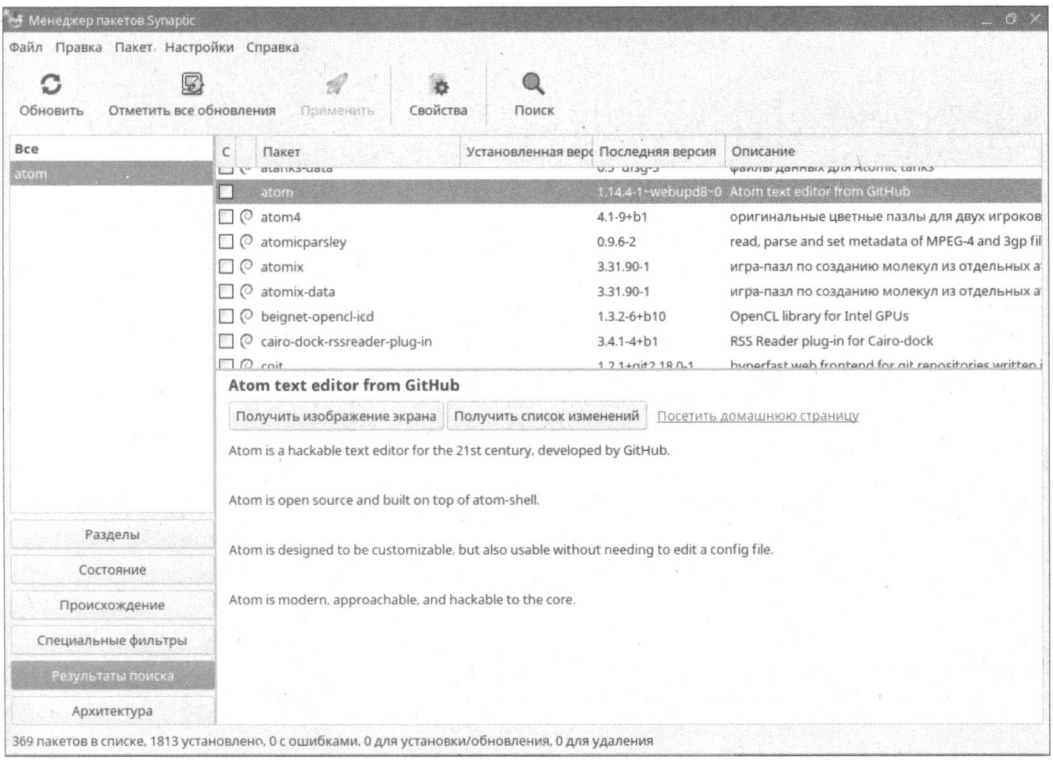

Рис. 11.5. Просмотр информации о пакете в менеджере Synaptic

6. Если подходящий пакет найден, щелкните на его названии правой кнопкой мыши и выберите команду **Отметить для установки**.
7. В случае, когда устанавливаемый пакет требует установки каких-либо дополнительных пакетов, менеджер отобразит их список. Нажмите кнопку **Применить**, если вы согласны установить дополнительные пакеты, или же кнопку **Отмена**, если вы передумали устанавливать выбранный пакет (рис. 11.6).
8. Нажмите на панели инструментов менеджера кнопку **Применить** (со значком ракеты).
9. В открывшейся панели **Краткое описание** (рис. 11.7) просмотрите представленную информацию (в том числе о том, сколько дискового пространства будет использовано) и нажмите там кнопку **Применить**.
10. Дождитесь установки пакетов (рис. 11.8) и нажмите кнопку **Закрыть**, когда установка пакетов будет завершена.

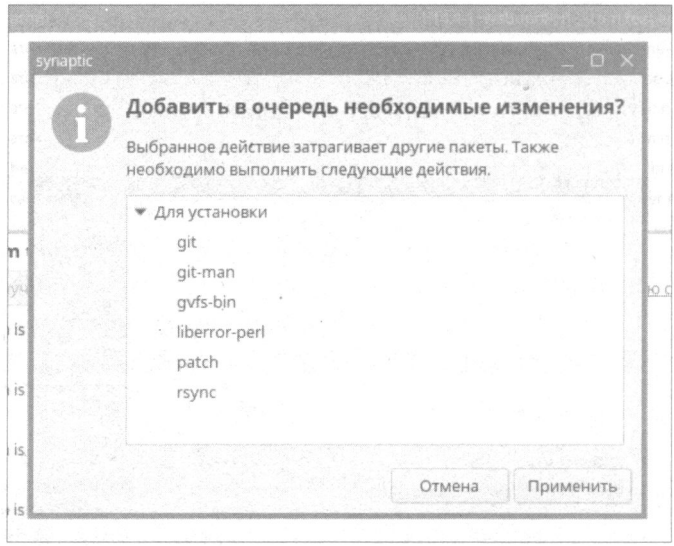

Рис. 11.6. Список дополнительных пакетов

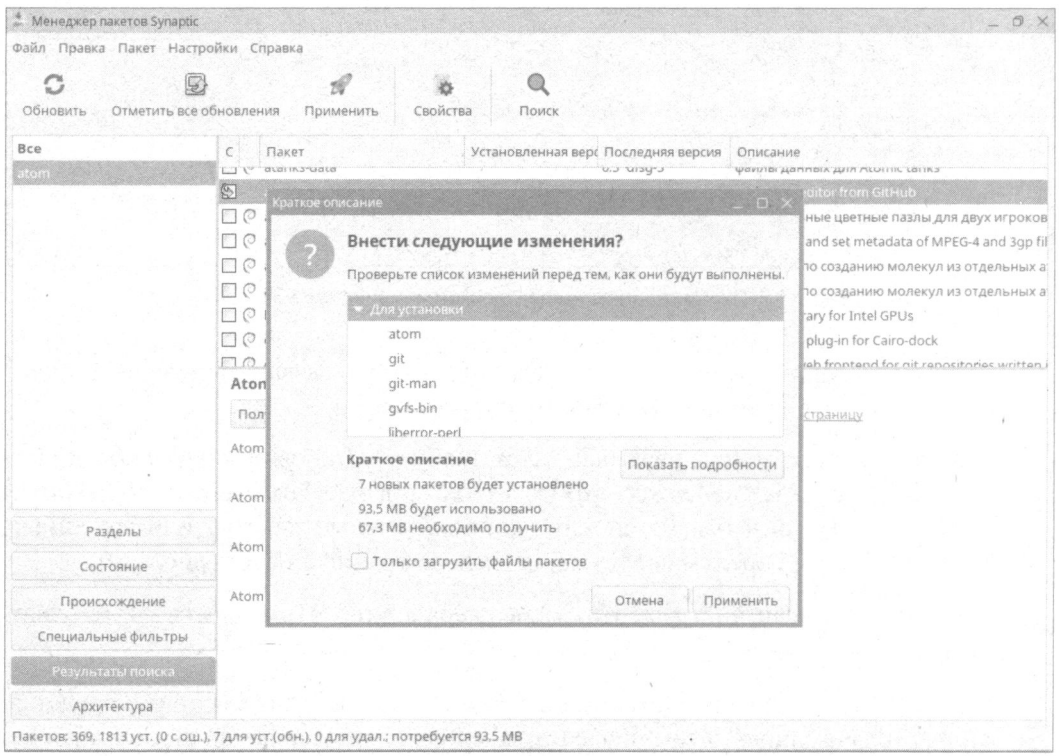

Рис. 11.7. Панель **Краткое описание**: нажмите в ней кнопку **Применить**

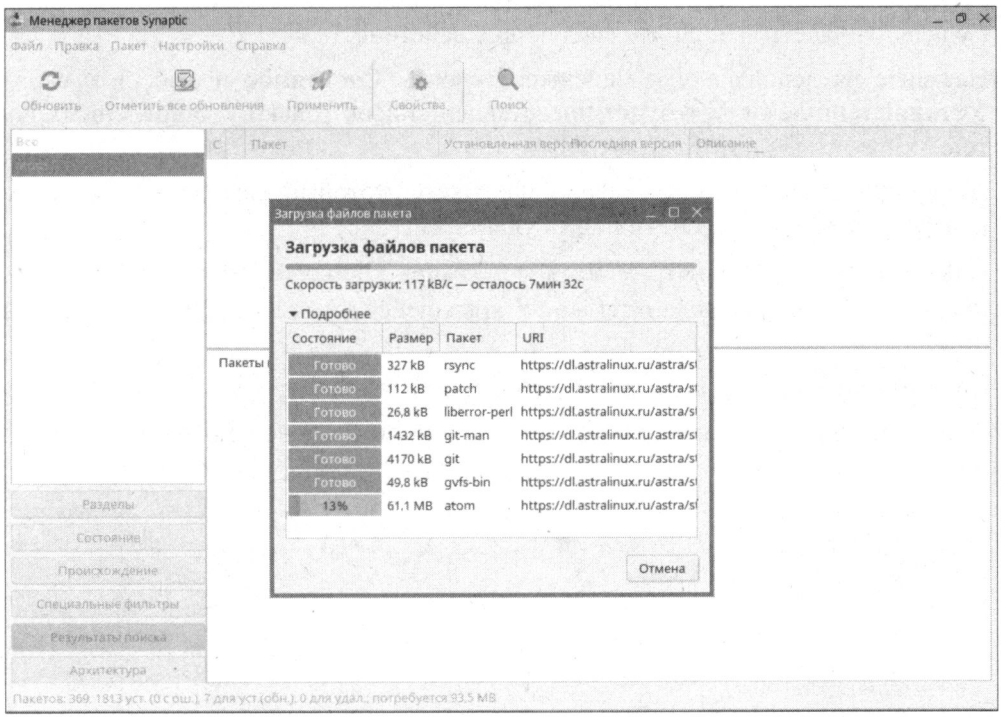

Рис. 11.8. Установка выбранных пакетов

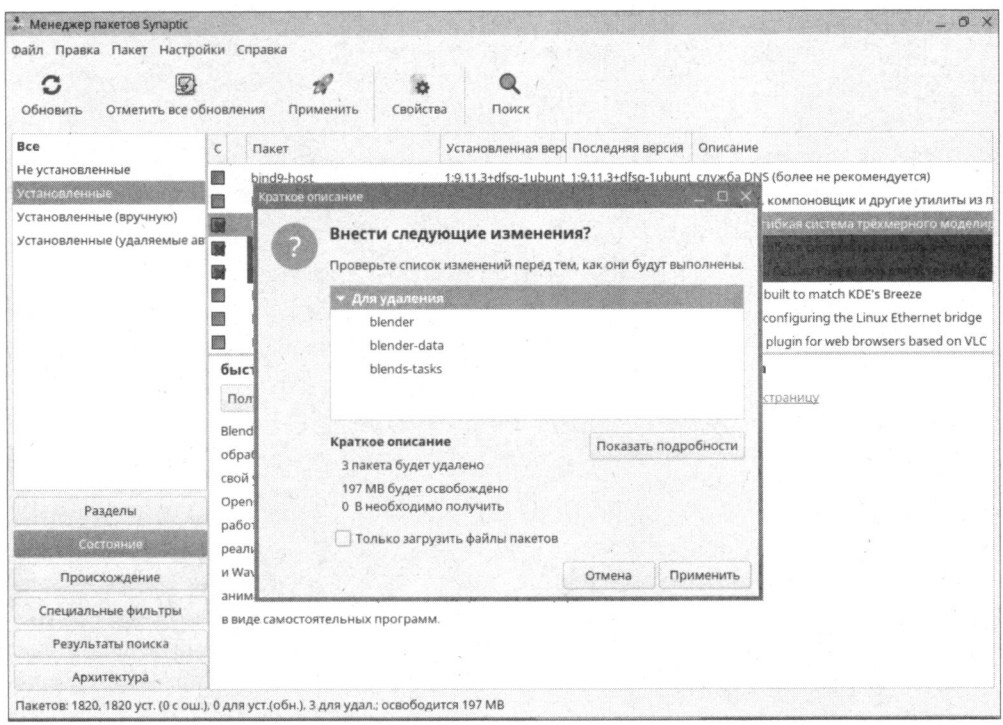

Рис. 11.9. Пакеты, отмеченные для удаления

Для удаления пакета выполните следующие действия:

1. Нажмите на левой панели менеджера кнопку **Состояние** и выберите вариант **Установленные** — вам будет представлен список только установленных пакетов.
2. Щелкните правой кнопкой мыши на пакете, который вам больше не нужен, и выберите команду **Отметить для удаления**.
3. Нажмите на панели инструментов менеджера кнопку **Применить**  , а затем нажмите кнопку **Применить** в открывшейся панели **Краткое описание** (рис. 11.9).
4. Нажмите кнопку **Закрыть**, когда выбранные пакеты будут удалены.

Как видите, работать с менеджером пакетов Synaptic довольно просто, и с этим справится даже самый начинающий пользователь.

ASTRA LINUX®

ГЛАВА 12

Снапы

- Что такое снапы?
- Подготовка к установке программ
- Установка популярных программ

12.1. Что такое снапы?

В предыдущей главе мы рассмотрели традиционный способ установки программного обеспечения в Linux — пакеты. Напомню, пакет — это подобие архива, который содержит саму программу, вспомогательные ресурсы (вроде документации и начального конфигурационного файла), а также инструкции, необходимые для установки и удаления пакетов. Кроме этого, пакет также содержит список зависимостей и конфликтов, т. е. список пакетов, от которых зависит устанавливаемый пакет и с которыми он конфликтует.

Казалось бы, все хорошо. Например, вы устанавливаете пакет, который требует библиотеку glibc версии 2.5 или выше. Если эта библиотека еще не установлена, то она будет установлена из репозитория при установке нужного вам пакета. Когда вы устанавливаете пакеты из репозитория вашего дистрибутива, все так и происходит, поскольку разработчики дистрибутива обычно не настолько глупы, чтобы добавлять в репозиторий программы и библиотеки несовместимых версий.

Все гораздо хуже, когда вы устанавливаете пакет, скачанный со стороннего источника, — например, с сайта разработчика или из репозитория другого дистрибутива (пусть и из родственного Debian). Тогда все может пойти уже не столь гладко. Пакету нужна, допустим, библиотека версии 2.7, но она не входит в состав репозитория вашего дистрибутива, а если ее взять из репозитория того же Debian, не факт, что установка пройдет гладко и уже установленные у вас ранее пакеты будут нормально работать с библиотекой версии 2.7. В общем, установка программы из стороннего пакета превращается в достаточно сложный квест. Из-за этого многие Windows-пользователи и не любят Linux. В Windows в большинстве случаев такого нет бывает, и даже вполне современные программы все еще можно установить на весьма древние системы вроде Windows 7/8.

Все эти проблемы решают *снапы* (snap) — относительно новый инструмент установки пакетов (по сравнению с традиционными пакетами). Как и традиционные пакеты, снапы представляют собой архивы-контейнеры, содержащие все необходимое для запуска и работы приложения. Например, если приложению нужна какая-либо библиотека, она напрямую включается в снап. При этом можно не бояться, что система превратится в свалку старых и новых версий библиотек, поскольку снап устанавливается как образ, защищенный от записи, в отдельную папку, — все изменения, которые снап должен внести в файловую систему (например, в каталог /lib), вносятся в виртуальную файловую систему. Поэтому установка снапа не влияет на основную систему. Это решает все проблемы совместимости с различными версиями приложения.

В итоге установка программы из снапа очень напоминает установку программы в Windows — вы просто указываете, какой снап хотите установить, и ждете, пока он загрузится.

Единственный недостаток этого подхода — далеко не все нужные программы бывают собраны в снапы. Тем не менее очень многие полезные программы можно установить с использованием снапов, и по крайней мере, с ними у вас не будет головной боли.

12.2. Подготовка к установке программ

Снапы — очень полезный инструмент, но почему-то разработчики Astra Linux не уделяют ему должного внимания, — демон snapd не только не установлен по умолчанию, но и сама его установка похожа на танец с бубнами.

Обратимся к официальной документации: «Установка системы управления пакетами Snap»[1].

Первым делом нужно открыть файл /etc/apt/sources.lst и добавить в него строку:

```
deb [trusted=yes] https://mirror.yandex.ru/debian/ buster main contrib non-free
```

> ***Еще о правах root***
>
> Для редактирования файла sources.lst вам нужны права root. Введите команду:
>
> ```
> sudo kate /etc/apt/sources.lst
> ```

Затем добавьте в него ключи подписей репозитория:

```
gpg --keyserver keyserver.ubuntu.com --recv-key 648ACFD622F3D138
gpg -a --export 648ACFD622F3D138 | sudo apt-key add -

gpg --keyserver keyserver.ubuntu.com --recv-key 0E98404D386FA1D9
gpg -a --export 0E98404D386FA1D9 | sudo apt-key add -

gpg --keyserver keyserver.ubuntu.com --recv-key DCC9EFBF77E11517
gpg -a --export DCC9EFBF77E11517 | sudo apt-key add -
```

[1] См. **https://wiki.astralinux.ru/pages/viewpage.action?pageId=187797332**.

После чего обновите список пакетов и установите демон snapd:

```
sudo apt update
sudo apt install snapd –y
```

Все бы хорошо, но в процессе установки вы увидите сообщение о том, что snapd требует версию библиотеки libc6 более новую, чем может быть установлена: нужна версия 2.27, система же может установить только 2.24 (рис. 12.1).

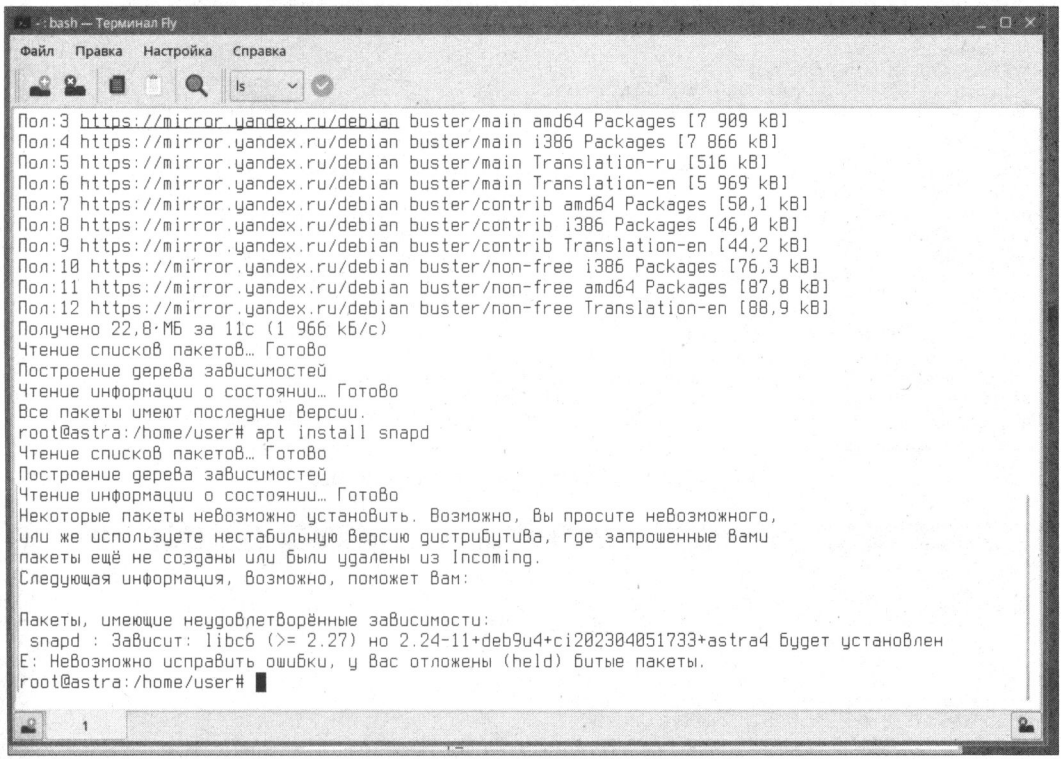

Рис. 12.1. Ошибка при установке пакета snapd

То есть для установки snapd нужно сначала обновить версию libc6 до более новой. Для этого введите команду:

```
sudo apt –t buster install libc6
```

Обратите внимание, что в результате выполнения этой команды будет установлена версия 2.28 (рис. 12.2).

Теперь можно установить snapd (рис. 12.3):

```
sudo apt install snapd
```

Установив демон snapd, можно попробовать установить с его помощью первый снап. Но пока мы не знаем, как он называется. Для поиска доступных снапов введите команду:

```
snap find <название>
```

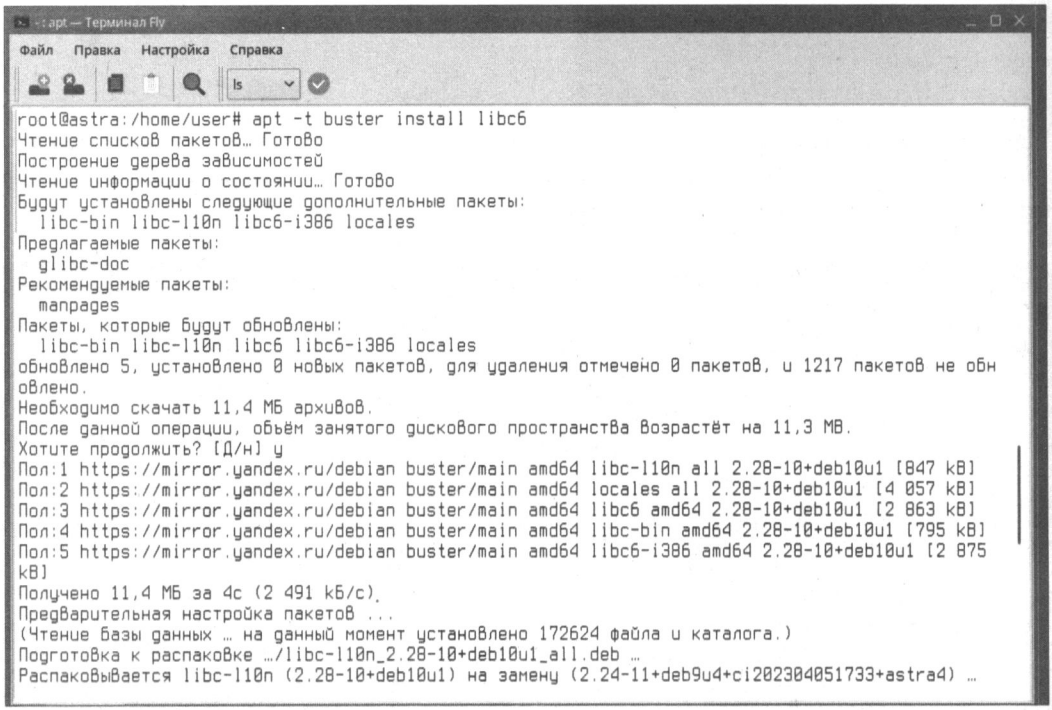

Рис. 12.2. Установка актуальной версии libc6

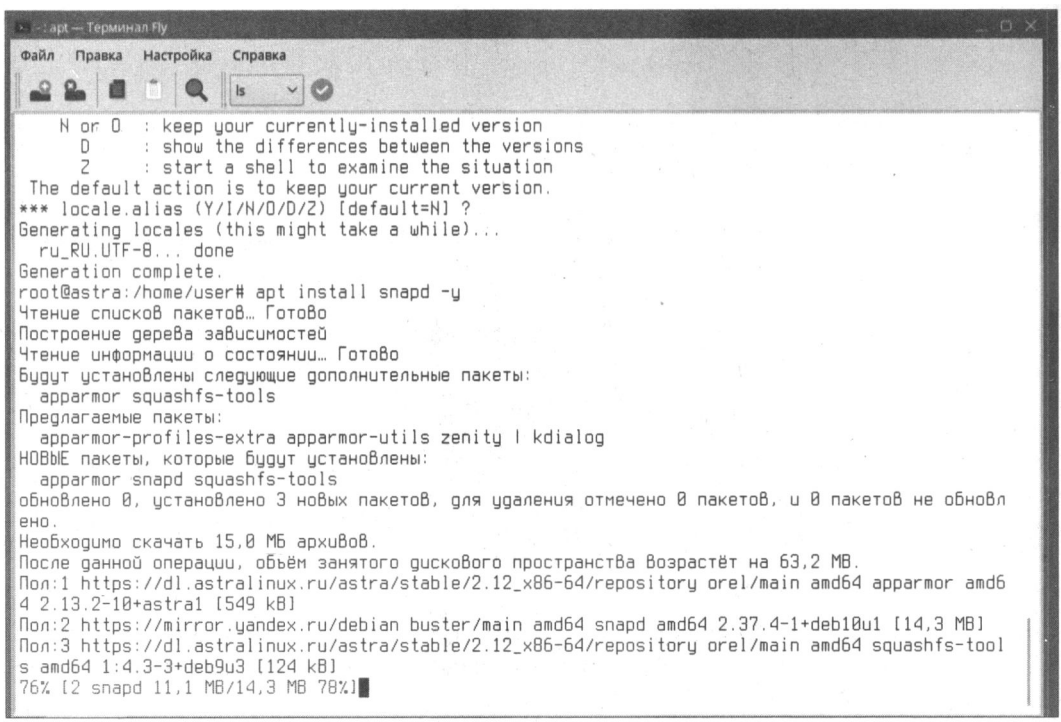

Рис. 12.3. Установка snapd

Например:

```
snap find hello
```

Вывод будет примерно таким (рис. 12.4):

```
hello-node-snap  1.0.2  bhdouglass  A simple hello world command
hello-mdeslaur   2.10   mdeslaur    GNU Hello, the "hello world" snap
hello-snap       0.01   muhammad    GNU hello-snap, the "Hello, Snap!" snap
hello            2.10   canonical   GNU Hello, the "hello world" snap
hello-world      6.3    canonical   The 'hello-world' of snaps
hello-sergiusens 1.0    sergiusens  hello world example
hello-gabriell   0.1    gabriell    Qt Hello World example
hello-bluet      0.1    bluet       Qt Hello World example
so-hello-world   0.2    shadowen    the old classic
hello-huge       1.0    noise       a really big snap
```

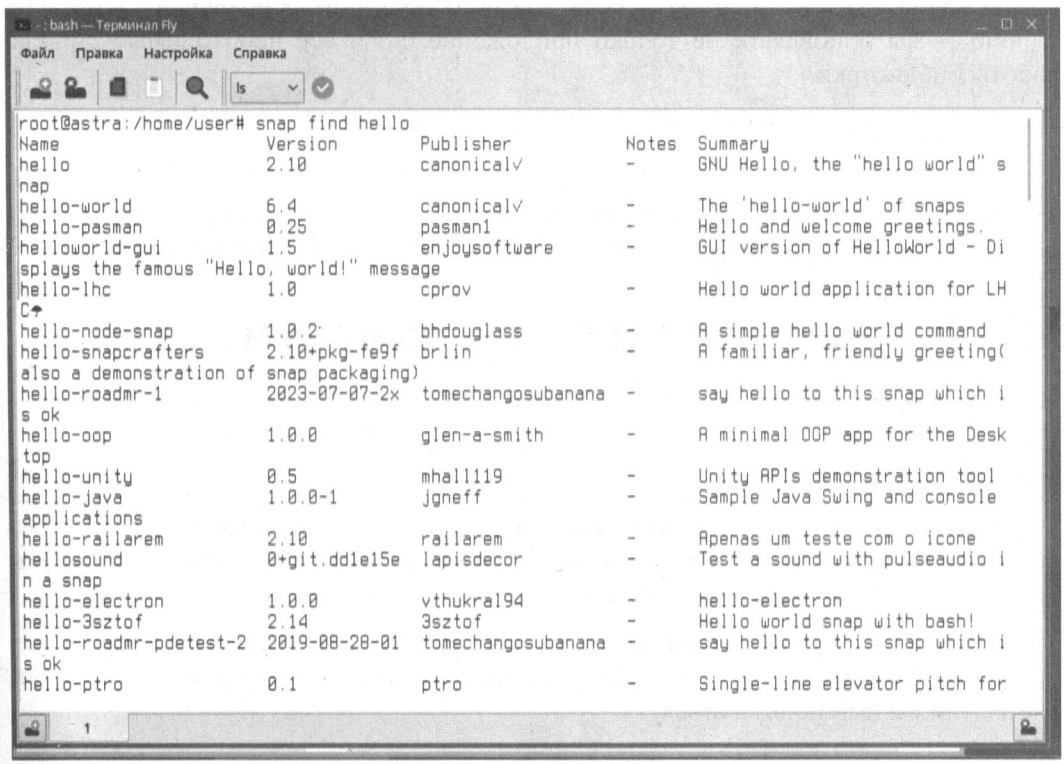

Рис. 12.4. snap работает в Astra Linux!

В левой колонке приводится название первого найденного снапа — hello. Давайте его и установим:

```
sudo snap install hello
```

Как видите, команда установки снапа аналогична команде установки пакета, только вместо команды `apt` используется команда `snap`.

Кстати, после установки снапа можно запустить имеющуюся в нем программу:

```
hello
Hello, world!
```

Просмотреть установленные снапы можно командой `snap list`:

```
snap list
Name      Version    Rev  Developer  Notes
hello       2.10     20   canonical  -
core        16.04.1423   canonical  -
```

Снап core — это базовая система снапов, среди всего прочего она как раз и содержит тот самый демон snapd, который вы установили ранее.

Как и пакеты, снапы можно обновлять. Например:

```
sudo snap refresh hello
```

Эта команда обновит снап hello, если для него доступны обновления. Это очень удобно — вы обновляете не только приложение, но и все необходимые для его работы библиотеки.

Для обновления всех снапов служит другая команда:

```
sudo snap resresh
```

Теперь вы знаете, для чего используются снапы и что это вообще такое. А вот обновлять систему целиком или использовать снапы — каждый решает сам.

12.3. Установка популярных программ

Прежде всего нужно установить снап core, необходимый для правильной работы многих программ:

```
sudo snap install core
```

Поскольку книга посвящена все-таки администрированию Linux, здесь не будет подробных инструкций — профессиональная аудитория не поймет излишней детализации, да и устанавливается все без каких-либо хлопот.

Таблица 12.1 содержит список команд, необходимых для установки популярных программ. Устанавливать их или нет — решайте сами, в зависимости от того, собираетесь ли вы ими пользоваться.

***Таблица 12.1.** Команды установки программ из снапов*

Команда	Что устанавливает
`sudo snap install atom`	Atom — удобный текстовый редактор, популярный среди разработчиков и просто опытных пользователей
`sudo snap install code`	Visual Studio Code — удобный и бесплатный редактор кода от Microsoft

Таблица 12.1 (окончание)

Команда	Что устанавливает
`sudo snap install skype`	Популярная программа для аудио и видеозвонков Skype
`sudo snap install telegram-desktop`	Настольная версия Telegram для Linux
`sudo snap install spotify`	Клиент для стримингового сервиса Spotify
`sudo snap install drawio`	Популярная рисовалка всяких схем
`sudo snap install signal-desktop`	Настольная версия мессенджера Signal
`sudo snap install viber-unofficial`	Неофициальная версия Viber, но вполне рабочая
`sudo snap install whatsapp-for-linux`	Linux-версия популярного мессенджера WhatsApp
`sudo snap install gnome-system-monitor`	Системный монитор GNOME
`sudo snap install termius-app`	SSH/SFTP/MOSH/Telnet-клиент в одном флаконе — Termius
`sudo snap install vidcutter`	Приложение для разрезки и объединения видеороликов

Таблицу 12.1 можно продолжать до бесконечности. Если вам стало интересно, посетите сайт **https://snapcraft.io/** — там вы найдете информацию о доступных снапах и инструкции для установки каждого из них.

ASTRA LINUX®

ГЛАВА 13

Запуск Windows-приложений в ОС Linux

- Программа Wine
- Виртуальная машина VirtualBox: установка и запуск

13.1. Программа Wine

13.1.1. Знакомство с Wine

Акроним Wine означает Wine Is Not an Emulator, т. е. «Wine — это не эмулятор». Программа Wine обеспечивает уровень совместимости, позволяющий запускать Windows-приложения на различных POSIX-совместимых операционных системах (Linux, macOS, BSD). Вместо эмуляции внутренней логики Windows, как это делают эмуляторы или виртуальные машины, Wine транслирует вызовы Windows API в подобные системные вызовы POSIX на лету, что позволяет интегрировать Windows-приложения в вашу систему.

Представьте, что вы установили виртуальную машину вроде VirtualBox и в нее в качестве гостевой операционной системы установили Windows. У этого решения есть серьезные недостатки:

- вы должны купить лицензионную копию Windows, которая будет работать внутри виртуальной машины.

  То есть вы используете бесплатную ОС Linux, а затем для запуска приложения Windows-приложений покупаете Windows и устанавливаете ее внутри виртуальной системы. Где логика? Тогда уж проще установить сразу Windows и забыть о Linux;

- перерасход дискового пространства.

  Вам нужно установить в качестве гостевой целую операционную систему, которая «скушает» несколько гигабайт дискового пространства, и все это только ради запуска 1–2 приложений из-под этой операционную системы? Не очень умное решение, на мой взгляд;

- вы получите две абсолютно разные и никак не связанные системы: одна гостевая — Windows, и одна основная — Linux.

Затем вам нужно будет организовать их взаимодействие по сети, чтобы хоть как-то обмениваться данными.

К преимуществам Wine можно отнести следующие:

- не нужна копия Windows — т. е. не придется покупать ее лицензию;
- нет перерасхода дискового пространства, поскольку не потребуется устанавливать копию Windows в виртуальную машину;
- если у вас хорошая видеокарта и вы установили драйверы для нее, возможен запуск Windows-игр.

Вот неполный список программ, которые вы сможете запустить в Wine: «1С:Предприятие», «Гарант», «КонсультантПлюс», Adobe Animate, Adobe Photoshop, Microsoft Office, Total Commander, World of Warcraft, Counter-Strike, Half-Life, Final Fantasy XI, The Sims 3 All.

Подробный список программ, работа которых подтверждается в Wine, доступен по адресу **https://appdb.winehq.org/**.

13.1.2. Установка Wine

Для установки wine достаточно установить пакет winehq-staging. Вы можете сделать это с помощью Synaptic или в командной строке — в зависимости от того, какой способ вам больше нравится (рис. 13.1)

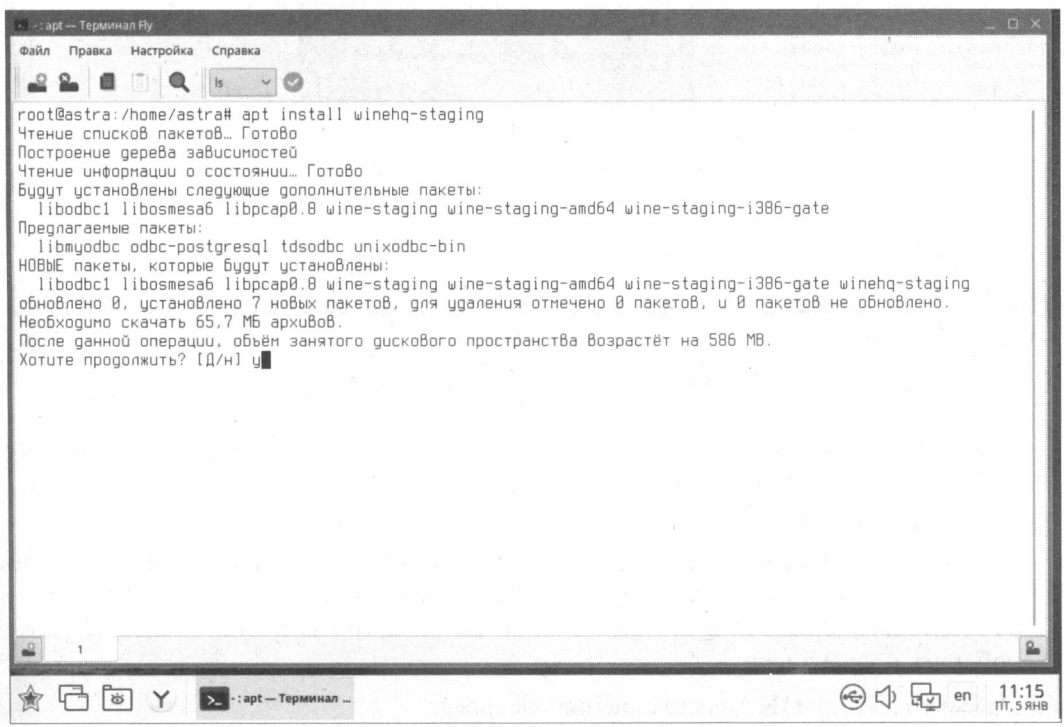

Рис. 13.1. Установка Wine

13.1.3. Настройка Wine после установки

Перед запуском в Wine Windows-приложений нужно сначала Wine настроить. Для этого выполните следующие действия:

1. Откройте терминал и убедитесь, что работаете под тем пользователем, под которым будете запускать Windows-приложения. Не используйте команду `sudo` для запуска конфигуратора, иначе созданная конфигурация будет создана для пользователя root.
2. Введите команду `winecfg`.
3. При первом запуске `winecfg` попросит установить пакет wine-mono, необходимый для запуска .NET-приложений. Нажмите кнопку **Установить** и дождитесь установки пакета (рис. 13.2).

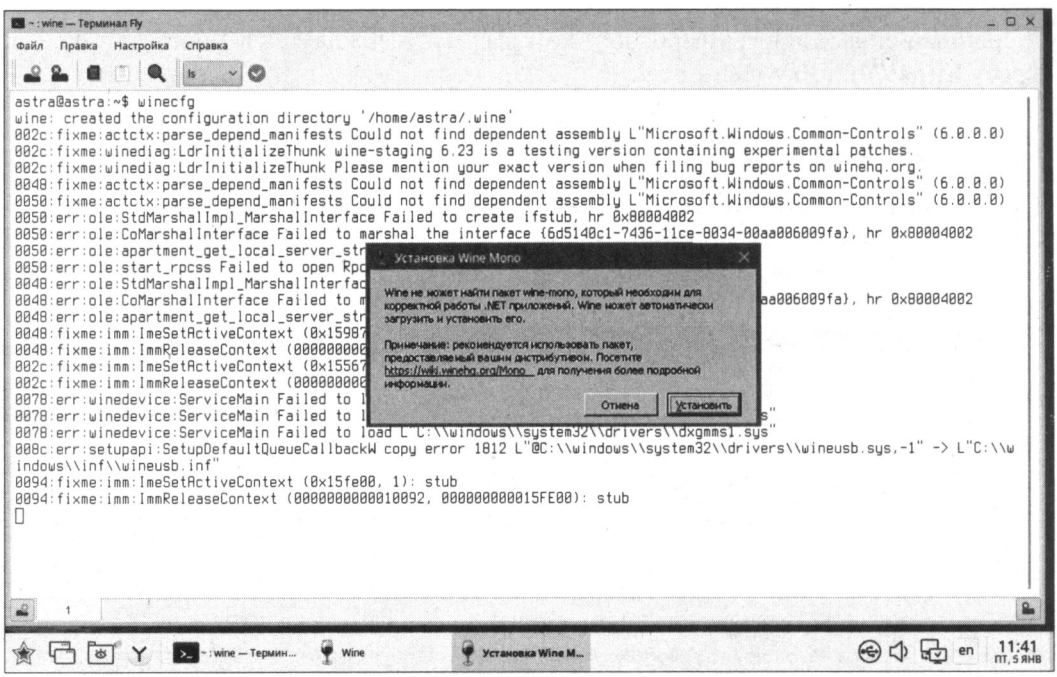

Рис. 13.2. Установка поддержки .NET

4. В открывшемся окне конфигуратора (рис. 13.3) выберите самую новую версию Windows. Пока еще в списке там присутствует Windows 10, но со временем может появиться и Windows 11.
5. Перейдите на вкладку **Диски**. Здесь содержится список дисков, которые доступны Windows-приложениям. По умолчанию содержимое диска C: будет храниться в каталоге ~/.wine/drive_c. При желании вы можете изменить каталог, а также добавить новые диски.
6. Нажмите кнопку **OK** для сохранения настроек.

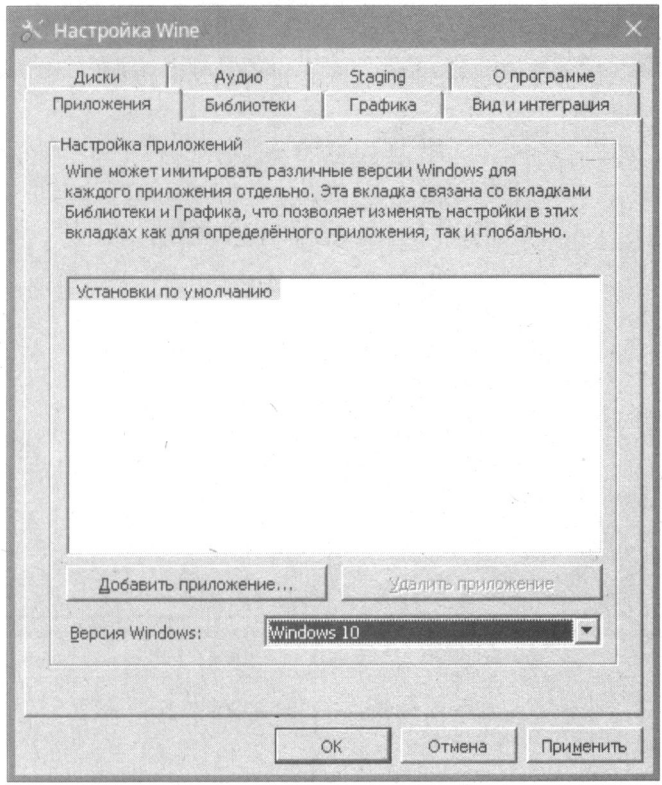

Рис. 13.3. Выбор версии Windows

13.1.4. Установка и запуск Windows-программы

Здесь в качестве примера будет показано, как с помощью Wine установить и запустить Windows-версию мессенджера Viber.

Для установки программы выполните следующие действия:

1. Скачайте установочный файл программы. В нашем случае он называется ViberSetup.exe.
2. Откройте терминал и перейдите в папку загрузки командой: `cd Загрузки`.
3. Запустите установку программы командой: `wine ViberSetup.exe`.
4. В открывшемся окне инсталлятора Viber (рис. 13.4) нажмите кнопку **Установить** и дождитесь установки программы (рис. 13.5). Окно запущенной в результате программы Viber показано на рис. 13.6.
5. Для повторного запуска уже установленной программы поищите ее в группе **Прочие** (рис. 13.7)

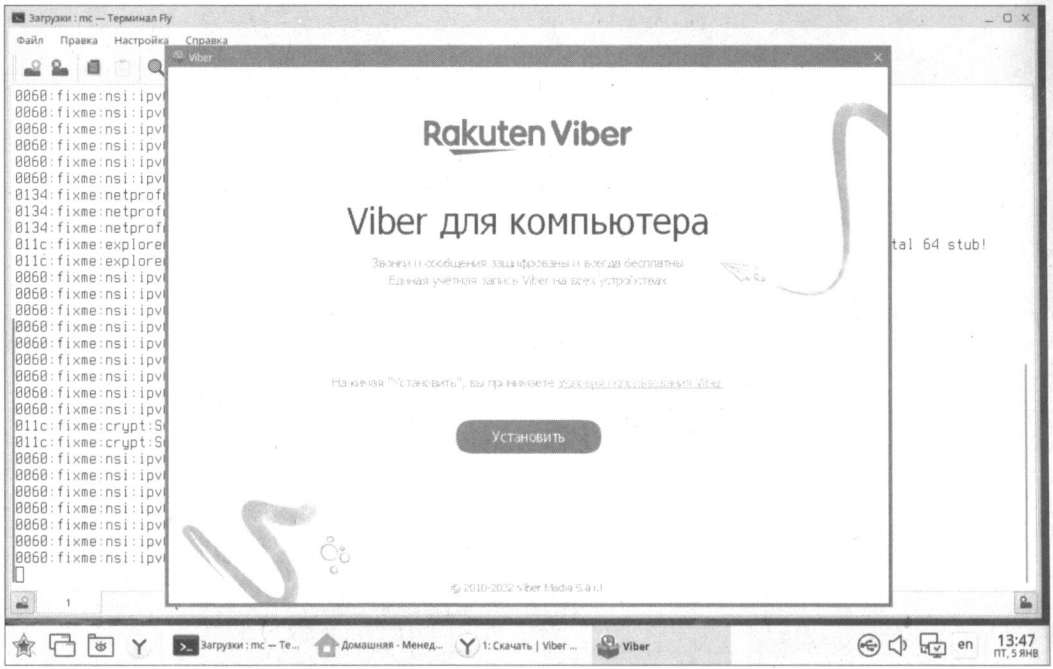

Рис. 13.4. Инсталлятор Viber

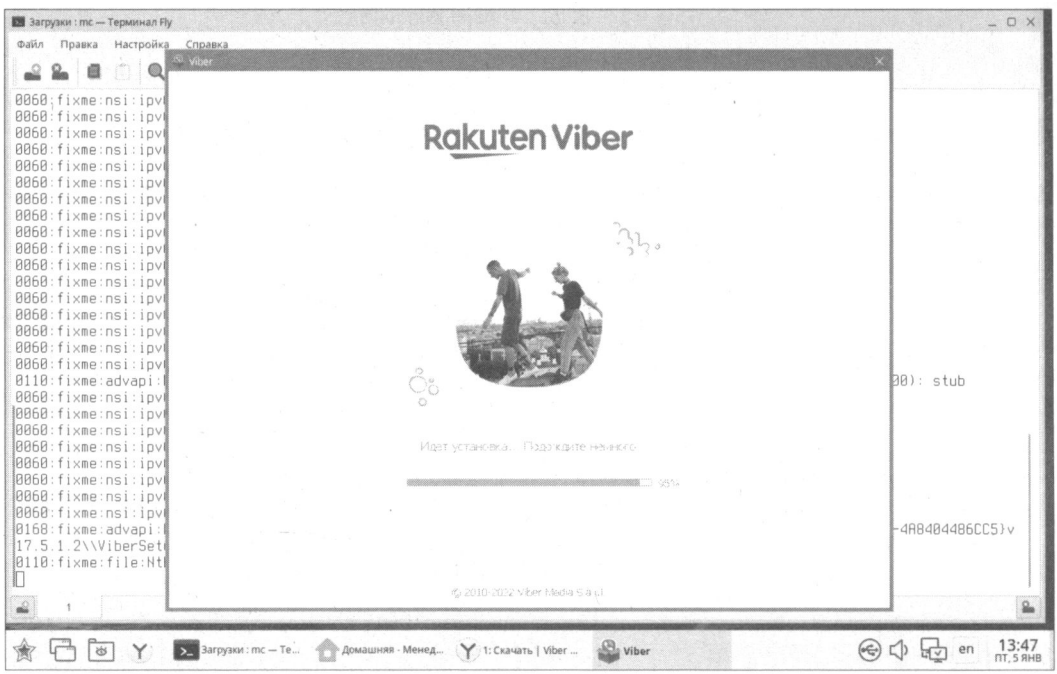

Рис. 13.5. Процесс установки Windows-программы

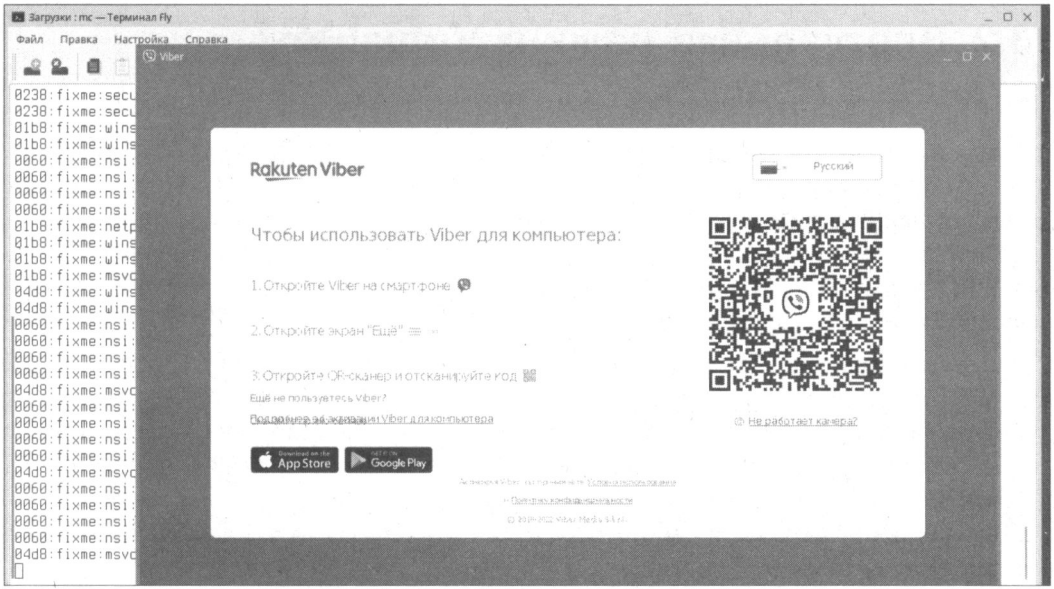

Рис. 13.6. Viber для Windows установлен и запущен

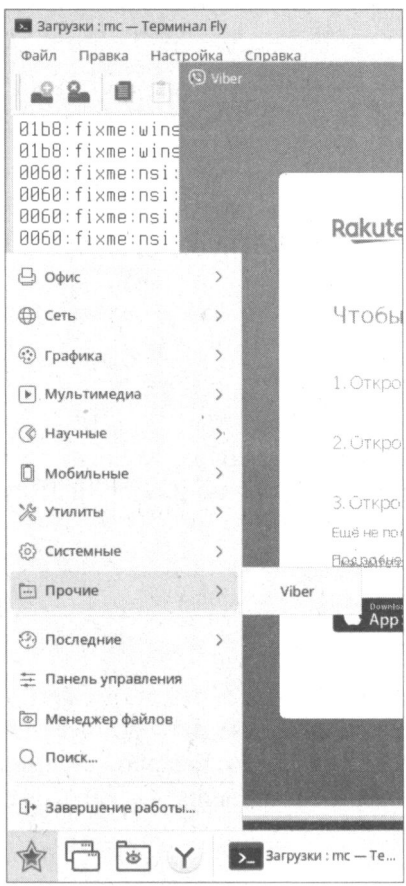

Рис. 13.7. Повторный запуск уже установленной программы

13.2. Виртуальная машина VirtualBox: установка и запуск

Что такое VirtualBox, думаю, пояснять не стоит. В отличие от Wine, VirtualBox позволяет создать виртуальную машину и установить в нее Windows. Преимущество такого решения в том, что все приложения будут работать так, как они, по сути, работают в Windows. Недостаток — в том, что вам понадобится лицензионная версия Windows. То есть тот факт, что вы используете «окна» в виртуальной машине, не освобождает от покупки лицензии (см. *разд. 13.1.1*).

Установка VirtualBox в Astra Linux Special Edition 1.7.5 — это боль. И чтобы ее для вас уменьшить, я написал этот раздел. Прежде чем приступить к работе, нужно отметить, что номер версии (1.7.5) упомянут не зря. В Community Edition и более старых версиях Special Edition используется ядро 5.*x*. Так вот у меня не получилось запустить виртуальную машину VirtualBox на этих ядрах — модули ядра VirtualBox попросту не хотели работать ни на 5.4, ни на 5.15. А вот на версии ядра 6.1 все завелось, но тоже не без танцем с бубном.

Итак, приступим. Сначала нужно установить пакет ca-certificates, если он не был ранее установлен:

```
sudo apt install ca-certificates
```

Затем добавить ключ для репозитория VirtualBox (вторая строчка — это одна команда):

```
sudo bash
wget https://www.virtualbox.org/download/oracle_vbox_2016.asc -O - | apt-key add
```

Теперь откройте файл /etc/apt/sources.list и добавьте в него строку:

```
deb [arch=amd64] https://download.virtualbox.org/virtualbox/
                                                    debian buster contrib
```

Обновите список пакетов:

```
sudo apt update
```

Чтобы сэкономить вам время, я пропущу сагу о libvpx. Коротко говоря, у меня уже была установлена эта библиотека версии libvpx6, но для VirtualBox нужна именно версия libpbx5:

```
wget http://ftp.ru.debian.org/debian/pool/main/libv/libvpx/
                                              libvpx5_1.7.0-3+deb10u1_amd64.deb

sudo apt install ./libvpx5_1.7.0-3+deb10u1_amd64.deb
```

Просто так ввести команду `apt install virtualbox` мы не можем, поскольку устанавливаем VirtualBox из его репозитария, а в нем есть разные версии (рис. 13.8), поэтому нам нужно уточнить устанавливаемую версию. Лучше устанавливать самую новую, которая и предлагается в репозитории, — 7.0 (рис. 13.9), поэтому вводим команду:

```
sudo apt install virtualbox-7.0
```

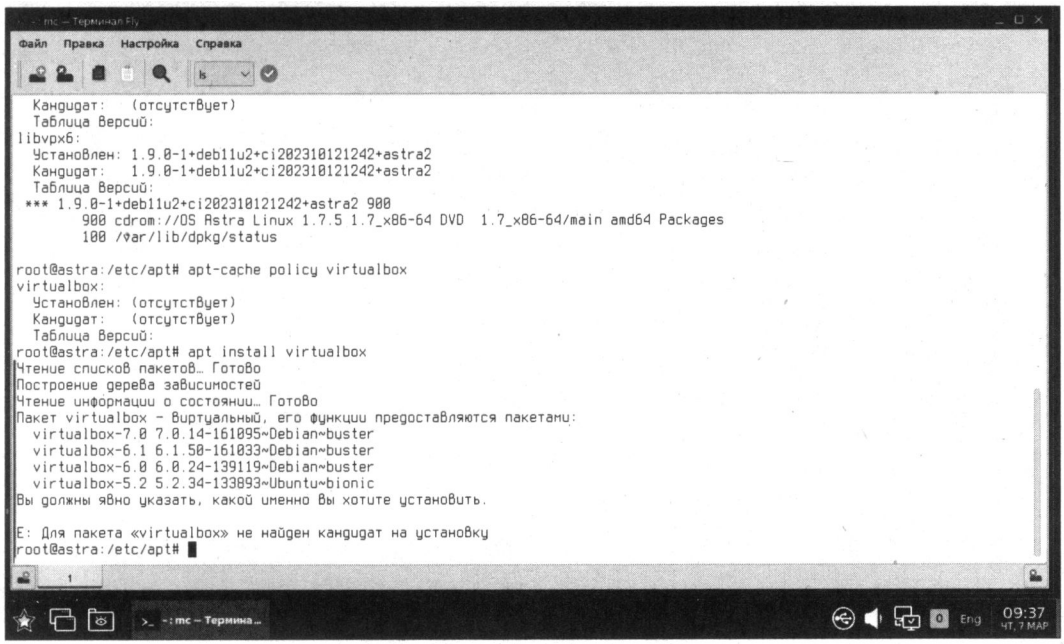

Рис. 13.8. Доступные версии VirtualBox

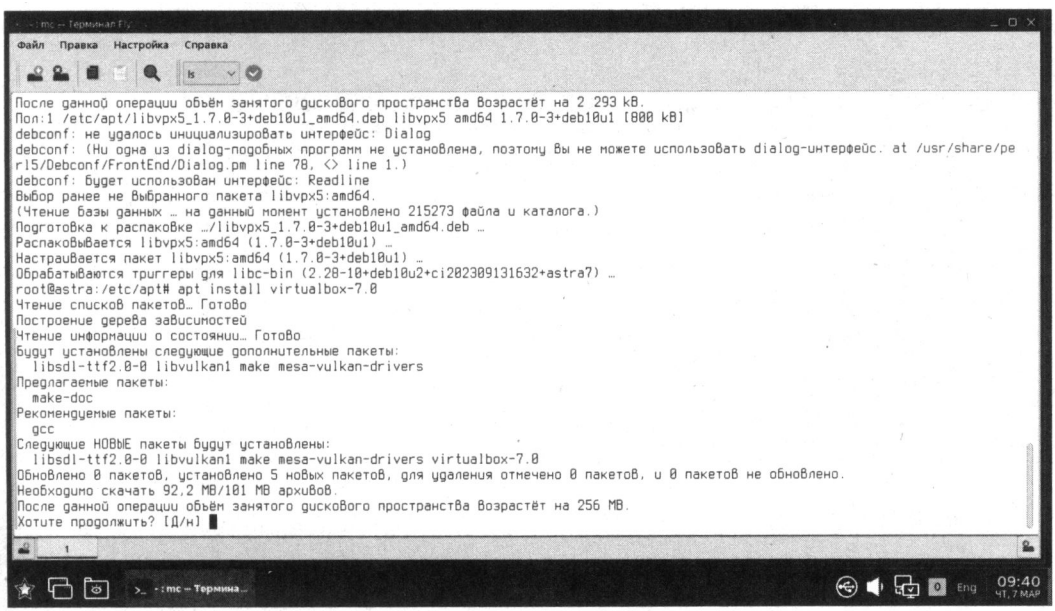

Рис. 13.9. Установка пакета virtualbox-7.0

Вроде бы VirtualBox установился (рис. 13.10), и можно создать виртуальную машину, нажав кнопку **Создать** (рис. 13.11). Однако при запуске создания виртуальной машины вы получите сообщение о том, что не установлен драйвер ядра (рис. 13.12).

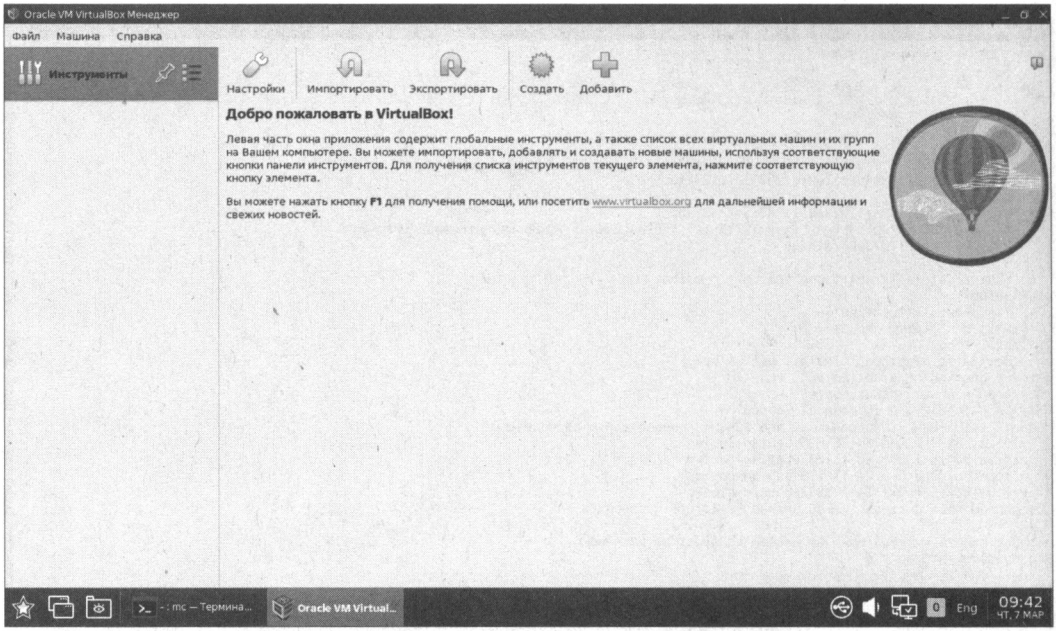

Рис. 13.10. VirtualBox установлен и запущен

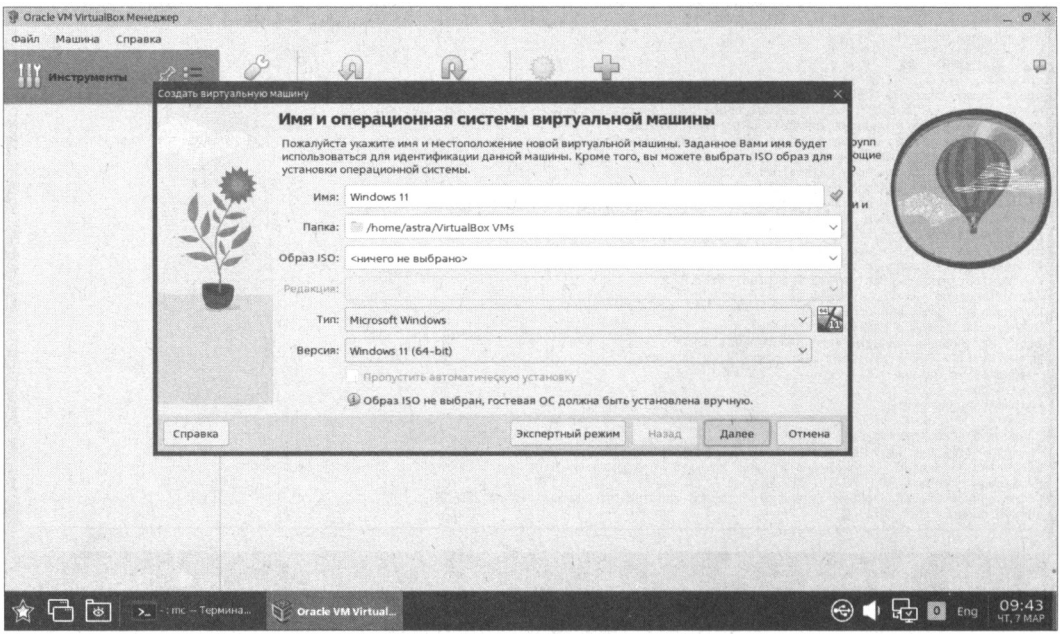

Рис. 13.11. Создание виртуальной машины

Это сообщение рекомендует нам запустить команду `/sbin/vboxconfig`. Что ж, сделаем это. Проверка модулей показала, что у нас не установлены пакеты gcc, make и perl. Я проверил — последние два пакета установлены. А вот gcc не установлен.

И установить его командой `apt install gcc` не получается — нет такого пакета. Чтобы поправить это, откройте файл /etc/apt/sources.list, закомментируйте репозиторий DVD-диска и раскомментируйте интернет-репозитории (рис. 13.13).

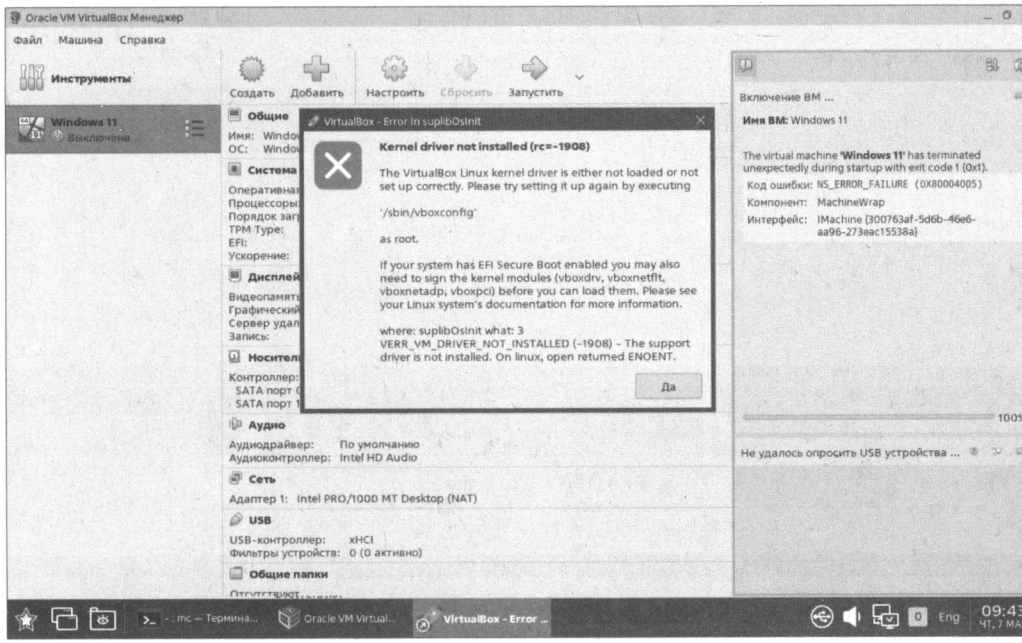

Рис. 13.12. Упс! Что-то пошло не так..

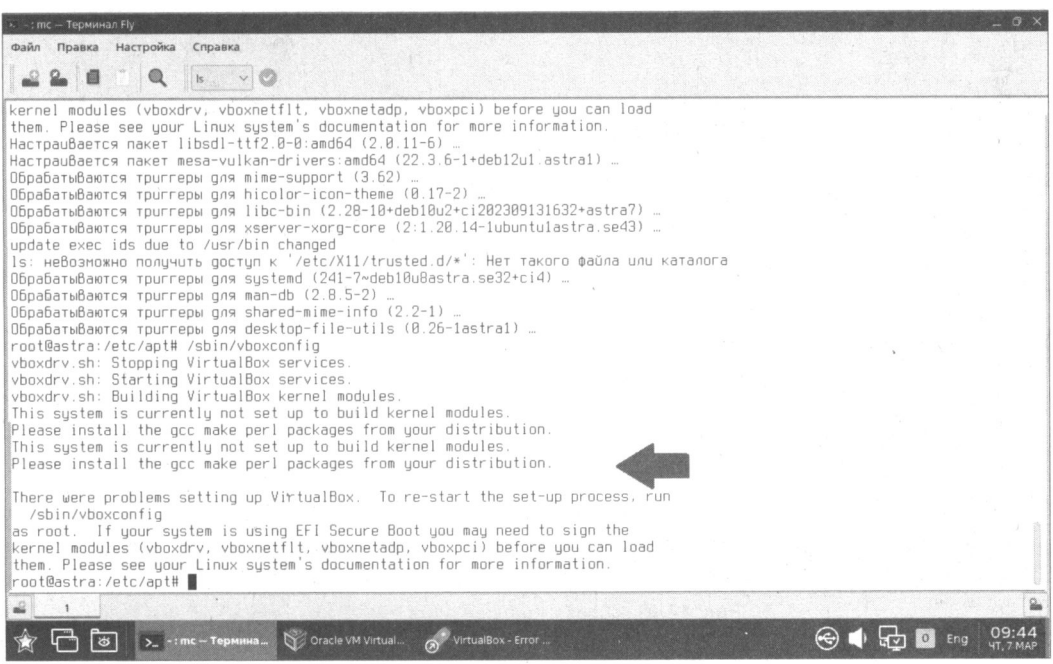

Рис. 13.13. Редактируем список репо

Рис. 13.14. Установка gcc

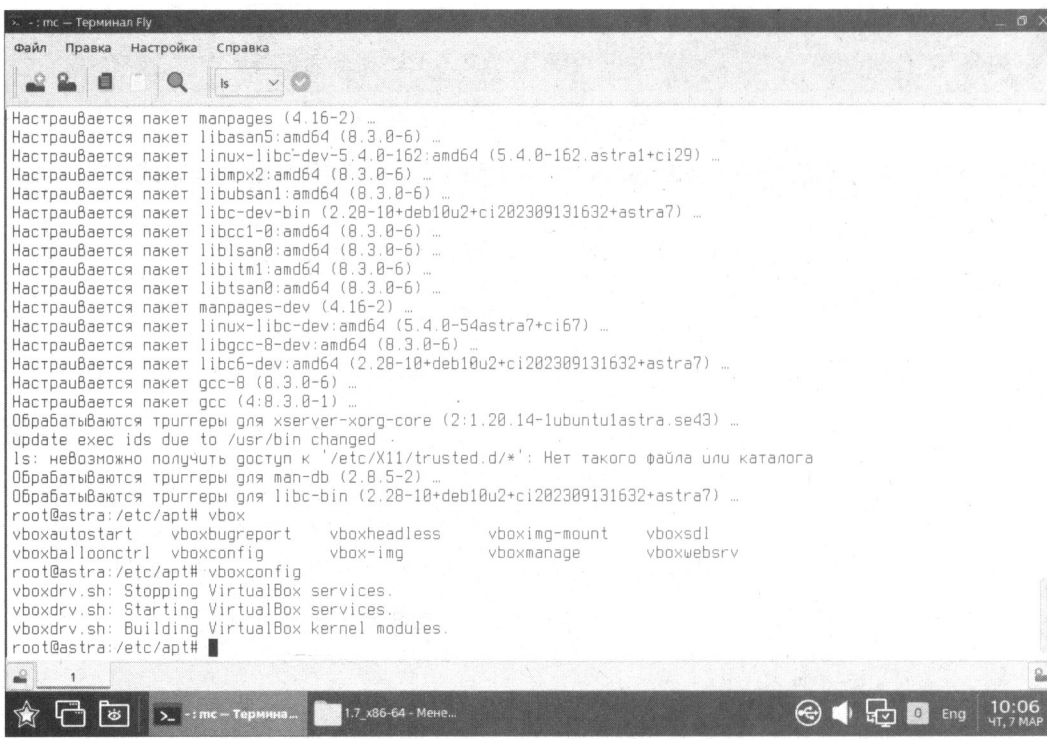

Рис. 13.15. Сборка модулей ядра VirtualBox

После этого нужно ввести команды:

```
sudo apt update
sudo apt install gcc
```

Наконец-то, gcc будет установлен. Повторите теперь ввод команды `/sbin/vboxconfig`. На рис. 13.14 показан процесс установки gcc, а на рис. 13.15 — процесс построения модулей ядра VirtualBox.

После этого нужно перезапустить само приложение и повторить запуск гостевой операционной системы. Все должно работать. Для сравнения — в Ubuntu для всего этого нужно ввести только команду установки пакета virtualbox...

ЧАСТЬ IV

Подсистема хранения данных

ASTRA LINUX®

ГЛАВА 14

Архитектура подсистемы хранения данных

- Виртуальная файловая система
- Имена устройств
- Идентификаторы UUID
- Поддерживаемые файловые системы

В этой главе мы разберемся, как устроена подсистема хранения данных «под капотом». Мы не станем погружаться в никому не нужные структуры ядра и тому подобные детали. Мы не программисты, и нам это не нужно. Но основные концепции мы все-таки рассмотрим.

14.1. Виртуальная файловая система

Для начала надо определиться с тем, что такое файловая система. *Файловой системой* называется способ представления информации на физическом носителей данных и часть операционной системы, предоставляющей средства для операций над файлами и каталогами. Может, это не самое точное определение, но оно отображает суть. Взять и записать данные на диск в определенном формате — этого явно недостаточно. Нужно, чтобы операционная система понимала этот формат и предоставляла соответствующие средства, а именно — системные вызовы для работы с файлами. Посмотрите на рис. 14.1. На нем показана подсистема хранения данных Linux.

Приложение может использовать функции glibc (библиотеки GNU C) или же напрямую задействовать системные вызовы ядра — тут уж как будет угодно программисту. Использовать функции glibc удобнее, но, обращаясь непосредственно к системным вызовам (например, `open()`, `read()`, `write()`, `close()`), можно немного повысить производительность приложения, — ведь вы минуете glibc, которая все равно использует те же системные вызовы.

Linux поддерживает файловые системы следующих типов:

- дисковые файловые системы — локальные файловые системы, основанные на использовании блочных устройств. К ним относятся ext2/3/4, xfs, reiserfs и т. д.;

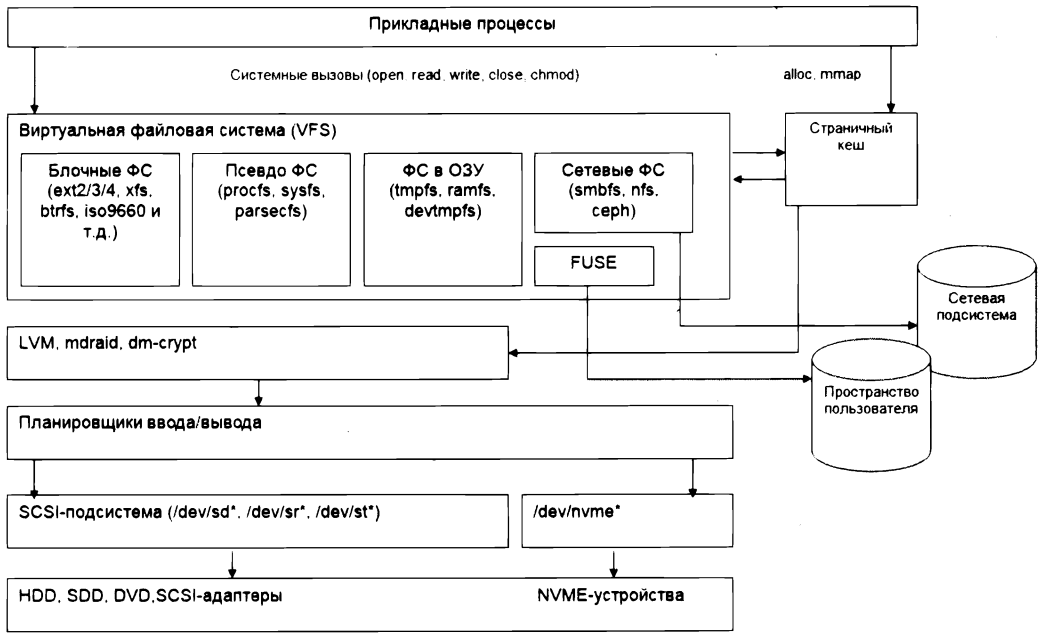

Рис. 14.1. Подсистема хранения данных Linux

- псевдофайловые системы — они обеспечивают доступ из пространства пользователя к структурам данных, которые хранятся в пространстве ядра. Вы можете получить информацию о процессах и даже тонко настроить операционную систему с помощью этих файловых систем (подробно мы их рассмотрим в *главе 19*);
- файловые системы в памяти — сюда относятся так называемые RAM-диски, временная файловая система (tmpfs) и др.;
- сетевые файловые системы — позволяют получать доступ к файлам, которые физически находятся на других устройствах.

Особняком стоит модуль ядра FUSE (Filesystem in USEr space). Он вместе с библиотекой libfuse позволяет разработчикам создавать код файловой системы, которая работает в пользовательском пространстве, и экспортировать эту файловую систему в пространство ядра. С помощью модуля FUSE реализована поддержка NTFS — NTFS-3G, а также sshfs.

VFS — это виртуальная файловая система. Она предоставляет процессам из пользовательского пространства (user space) возможность получать доступ к файлам и каталогам. Именно благодаря VFS мы можем добиться существующего сейчас уровня абстракции. Каждая файловая система имеет свои особенности. Если бы не было VFS, то пришлось бы разрабатывать разные версии системных вызовов для каждого типа поддерживаемой файловой системы: например, `open_ext2()` — для открытия файла, находящегося в файловой системе ext2, или `open_vfat()` — для ФС VFAT. Другими словами, VFS делает системные вызовы независимыми от типа используемой файловой системы.

VFS оперирует следующими объектами:

- суперблоками — содержат метаданные файловой системы;
- индексными дескрипторами — метаданные файла, иноды (inode);
- элементами каталога (directory entry) — связывают имя файла с индексным дескриптором;
- файловыми объектами — структурами, содержащими информацию об открытых файлах.

VFS кеширует метаданные (суперблоки, индексные дескрипторы и элементы каталога) посредством специального механизма, который называется *распределением slab* (slab allocation[1]). Утилита slabtop выводит статистику по использованию кешей slab (рис. 14.2).

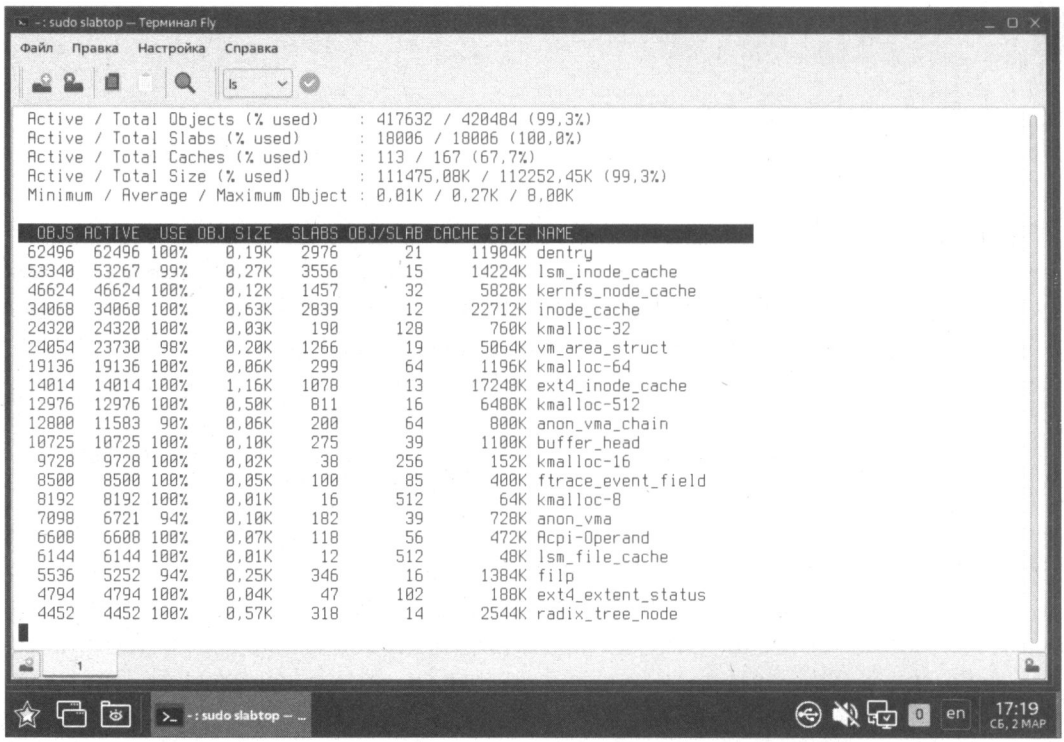

Рис. 14.2. Выполнение команды `sudo slabtop`

Очистить кеш можно командой:

```
# echo 3 > /proc/sys/vm/drop_caches
```

Управлять кешированием VFS можно посредством переменной ядра vm.vfs_cache_pressure (/proc/sys/vm/vfs_cache_pressure):

[1] См. **https://en.wikipedia.org/wiki/Slab_allocation**.

- 0 — ядро сохраняет кеш (не будет освобождать кеш вообще);
- 100 — значение по умолчанию;
- >100 — ядро будет активнее выгружать метаданные VFS.

После загрузочного сектора на стартовом диске компьютера находится суперблок, хранящий всю информацию о файловой системе. Размер суперблока — 1 Кбайт (1024 байта). В каждой группе блоков суперблок дублируется, что позволяет восстановить его в случае повреждения файловой системы.

Структура суперблока описана в файле /usr/src/linux/include/linux/fs.h:

```
struct super_block {
        struct_head s_list;   // двусвязный список всех смонтированных ФС
        unsigned long s_blocksize;
        struct file_system_type *s_type;
        struct super_operations *s_op;
        struct semaphore      s_lock;
        int s_need_sync_fs;
...
}
```

Теперь перейдем к группе блоков. После копии суперблока следует дескриптор группы блоков, который хранит информацию о физических «координатах» карт блоков и i-узлов (inode, информационный узел), а также таблицы i-узлов.

Карта блоков (block map) содержит информацию об используемых блоках и служит для поиска свободных блоков при выделении места для файла.

Каждому файлу соответствует только один i-узел, хранящий метаданные файла, — все атрибуты файла, кроме его имени. Карта i-узлов следует сразу после карты блоков. С помощью карты i-узлов можно определить, какой i-узел используется, а какой занят.

Кроме атрибутов файла, в i-узле хранится указатель на данные файла. Обычно это массив из 15 адресов блоков, 12 из которых непосредственно ссылаются на номера блоков, хранящих данные файла. Если данные занимают больше 12 блоков (напомню, что обычно 1 блок = 1 Кбайт), то будет использована косвенная адресация. Поэтому следующий адрес (13-й) — это адрес блока, содержащего список адресов других блоков, содержащих данные файла.

Ранее было сказано, что в i-узле хранится вся информация о файле, кроме его имени. Имя файла хранится в каталоге, к которому принадлежит файл. А отсюда следует вот что — одному i-узлу может соответствовать неограниченное количество имен файла (ссылок). При этом ссылки (дополнительные имена) могут находиться как в одном каталоге с исходным файлом, так и в любом другом каталоге файловой системы.

Как мы уже знаем, в Linux есть обычные файлы и есть файлы устройств. В чем между ними разница? А разница проявляется на уровне i-узла: i-узел обычного файла указывает на блоки данных, а i-узел файла устройства указывает на адрес драйвера в ядре Linux.

На рис. 14.1 вы можете увидеть непрокомментированный блок — «Планировщики ввода/вывода». Всего существуют четыре планировщика:

- noop — реализует простую очередь по принципу FIFO (First In First Out). Планировщик noop не вносит изменений в порядок следования запросов к дискам, полагаясь на контроллеры самих дисков и RAID-контроллеры;
- cfq (Completely Fair Queuning) — делит пропускную способность между всеми процессами. Для синхронных запросов создается по одной очереди на процесс, в то время как асинхронные запросы объединяются в очереди по приоритетам;
- deadline — за основу берется время нахождения запроса в очереди. Поскольку большинство приложений блокируются на чтении, то этот планировщик по умолчанию отдает приоритет запросам на чтение;
- anticipatory — предполагает, что следующий запрос будет адресован блоку, идущему после текущего, только что обработанного. После выполнения операции чтения или записи планировщик делает некоторую задержку с тем, чтобы дать приложению время послать запрос на действие со следующим блоком. Если часто приходится работать с большими файлами, этот планировщик может повысить производительность (но при условии, что эти файлы последовательно размещены на диске).

Узнать, какой именно планировщик используется у вас, можно командой:

```
sudo dmesg | grep -I scheduler
```

По умолчанию в Astra Linux применяется планировщик deadline. Сменить планировщик можно так (вместо `noop` укажите здесь нужный вам):

```
#echo noop > /sys/block/sdb/queue/scheduler
```

Если вас устраивает альтернативный планировщик, тогда нужно изменить файл конфигурации GRUB2, чтобы использовать его на постоянной основе. Для этого в список опций `GRUB_CMDLINE_LINUX_DEFAULT` добавьте параметр `elevator=<`*имя*`>`, например:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash elevator=noop"
```

После этого обновите GRUB2 и перезагрузите компьютер.

14.2. Имена устройств

Пользователи Windows привыкли к тому, что файл — это именованная область данных на диске. Отчасти так оно и есть. Отчасти — потому что приведенное определение файла было верно для DOS (Disk Operating System) и до сих пор верно для Windows.

В Linux же понятие файла значительно шире. Сейчас Windows-пользователи будут очень удивлены: в Linux есть файлы устройств, позволяющие обращаться с устройством как с обычным файлом. Файлы устройств находятся в каталоге **/dev** (от devices). Да, через файл устройства мы можем обратиться к устройству! Если вы

когда-либо работали в DOS, то, наверное, помните, что нечто подобное было и там, — существовали зарезервированные имена файлов: PRN (принтер), CON (клавиатура при вводе, дисплей при выводе), LPT*n* (параллельный порт, *n* — номер порта), COM*n* (последовательный порт).

> ***ФАЙЛЫ УСТРОЙСТВ***
>
> Кому-то может показаться, что разработчики Linux «увели» идею специальных файлов у Microsoft, ведь Linux появилась в начале 90-х, а DOS — в начале 80-х годов прошлого века. На самом деле это не так. Наоборот, Microsoft позаимствовала идею файлов устройств из операционной системы UNIX, которая была создана еще до появления DOS. Однако сейчас не время говорить об истории развития операционных систем, поэтому лучше вернемся к файлам устройств.

Вот некоторые примеры файлов устройств:

- /dev/sd*x* — файл жесткого диска SATA/SCSI;
- /dev/sd*xN* — файл устройства раздела на жестком диске, *N* — это номер раздела;
- /dev/scd*N* — файл устройства CD/DVD-привода;
- /dev/mouse — файл устройства мыши;
- /dev/modem — файл устройства модема (на самом деле является ссылкой на файл устройства ttyS*n*);
- /dev/ttyS*n* — файл последовательного порта, *n* — номер порта (ttyS0 соответствует COM1, ttyS1 — COM2 и т. д.).

В свою очередь, файлы устройств бывают двух типов: блочные и символьные. Обмен информации с *блочными* устройствами, например с жестким диском, осуществляется блоками информации, а с *символьными* — отдельными символами. Пример символьного устройства — последовательный порт.

Нас в этой главе больше интересуют, конечно же, блочные устройства. Имя /dev/sd*x* (где *x* — буква, начиная с a) сейчас предназначается для SATA/SCSI-дисков. Старое имя /dev/hda, которое ранее использовалось для IDE/PATA-дисков, больше не применяется. Даже если у вас каким-то образом еще остался старый IDE-диск, то в современных дистрибутивах ему также будет присвоено имя /dev/sd*x* (если у вас один IDE-диск, он будет называться /dev/sda).

Итак, с именами /dev/hda и /dev/sda разобрались. В современных системах вы также можете встретить имена устройств вида /dev/nvmeXnY. Так, /dev/nvme0n1 — означает первый жесткий диск с интерфейсом NVMe (NVM Express). Как правило, на современных компьютерах SSD-накопители подключаются не через SATA, а — для более быстрого доступа — по интерфейсу NVME. Обратите внимание, что первый диск обозначается n1, а не a, как в случае с SATA-устройствами:

- /dev/sda — первый SATA/PATA-накопитель;
- /dev/nvme0n1 — первый SSD-накопитель, подключающийся по NVMe.

Именами /dev/vd*X* обозначают диски в виртуальной машине KVM, а именами /dev/xvd*X* — в виртуальной машине Xen. Как обычно, заглавная буква *X* обозначает порядковый номер диска: a, b, c и т. д.

14.3. Идентификаторы UUID

Файл /etc/fstab (подробнее мы его рассмотрим в *главе 16*) содержит список автоматически монтируемых файловых систем. Откройте его. Наверняка вы заметите, что вместо устройств вида /dev/sd* в нем используются совершенно другие идентификаторы (рис. 14.3) — вида 3422b448-2460-4fd2-9183-8000de6f8343.

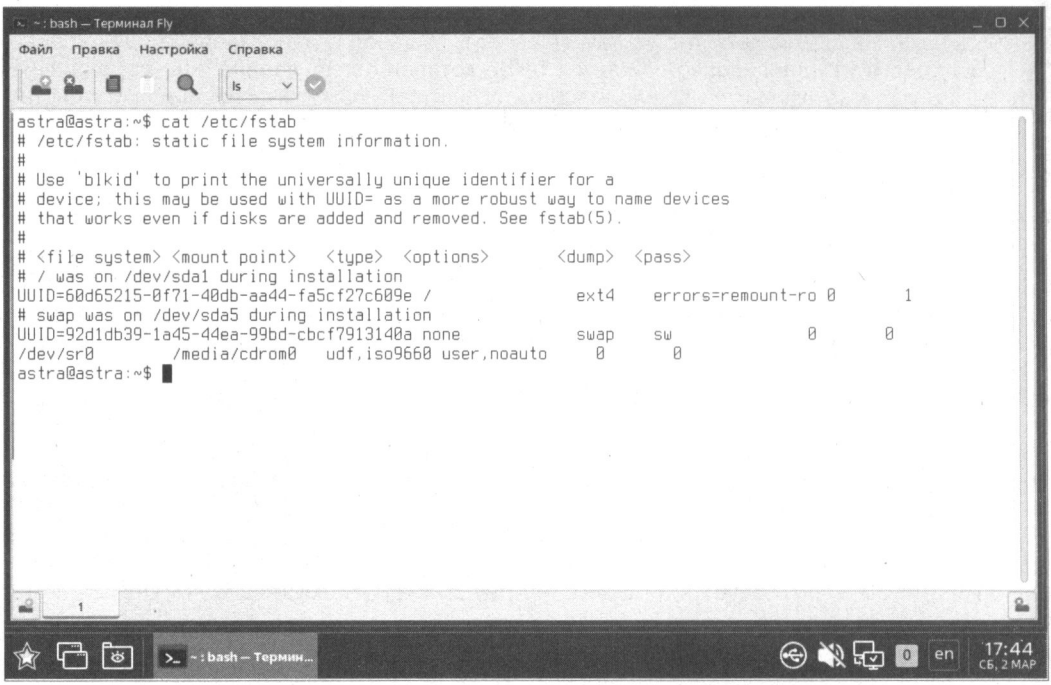

Рис. 14.3. Файл /etc/fstab

Такие идентификаторы называются UUID (Universally Unique Identifier) и представляют собой *длинные имена* дисков. Соответствие между обычными и длинными именами (рис. 14.4) можно просмотреть с помощью команды:

```
ls -l /dev/disk/by-uuid/
```

Спрашивается, зачем было вводить длинные имена, если короткие имена удобнее, во всяком случае для пользователей? Оказывается, разработчики Linux в первую очередь как раз о пользователях и позаботились. Возьмем обычный IDE-диск. Как известно, его можно подключить либо к первичному (primary), либо ко вторичному (secondary), если он есть, контроллеру. При этом, согласно положению перемычки выбора режима, винчестер может быть либо главным устройством (master), либо подчиненным (slave). Таким образом, в зависимости от контроллера, к которому подключается диск, изменяется его короткое имя: sda (primary master), sdb (primary slave), sdc (secondary master), sdd (secondary slave). То же самое происходит и с SATA/SCSI-винчестерами — при изменении параметров подключения изменяется и короткое имя устройства.

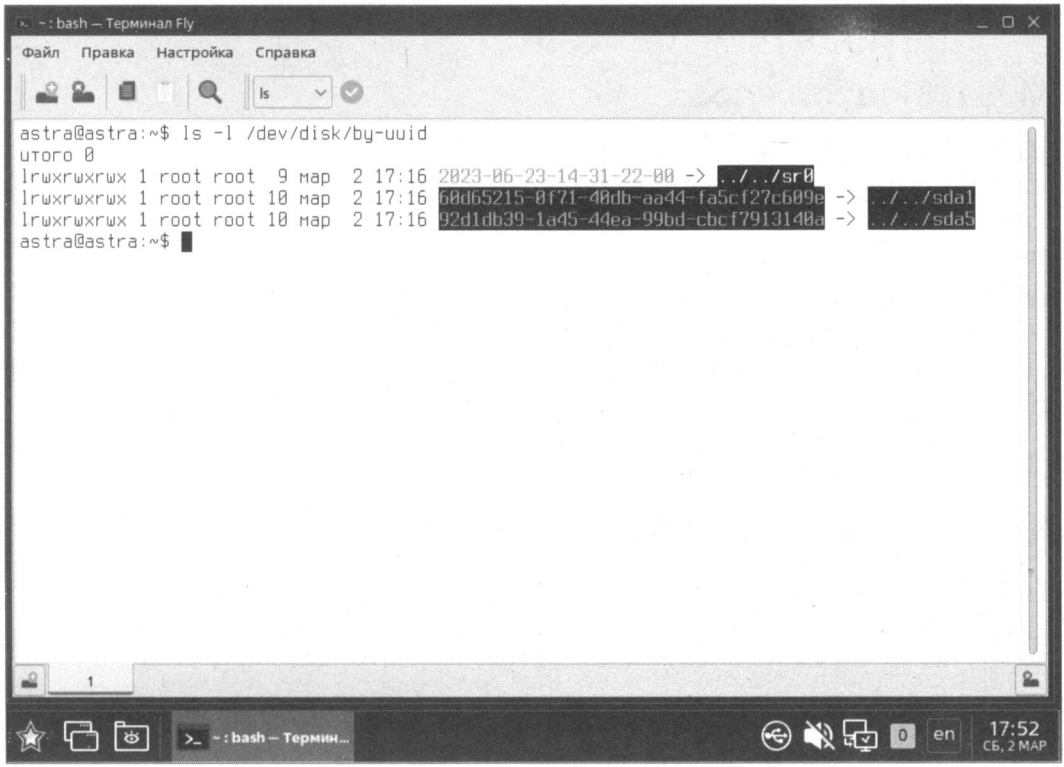

Рис. 14.4. Соответствие между обычными именами устройств и UUID

Получается, что если подключить один и тот же жесткий диск немного иначе, то его разделы, которые назывались, скажем, /dev/sda*N*, стали бы называться /dev/sdb*N*. Понятно, что загрузить Linux с такого диска не получится, поскольку везде указаны другие имена устройств.

При использовании же длинных имен идентификатор дискового устройства остается постоянным вне зависимости от типа подключения устройства к контроллеру. Именно поэтому длинные имена дисков часто также называются *постоянными* именами (persistent name). Таким образом, когда используются длинные имена дисков, система загрузится в любом случае, как бы вы ни подключили жесткий диск. Удобно? Конечно.

Но это еще не все. Постоянные имена — это только первая причина. Вторая причина заключается в обновлении библиотеки libata. В новой[1] версии libata все PATA-устройства, как мы уже отмечали ранее, именуются не как hd*x*, а как sd*x*, что вносит некую путаницу. Длинные же имена дисков от этого не изменяются и избавляют пользователя от беспокойства по поводу того, что его старый IDE-диск вдруг превратился в диск SATA/SCSI.

[1] Новой ее можно назвать весьма условно. С момента того обновления, когда IDE-диски стали называться sdX, прошло уже более 10 лет.

14.4. Поддерживаемые файловые системы

14.4.1. Модули ядра

Linux поддерживает много различных файловых систем. Начинающий пользователь просто теряется, когда видит такое многообразие выбора, — ведь в качестве корневой файловой системы доступны: ext2, ext3, ext4, XFS, ReiserFS, Btrfs, JFS и еще несколько.

Чтобы операционная система могла работать с файловой системой, нужно установить ее драйвер. Драйверы в Linux создаются в виде модулей ядра. Следующая команда выводит список модулей ядра файловых систем, установленных в вашей системе (рис. 14.5):

```
# ls /lib/modules/$(uname -r)/kernel/fs
```

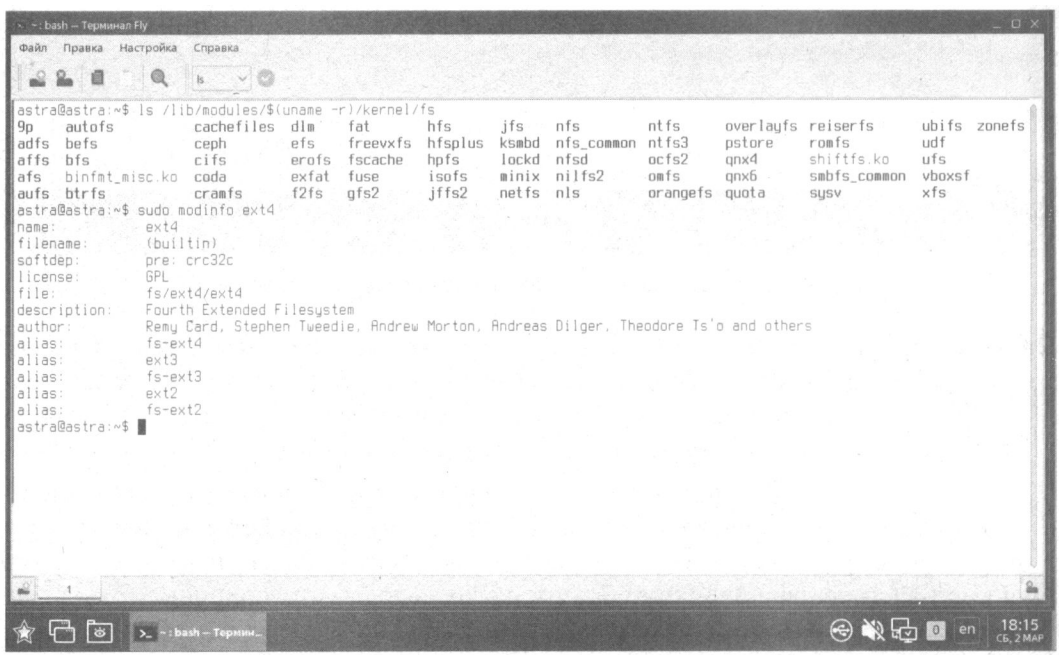

Рис. 14.5. Список поддерживаемых файловых систем

Просмотреть информацию о любом модуле (см. рис. 14.5) можно, выполнив команду:

```
sudo modinfo имя_модуля
```

Файл /proc/filesystems содержит список драйверов файловых систем, загруженных в текущий момент (рис. 14.6). Просмотреть его можно, выполнив команду:

```
cat /proc/filesystems
```

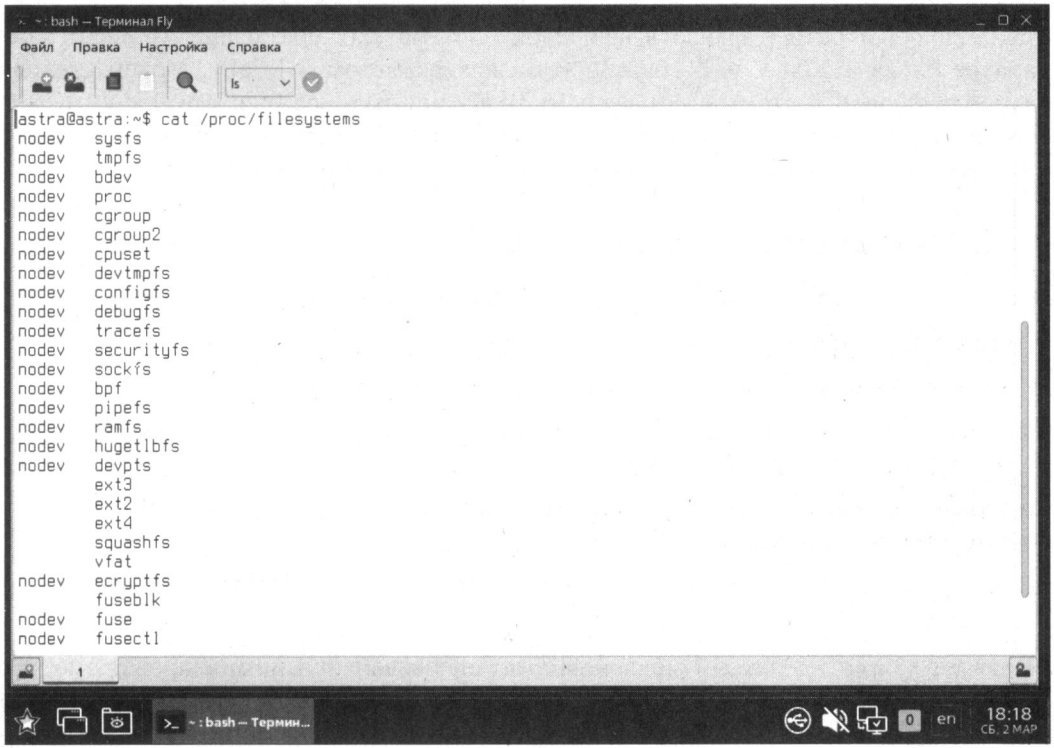

Рис. 14.6. Файл /proc/filesystems

14.4.2. Файловая система ext2

«Родной» файловой системой современных дистрибутивов Linux является журналируемая файловая система ext4, но при желании вы можете создать файловую систему другого типа.

Основное отличие систем ext3/ext4 от ext2 как раз и заключается в их журналируемости. Файловые системы ext3 и ext4 ведут журналы своей работы, что позволяет восстановить данные в случае сбоя. Осуществляется это следующим образом: перед тем как выполнить операцию, журналируемая файловая система записывает ее в особый файл — журнал, а после выполнения операции удаляет запись из журнала. Представим, что после занесения операции в журнал произошел сбой (например, отключилось электропитание). Позже, когда сбой будет устранен, файловая система по журналу выполнит все действия, которые в него занесены. Конечно, и это не всегда позволяет уберечься от последствий сбоя — стопроцентной гарантии никто не дает, но все же такая схема работы лучше, чем вообще ничего.

Иногда для повышения производительности выбирают файловую систему ext2, отказавшись от журналируемости. Действительно, так можно немного повысить производительность, если журналирование будет выключено. Впрочем, для ext3/4 есть возможность изменить режим журнала, что тоже может повысить производительность.

Максимальный размер файла в файловой системе ext2 составляет 16 Гбайт при размере блока 1 Кбайт и 1 Тбайт при блоках размером 4 Кбайт. Максимальный размер файловой системы — от 4 до 16 Тбайт при размере блока от 1 до 4 Кбайт соответственно. Сравните показатели этой файловой системы (по сути, древней) с той же FAT32, где файл размером больше 4 Гбайт создать невозможно.

14.4.3. Файловая система ext3

Основные отличия файловой системы ext3 от ext2 заключаются в следующем:

- наличие журнала;
- возможно увеличение размера файловой системы на лету;
- использование сбалансированного дерева для индексирования больших каталогов, что обеспечивает быстрый поиск файлов.

Файловая система ext3 совместима с ext2 и может быть преобразована в ext2 (и наоборот) при необходимости.

Журналируемая файловая система имеет три режима работы: `journal`, `ordered` и `writeback`:

- режим `journal` — самый медленный, но он позволяет минимизировать потери ваших данных в случае сбоя системы или отключения питания. В этом режиме в системный журнал записывается все, что только можно, и это позволяет максимально восстановить файловую систему в случае сбоя;
- в последовательном режиме (`ordered`) в журнал заносится информация только об изменении метаданных (служебных данных файловой системы). Этот режим используется по умолчанию и является компромиссным вариантом между производительностью и отказоустойчивостью;
- самым быстрым является режим обратной записи (`writeback`). Но использовать его я вам не рекомендую, поскольку особого толку от него не будет. Проще тогда уже при установке Linux выбрать файловую систему ext2 вместо ext3/ext4.

Если отказоустойчивость для вас на первом месте — выбирайте режим `journal`, во всех остальных случаях лучше выбрать `ordered`. Выбор режима осуществляется редактированием файла /etc/fstab. Например:

```
# режим ordered используется по умолчанию,
# поэтому ничего указывать не нужно
/dev/sda1 / ext3 defaults 1 0
# на этом разделе важные данные, используем режим journal
/dev/sda2 /var ext3 data=journal 1 0
# здесь ничего важного нет, режим writeback
/dev/sda2 /opt ext3 data=writeback 0 0
```

После изменения этого файла выполните команду:

```
# mount -a
```

Она заново смонтирует все файловые системы, чтобы изменения вступили в силу.

14.4.4. Файловая система ext4

Файловая система ext4 заслуживает отдельного разговора. Все, что было сказано о файловых системах ранее, справедливо и для ext4, но у этой файловой системы есть ряд особенностей, о которых мы сейчас и поговорим.

Поддержка ext4 как стабильной файловой системы появилась в ядре Linux версии 2.6.28. Если сравнивать эту файловую систему с ext3, то производительность и надежность в ext4 существенно увеличена, а максимальный размер раздела доведен до 1024 петабайт (1 эксбибайт). Максимальный размер файла — более 2 Тбайт. Ресурс Phoronix (**www.phoronix.com**) произвел тестирование файловой системы ext4 на SSD-накопителе (такие накопители устанавливаются на современные ноутбуки) — результат, как говорится, налицо: ext4 почти в два раза превзошла по производительности файловые системы ext3, XFS, JFS и ReiserFS.

Сравнение ext3 и ext4

Описание особенностей файловой системы ext4 и ее преимуществ по сравнению с ext3 сведены в табл. 14.1.

***Таблица 14.1.** Особенности ext4*

Особенность	Комментарий
Увеличенный размер файла и файловой системы	Для ext3 максимальный размер файловой системы составляет 32 Тбайт, а файла — 2 Тбайт, но на практике ограничения были более жесткими. Так, в зависимости от архитектуры, максимальный размер тома составлял до 2 Тбайт, а максимальный размер файла — до 16 Гбайт. В случае с ext4 максимальный размер тома составляет 1 эксбибайт (EiB) — это 2^{60} байт. Максимальный размер файла — 16 Тбайт. Такие объемы информации пока не нужны обычным пользователям, однако весьма пригодятся на серверах, работающих с большими дисковыми массивами
Экстенты	Основной недостаток ext3 — ее метод выделения места на диске. Дисковые ресурсы выделялись с помощью битовых карт свободного места, а такой способ не отличается ни скоростью, ни масштабируемостью. Получилось, что ext3 более эффективна для небольших файлов, но совсем не подходит для хранения больших. Для улучшения выделения ресурсов и более эффективной организации данных в ext4 были введены *экстенты*. Экстент — это способ представления непрерывной последовательности блоков памяти. Для эффективного представления маленьких файлов в экстентах применяется уровневый подход, а для больших файлов используются деревья экстентов. Например, один индексный дескриптор может ссылаться на четыре экстента, каждый из которых может ссылаться на другие индексные дескрипторы и т. д. Такая структура является мощным механизмом представления больших файлов, а также более защищена и устойчива к сбоям
Отложенное выделение пространства	Файловая система ext4 может отложить выделение дискового пространства до последнего момента, что увеличивает производительность системы

Таблица 14.1 (окончание)

Особенность	Комментарий
Контрольные суммы журналов	Контрольные суммы журналов повышают надежность файловой системы
Большее количество каталогов	В ext3 могло быть максимум 32 000 каталогов, в ext4 количество каталогов не ограничивается
Дефрагментация на лету	Файловая система ext3 не особо склонна к фрагментации, но все же такое неприятное явление имеется. В ext4 производится дефрагментация на лету, что позволяет повысить производительность системы в целом
Наносекундные временные метки	В большинстве файловых систем временные метки (timestamp) устанавливаются с точностью до секунды, в ext4 точность повышена до наносекунды. Также ext4 поддерживает временные метки до 25 апреля 2514 года, в отличие от ext3 (только до 18 января 2038 г.)

Совместимость с ext3

Файловая система ext4 является прямо и обратно совместимой с ext3, однако все же существуют некоторые ограничения. Предположим, что у нас на диске имеется файловая система ext4. Ее можно смонтировать и как ext3, и как ext4 (это и есть прямая совместимость) — и тут ограничений никаких нет. А вот с обратной совместимостью не все так безоблачно — если файловую систему ext4 смонтировать как ext3, то она будет работать без экстентов, что снизит ее производительность.

Переход на ext4

Перейти на ext4 можно без потери данных и в любой удобный для вас момент. Откройте терминал и введите команду:

```
sudo tune2fs -O extents,uninit_bg,dir_index /dev/имя_устройства
```

На момент ввода этой команды устройство должно быть размонтировано.

> ***Преобразование корневой файловой системы***
>
> Если нужно преобразовать в ext4 корневую файловую систему, то эту команду нужно вводить с LiveCD, поддерживающего ext4.

Теперь проверим файловую систему:

```
sudo fsck -pf /dev/имя_устройства
```

Затем смонтируем файловую систему так:

```
mount -t ext4 /dev/имя_устройства /точка_монтирования
mount -t ext4 /dev/disk/by-uuid/UUID-устройства /точка_монтирования
```

Если раздел автоматически монтируется через /etc/fstab, не забудьте исправить файловую систему на ext4:

```
UUID=UUID-раздела    /точка   ext4  defaults,errors=remount-ro,relatime  0  1
```

Если вы изменили тип файловой системы корневого раздела, тогда необходимо отредактировать файл конфигурации загрузчика GRUB2 (см. *главу 4*) и изменить тип файловой системы в конфигурации ядра.

14.4.5. Другие файловые системы

В качестве корневой файловой системы и файловой системы других Linux-разделов могут служить файловые системы ext3 и ext4, а также ReiserFS, XFS, JFS и др. Рассмотрим особенности этих файловых систем, чтобы понять, использовать ли их или же остановить свой выбор на стандартной ext4.

- **Файловая система ReiserFS** (она же **Reiser3**) считается самой экономной, поскольку позволяет хранить несколько файлов в одном блоке (другие файловые системы могут хранить в одном блоке только один файл или одну его часть). Например, если размер блока равен 4 Кбайт, а файл занимает всего 512 байтов (а таких файлов в разных каталогах Linux очень много), то 3,5 Кбайт в этом блоке просто не будут использоваться. А вот ReiserFS позволяет задействовать буквально каждый байт вашего жесткого диска!

  Но у этой файловой системы есть два больших недостатка: она неустойчива к сбоям, и ее производительность сильно снижается при фрагментации диска. Поэтому, если вы выбираете ReiserFS, покупайте источник бесперебойного питания и почаще дефрагментируйте жесткий диск.

- **Файловая система Reiser4** впервые была представлена в 2004 году. Она поддерживает транзакции, задержку выделения пространства, а также сжатие и шифрование данных.

- **Файловая система XFS** была разработана компанией Silicon Graphics в 2001 году. Основная ее особенность — высокая производительность (до 7 Гбайт/с). Кроме того, XFS может работать с блоками размером от 512 байтов до 64 Кбайт. Ясно, что если у вас много небольших файлов, то в целях экономии дискового пространства можно установить самый маленький размер блока. А если вы работаете с файлами большого размера (например, с мультимедиа), выбирайте самые большие блоки, — тогда файловая система обеспечит максимальную производительность (конечно, если «железо» позволяет). Учитывая такие особенности этой файловой системы, ее нет смысла устанавливать на домашнем компьютере, предназначенном для выхода в Интернет и просмотра любительских фотографий, поскольку вы просто не сможете оценить все ее преимущества. А вот если вы реально работаете с файлами очень большого размера, XFS проявит себя с лучшей стороны.

- **Файловая система ZFS (Zettabyte File System)** создана в 2005 году компанией Sun Microsystems для операционной системы Solaris. Отличительные особенности ZFS: отсутствие фрагментации, создание снапшотов диска, которые можно использовать для восстановления данных, организация пулов хранения (storage pools), изменяемый размер блоков, 64-разрядный механизм контрольных сумм.

- **Файловая система Btrfs (B-tree FS** или **Butter FS)** изначально была представлена компанией Oracle. Многие считают эту файловую систему ответом Oracle на файловую систему ZFS.
- **Файловая система JFS** (разработка IBM) сначала появилась в операционной системе AIX, а потом была модифицирована под Linux. Основные достоинства этой файловой системы — надежность и высокая производительность (выше, чем у XFS). Однако у нее маленький размер блока (от 512 байтов до 4 Кбайт) — следовательно, она хороша на сервере баз данных, но не при работе с данными мультимедиа, поскольку блока в 4 Кбайт для обработки, например, видео в реальном времени будет маловато.
- **Файловая система Tux2** была создана Дэниэлем Филипсом как надстройка над ext2, но не получила публичного распространения.
- **Файловая система Tux3** задумывалась как надстройка над Btrfs. Эта файловая система, вместо журналирования, предлагает версионное восстановление файлов: для каждого файла создается измененная копия, а не переписывается текущая версия, что позволяет гибко управлять версиями.
- **Файловая система Xiafs** основана на файловой системе MINIX. Это весьма древняя разработка, она создавалась параллельно с ext2 на замену файловой системе ext и не получила распространения.

Узнать тип файловой системы того или иного раздела/устройства можно с помощью команды `file`, например:

```
sudo file -s /dev/sda1
```

Вывод этой команды будет примерно таким:

```
/dev/sda1: Linux rev 1.0 ext4 filesystem data, UUID=3762b167-9de0-4124-b7c1-
46d4a2fc2019 (extends) (64bit) (large files) (huge files)
```

Как видно из приведенного вывода, устройство `/dev/sda1` использует файловую систему `ext4`.

Команда `df -T` пригодится, если вы не знаете точно, как называется ваше устройство, или вам лень вводить его имя:

```
root@hosting:/srv/www/htdocs# df -T
Файл.система Тип      1K-blocks  Использовано Доступно  Использовано,% Смонтировано в
/dev/sda1    ext4     30829600   21994740     7434548       75%       /
udev         devtmpfs 16460416          4    16460412       1%        /dev
tmpfs        tmpfs     3294256       3088     3291168       1%        /run
none         tmpfs        5120          0        5120       0%        /run/lock
none         tmpfs    16471276          0    16471276       0%        /run/shm
none         tmpfs      102400          0      102400       0%        /run/user
/dev/sdb     ext4     41153856   35731076     3309244       92%       /srv
/dev/sdc1    ext4     82438800   62819492    15408624       81%       /var
/dev/sdd2    reiserfs 45087324    6462812    38624512       15%       /media/reiser-hdd
```

Эта команда выводит информацию о смонтированных файловых системах, в том числе указывает тип каждой файловой системы. Показанные в выводе команды

файловые системы `devtmpfs` и `tmpfs` — не совсем файловые системы в привычном понимании этого термина. Они представляют собой временные хранилища, размещаемые в оперативной памяти, а не на физическом накопителе, т. е. это так называемые *RAM-диски* (диски в оперативной памяти).

С точки зрения *производительности* рассматриваемых файловых систем напрашиваются следующие рекомендации:

- для рабочей станции и сервера общего назначения оптимальной файловой системой являются ext3/ext4 или ReiserFS (в крайнем случае);
- на сервере баз данных можно использовать JFS — в этом случае (особенно, если база данных огромная) будет наблюдаться определенный прирост производительности;
- Btrfs должна неплохо подойти для современных серверов — она оптимизирована под SSD-диски, поддерживает снапшоты и даже поддерживает сжатие. Вот только последняя ее характеристика заставляет задуматься о производительности;
- файловая система XFS — это удел станции мультимедиа, на обычной рабочей станции или обычном сервере ее использовать не следует.

Но производительность — это не единственный критерий выбора файловой системы, особенно для сервера. Да, производительность учитывать нужно, но, кроме того, нельзя пренебрегать и следующими факторами:

- *надежностью* — все-таки мы выбираем файловую систему для сервера, а не для домашнего компьютера;
- *наличием программ для восстановления файловой системы в случае сбоя* — сбой может произойти даже в случае использования самой надежной файловой системы, поэтому наличие программного комплекса для восстановления файловой системы будет не лишним;
- *максимальным размером файла* — сервер обрабатывает огромные объемы информации, поэтому этот критерий для нас также важен.

Файловые системы ext3/ext4, ReiserFS и XFS одинаково надежны, а вот надежность JFS иногда оставляет желать лучшего. Учитывая это, а также и то, что программы для восстановления файловой системы имеются только в системах ext*, на сервере лучше использовать все-таки ext3/ext4.

ГЛАВА 15

ASTRA LINUX®

Работа с файлами и каталогами

- Имена файлов в Linux
- Команды для работы с файлами и каталогами
- Использование ссылок. Команда `ln`
- Графический файловый менеджер
- Сжатие и распаковка файлов и каталогов

15.1. Имена файлов в Linux

В Linux, по сравнению с Windows, несколько иные правила построения имен файлов, и вам придется с этим смириться. Начнем с того, что в Linux нет такого понятия, как *расширение имени файла*. В Windows, например, для файла Document1.doc именем файла является фрагмент Document1, а doc — это его расширение. В Linux же Document1.doc — это имя файла целиком, никакого разделения на имя и расширение нет.

Максимальная длина имени файла — 254 символа. Имя может содержать любые символы (в том числе и кириллицу), кроме / \ ? < > * " |. Тем не менее кириллицу в именах файлов я бы не рекомендовал использовать вовсе. Впрочем, если вы уверены, что не будете эти файлы передавать Windows-пользователям (на флешке, по электронной почте или еще как-то через Интернет) — используйте на здоровье. А при обмене файлами с Windows-пользователями из-за возможных несовпадений кодировок вместо русскоязычного имени файла адресат может увидеть абракадабру... Так что имена файлов во всех случаях лучше писать латиницей.

Придется вам привыкнуть и к тому, что Linux чувствительна к регистру в имени файла: FILE.txt и FiLe.Txt — это два разных файла.

Разделение элементов пути осуществляется символом / (прямой слеш), а не \ (обратный слеш), как в Windows.

15.2. Команды для работы с файлами и каталогами

15.2.1. Работа с файлами

Здесь мы рассмотрим основные команды для работы с файлами в Linux (табл. 15.1), а в последующих разделах этой главы — команды для работы с каталогами, ссылками и поговорим о правах доступа к файлам и каталогам.

Таблица 15.1. *Основные команды Linux, предназначенные для работы с файлами*

Команда	Назначение
`touch <файл>`	Создает пустой файл
`cat <файл>`	Просмотр текстового файла
`tac <файл>`	Вывод содержимого текстового файла в обратном порядке, т. е. сначала выводится последняя строка, потом предпоследняя и т. д.
`cp <файл1> <файл2>`	Копирует файл `<файл1>` в файл `<файл2>`. Если `<файл2>` существует, программа попросит разрешение на его перезапись
`mv <файл1> <файл2>`	Перемещает файл `<файл1>` в файл `<файл2>`. Эту же команду можно использовать и для переименования файла
`rm <файл>`	Удаляет файл
`locate <файл>`	Производит быстрый поиск файла
`which <программа>`	Выводит каталог, в котором находится программа, если она вообще установлена. Поиск производится в каталогах, указанных в переменной окружения `PATH` (это путь поиска программ)
`less <файл>`	Используется для удобного просмотра файла с возможностью скроллинга (постраничной прокрутки)

Еще раз о консоли...

Все представленные здесь команды предназначены для работы в консоли, т. е. в текстовом режиме. Понятно, что большинство современных дистрибутивов запускаются в графическом режиме, поэтому некоторые пользователи Linux даже не подозревают о том, что существует консоль. Да, таково новое поколение Linux-пользователей, которым проще использовать графический файловый менеджер, чем вводить команды. Но если вы хотите стать квалифицированным пользователем Linux, то просто обязаны знать, как работать в консоли, иначе уподобитесь Windows-пользователям, которые при каждом сбое переустанавливают операционную систему... Если вы пропустили *главу 5*, в которой рассматривается работа с консолью, настоятельно рекомендую вернуться и прочитать ее!

Рассмотрим небольшую серию команд:

```
touch file.txt
echo "some text" > file.txt
cat file.txt
cp file.txt file-copy.txt
cat file-copy.txt
```

```
rm file.txt
cat file.txt
mv file-copy.txt file.txt
cat file.txt
```

Первая команда (touch) создает в текущем каталоге файл file.txt. Вторая команда (echo) записывает строку some text в этот же файл. Обратите внимание на символ > — это символ перенаправления ввода/вывода, о котором мы поговорим чуть позже.

Третья команда (cat) выводит содержимое файла — в файле записанная нами строка some text. Четвертая команда (cp) копирует файл file.txt в файл с именем file-copy.txt. После этого мы опять используем команду cat, чтобы вывести содержимое файла file-copy.txt — надо же убедиться, что файл действительно скопировался.

Шестая команда (rm) удаляет файл file.txt. При удалении система спрашивает, хотите ли вы удалить файл. Если хотите удалить, то нужно нажать клавишу <Y>, а если нет, то клавишу <N>. Точно ли файл удален? Убедимся в этом: введите команду cat file.txt. Система нам сообщает, что нет такого файла.

Восьмая команда (mv) переименовывает файл file-copy.txt в файл file.txt. Последняя команда выводит новый файл file.txt. Думаю, особых проблем с этими командами у вас не возникло, тем более что принцип действия этих команд вам должен быть знаком по командам DOS, которые как квалифицированный пользователь Windows вы должны знать наизусть.

Вместо имени файла иногда очень удобно указать *маску имени файла*. Например, у нас есть много временных файлов, имена которых заканчиваются фрагментом tmp. Для их удаления нужно воспользоваться командой: rm *tmp.

Если же требуется удалить все файлы в текущем каталоге, можно просто указать звездочку: rm *.

Аналогично можно использовать символ ?, который, в отличие от звездочки, заменяющей последовательность символов произвольной длины, заменяет всего один символ. Например, нам нужно удалить все файлы, имена которых состоят из трех букв и начинаются на s:

```
rm s??
```

Будут удалены файлы s14, sqm, sr6 и т. д., но не будут затронуты файлы, имена которых состоят более чем из трех букв и которые не начинаются на s.

Маски имен можно также использовать и при работе с каталогами.

15.2.2. Работа с каталогами

Основные команды для работы с каталогами приведены в табл. 15.2.

При указании имени каталога можно использовать следующие символы:

- . — означает текущий каталог. Если вы введете команду cat ./file, то она выведет файл file, который находится в текущем каталоге;

Таблица 15.2. Основные команды для работы с каталогами

Команда	Описание
mkdir <каталог>	Создание каталога
cd <каталог>	Изменение каталога
ls <каталог>	Вывод содержимого каталога
rmdir <каталог>	Удаление пустого каталога
rm -r <каталог>	Рекурсивное удаление каталога

- .. — родительский каталог. Например, команда cd .. переведет вас на один уровень вверх по дереву файловой системы;
- ~ — домашний каталог пользователя (об этом мы поговорим позже).

Теперь рассмотрим пример работы с каталогами на практике. Выполните следующие команды:

```
mkdir directory
cd directory
touch file1.txt
touch file2.txt
ls
cd ..
ls directory
rm directory
rmdir directory
rm -r directory
```

Первая команда (mkdir) создает каталог directory в текущем каталоге. Вторая команда (cd) переводит (изменяет каталог) в только что созданный каталог. Следующие две команды touch создают в новом каталоге два файла: file1.txt и file2.txt.

Команда ls без указания каталога выводит содержимое текущего каталога. Команда cd .. переводит в родительский каталог. Как уже было отмечено, в Linux родительский каталог обозначается так: .. (две точки), а текущий так: . (одна точка). То есть, находясь в каталоге directory, мы можем обращаться к файлам file1.txt и file2.txt без указания каталога или же так: ./file1.txt и ./file2.txt.

> ***Прямой слеш!***
>
> Еще раз обратите внимание — в Linux, в отличие от Windows, для разделения элементов пути служит прямой слеш (/), а не обратный (\).

Кроме обозначений .. и . в Linux часто используется символ «тильда» (~) — так обозначается *домашний каталог*. Предположим, что наш домашний каталог /home/astra. В нем мы создали подкаталог dir и поместили в него файл file1.txt. Полный путь к файлу можно записать так:

```
/home/astra/dir/file1.txt
```

или же так:

```
~/dir/file1.txt
```

Как видите, тильда (~) заменяет здесь часть пути. Удобно? Конечно!

Поскольку мы находимся в родительском для каталога `directory` каталоге, чтобы вывести содержимое только что созданного каталога, в команде `ls` нам нужно четко указать имя каталога:

```
ls directory
```

Команда `rm` служит для удаления каталога. Но что мы видим — система отказывается удалять каталог! Пробуем удалить его командой `rmdir`, но и тут отказ. Система сообщает нам, что каталог не пустой, т. е. содержит файлы. Для удаления каталога нужно сначала удалить все имеющиеся в нем файлы. Конечно, делать это не сильно хочется, поэтому проще указать опцию `-r` команды `rm` для рекурсивного удаления каталога. В этом случае сначала будут удалены все подкаталоги (и все файлы в этих подкаталогах), а затем будет удален сам каталог.

Команды `cp` и `mv` работают аналогично: для копирования (перемещения/переименования) сначала указывается каталог-источник, а потом каталог-назначение. Для каталогов желательно указывать параметр `-r`, чтобы копирование (перемещение) производилось рекурсивно.

15.3. Использование ссылок. Команда *ln*

15.3.1. Жесткие и мягкие ссылки

Файлы и каталоги физически хранятся на носителе в виде набора блоков. Информация о файле (владелец, права доступа, размер файла, время последнего обращения, признак каталога и т. д.) хранится в inode — индексном дескрипторе.

Номер inode также называют *порядковым номером файла*. Этот номер является уникальным в пределах отдельной файловой системы. Запись каталога содержит имя файла (или каталога), а также указатель на дескриптор inode, в котором хранится информация об этом файле (каталоге).

Ссылки — это дополнительные записи каталога, позволяющие обращаться к файлам или каталогам по нескольким именам. *Жесткая* ссылка — это запись каталога, указывающая на дескриптор inode, а *мягкая* (или *символическая*) ссылка — это запись каталога, указывающая на имя объекта с другим inode.

Механизмы хранения дополнительных имен (ссылок) зависят от типа файловой системы и длины имени.

Жесткие ссылки можно создать только для файлов, для каталогов их создать невозможно. Исключение составляют лишь специальные записи каталогов, указывающие на сам каталог и на ее родительский каталог, т. е. . и .. являются жесткими ссылками. При попытке создать ссылку для каталога вы увидите следующее сообщение об ошибке:

```
ln: 'link': hard link not allowed for directory
ln: 'link': не допускается создавать жесткие ссылки на каталоги
```

Жесткие ссылки можно использовать только в пределах одной файловой системы, поскольку они являются указателями на дескрипторы inode, которые, как уже отмечалось, являются уникальными только в пределах отдельной файловой системы.

Файл удаляется только тогда, когда удаляется последняя ссылка на его inode и счетчик ссылок сбрасывается до 0. Об удалении ссылок мы поговорим позже, т. к. этот вопрос заслуживает отдельного рассмотрения.

Мягкая (или символическая ссылка, symlink) указывает на имя другого файла или каталога, а не на его inode. В этом и есть отличие мягких ссылок от жестких. Мягкие ссылки можно создавать на объекты разных файловых систем, а также на каталоги. Удаление мягкой ссылки не приводит к удалению файла или каталога, на которую она указывает, а удаление целевого объекта не приводит к автоматическому удалению мягких ссылок. Другими словами, если у вас есть файл file.txt и вы создали на него символическую ссылку symlink.txt, то в случае удаления файла file.txt ссылка окажется «битой» — она не будет ни на что указывать.

15.3.2. Создание ссылок

Для создания ссылок служит команда `ln`:

```
ln file.txt link1
ln -s file.txt link2
```

Первая команда создает жесткую ссылку `link1`, ссылающуюся на текстовый файл `file.txt`. Вторая команда создает символическую ссылку `link2`, которая ссылается на этот же текстовый файл `file.txt`.

Модифицируя ссылку (все равно какую: `link1` или `link2`), вы автоматически модифицируете исходный файл `file.txt`.

15.3.3. Определение ссылок

Мы только что научились создавать ссылки. Теперь давайте посмотрим, как различить — где файл, а где ссылка. Представим, что мы создали файл и две ссылки на него:

```
echo "Hello" > file
ln -s file symlink
ln file hardlink
```

Если ввести команду `ls`, то символическая ссылка будет выделена цветом. Каким именно — зависит от вашего дистрибутива. Так, в Ubuntu символические ссылки выделяются цветом ближе к бирюзовому. Если в вашем дистрибутиве символические ссылки не выделяются, попробуйте указать опцию `--color=auto`:

```
ls --color=auto
```

А вот жесткие ссылки никак не выделяются, и визуально нельзя понять: перед нами файл или ссылка — по крайней мере, в Ubuntu. Однако есть дистрибутивы, где жесткие ссылки выделяются каким-либо цветом, — например, темно-синим.

Рассмотрим вывод команды `ls -l`:

```
ls -l
итого 8
-rw-rw-r-- 2 astra astra 6 июл 15 08:46 file
-rw-rw-r-- 2 astra astra 6 июл 15 08:46 hardlink
lrwxrwxrwx 1 astra astra 4 июл 15 08:46 symlink -> file
```

Как можно видеть, проще всего с символическими ссылками — с помощью стрелки (->) сразу показывается, на какой файл указывает ссылка.

Также найти все символические ссылки можно с помощью команды `find`:

```
find . -type l
```

Найти с помощью этой команды все символические ссылки на файл `"file"` можно так:

```
find . -lname "file"
```

С жесткими ссылками все не так просто. Первая колонка вывода команды `ls -l` — это права доступа. Вторая колонка — счетчик жестких ссылок.

Посмотрим на вывод команды `ls -i`, которая показывает индексные дескрипторы файлов:

```
ls -i
2130783 file
2130783 hardlink
2130784 symlink
```

Как можно видеть, здесь есть два одинаковых дескриптора (`2130783`) и два имени для этого дескриптора.

15.3.4. Удаление файлов и жесткие ссылки

Мы подходим к самому интересному вопросу. Как уже отмечалось, файл не будет удален, пока на него указывает хоть одна жесткая ссылка.

В предыдущем разделе приводился вывод команды `ls` с опцией `-i`. Посмотрите также на вывод команды `ls -l` — вторая колонка показывает количество жестких ссылок на inode. По сути, в этом контексте нет особой разницы между ссылкой и файлом. Вы можете удалить файл `file`:

```
rm file
ls -l
итого 4
-rw-rw-r-- 1 astra astra 6 июл 15 08:46 hardlink
lrwxrwxrwx 1 astra astra 4 июл 15 08:46 symlink -> file
```

И вы увидите, что счетчик ссылок оказался уменьшен на единицу. Но ваши данные останутся в *файле* `hardlink`, и вы сможете их прочитать. Почему же файл `file` был удален, *если на него указывала жесткая ссылка* `hardlink`? Да потому, что жесткая ссылка *указывает не на имя файла, а на inode*! А ведь индексный дескриптор `2130783` остался, и он не будет удален, пока счетчик жестких ссылок больше 0!

Именно поэтому, если вы хотите защитить файл от удаления путем создания на него жесткой ссылки, — это неправильный вариант. Для защиты файла от случайного удаления используйте команду `chattr +i` (она также будет рассмотрена в этой главе позже):

```
touch f
sudo chattr +i f
rm f
rm: удалить защищенный от записи пустой обычный файл 'f'? y
rm: невозможно удалить 'f': Операция не позволена
```

Удалить файл можно, только сняв флаг `i`:

```
chattr -i f
```

Кстати, после удаления файла `file`, на который указывала символическая ссылка `symlink`, последняя превращается в «битую» ссылку — в выводе команды `ls` она отмечается красным цветом на черном фоне. Таким образом, символическая ссылка превращается в «битую» по следующим причинам:

- удаление целевого файла;
- переименование целевого файла;
- переименование элементов пути к целевому файлу. Например, ссылка указывала на файл /home/astra/links/file, а потом каталог links был переименован в test.

Возвращаемся к жестким ссылкам — найти все жесткие ссылки на файл можно командой `find` с опцией `-samefile`:

```
find . -samefile file
```

15.3.5. Разница между копированием и созданием жесткой ссылки

Учитывая, что жесткая ссылка — это практически то же самое, что и файл, возникает вопрос: когда лучше копировать файл, а когда создавать ссылки?

Все зависит от поставленных задач. Ведь жесткая ссылка ссылается на тот же inode, что и файл, следовательно, при изменении файла (или жесткой ссылки) изменяется и содержимое файла. Если же вы создаете копию файла, то ей будет присвоен другой inode и, следовательно, изменение копии никак не отразится на оригинале, и наоборот.

15.4. Графический файловый менеджер

15.4.1. Знакомство с программой

Графическая оболочка Astra Linux содержит удобный файловый менеджер, который вы будете использовать в повседневной работе (рис. 15.1).

> ***Старый, но проверенный***
>
> Приложение так и называется Файловый менеджер рабочего стола Fly. Самое интересное, что́ если зайти в раздел меню **Справка | О программе**, то, кроме названия, вы увидите и строку копирайта, а год там указан 2009-й. Видимо, в этой программе ничего не менялось уже 15 лет...

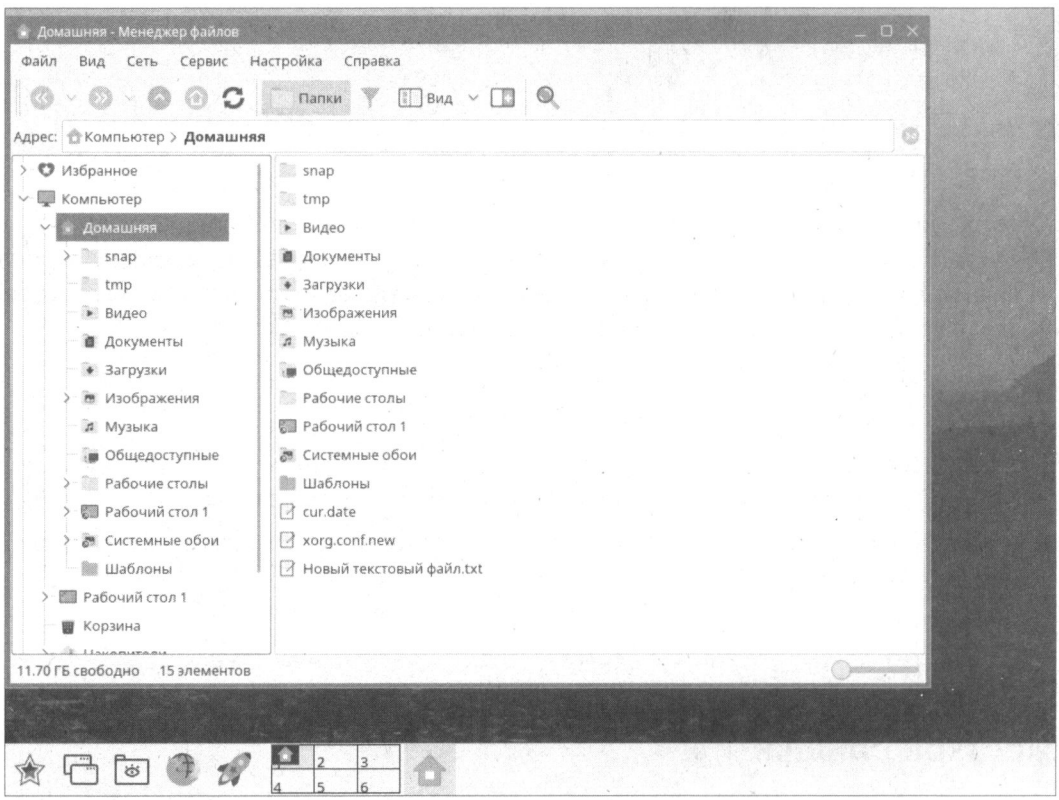

Рис. 15.1. Файловый менеджер в Astra Linux

Приложение построено по классической двухпанельной схеме: слева находится дерево каталогов, справа — содержимое выбранного на левой панели каталога. По умолчанию файловый менеджер отображает содержимое домашнего каталога пользователя.

Для просмотра подмонтированных накопителей разверните узел **Компьютер | Накопители**, а для просмотра корневой файловой системы — узел **Компьютер | Файловая система**, что и показано на рис. 15.2.

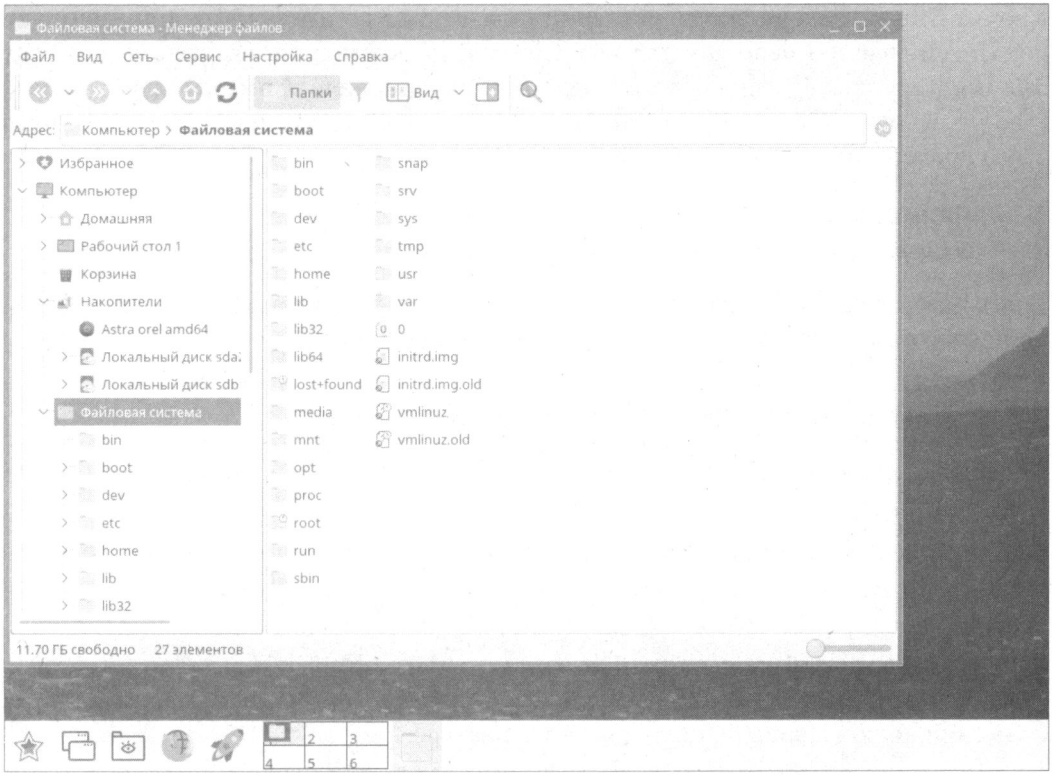

Рис. 15.2. Просмотр подключенных накопителей и файловой системы

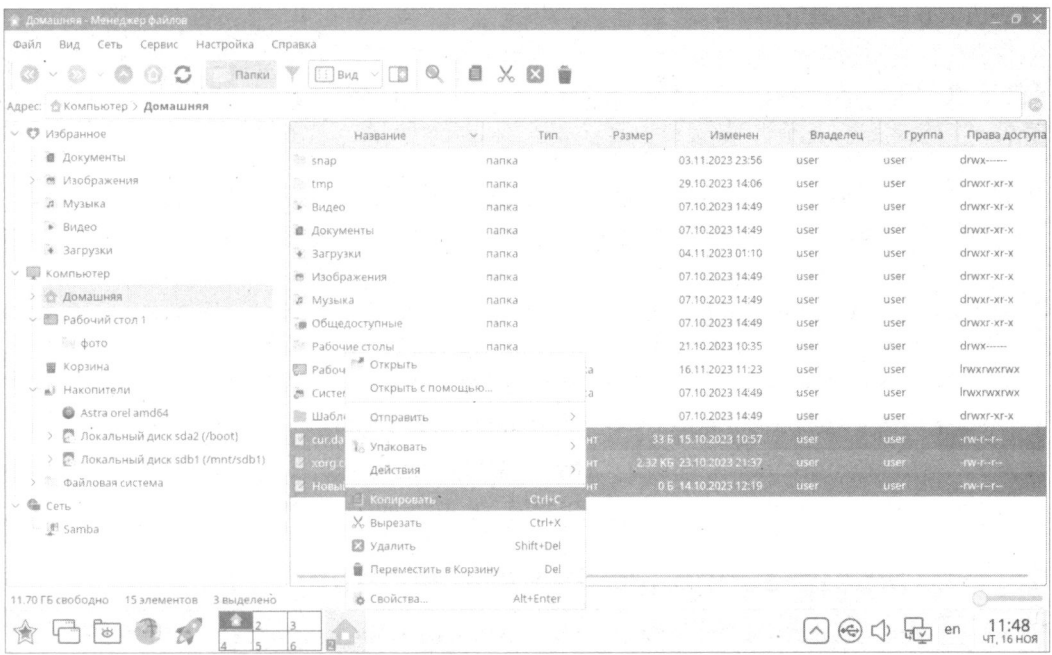

Рис. 15.3. Вид правой панели **Таблица** (открыто контекстное меню выделенной группы файлов)

Кнопка **Вид** на панели инструментов позволяет быстро изменять вид правой панели. По умолчанию используется вид **Список**, но вы можете выбрать два других: **Значки** (крупные значки вместо мелких, как при виде **Список**) или **Таблица** (рис. 15.3) — наиболее информативный вид, где вы можете просмотреть, помимо всего прочего, и права доступа к файлам и каталогам.

Примечание

О правах доступа будет подробно рассказано в *главе 18*.

Далее мы рассмотрим основные операции с файлами и каталогами — как их выполнить с помощью файлового менеджера по умолчанию.

15.4.2. Копирование файлов и каталогов

Последовательность действия, которые нужно выполнить для копирования файлов и каталогов:

1. С помощью мыши выделите файлы и каталоги, которые нужно скопировать. Для этого щелкайте мышью на нужных элементах при нажатой клавише <Ctrl> (или <Shift> — если нужно выделить сразу диапазон файлов).
2. Щелкните правой кнопкой мыши на выделенных файлах и выберите в открывшемся контекстном меню команду **Копировать** (см. рис. 15.3) или нажмите комбинацию клавиш <Ctrl>+<C> на клавиатуре.
3. Перейдите в каталог, в который нужно скопировать выделенные элементы.
4. Нажмите комбинацию клавиш <Ctrl>+<V> или щелкните правой кнопкой мыши и выберите в открывшемся контекстном меню команду **Вставить**.

15.4.3. Перемещение файлов и каталогов

Для перемещения файлов и каталогов выполните следующие действия:

1. С помощью мыши выделите файлы и каталоги, которые нужно переместить. Для этого щелкайте мышью на нужных элементах при нажатой клавише <Ctrl> (или <Shift> — если нужно выделить сразу диапазон файлов).
2. Щелкните правой кнопкой мыши на выделенных файлах и выберите в открывшемся контекстном меню (см. рис. 15.3) команду **Вырезать** или нажмите комбинацию клавиш <Ctrl>+<C> на клавиатуре.
3. Перейдите в каталог, в который нужно переместить выделенные элементы.
4. Нажмите комбинацию клавиш <Ctrl>+<V> или щелкните правой кнопкой мыши и выберите в открывшемся контекстном меню команду **Вставить**.

15.4.4. Переименование файлов и каталогов

Чтобы переименовать файл или каталог, щелкните на нем правой кнопкой мыши и выберите в открывшемся контекстном меню команду **Переименовать**.

Можно также щелкнуть на файле или каталоге левой кнопкой мыши и нажать клавишу <F2> на клавиатуре

15.4.5. Создание файлов и каталогов

Щелкните правой кнопкой мыши на свободном пространстве правой панели и выберите команду **Создать**. Затем выберите в меню одну из команд в зависимости от создаваемого объекта (рис. 15.4).

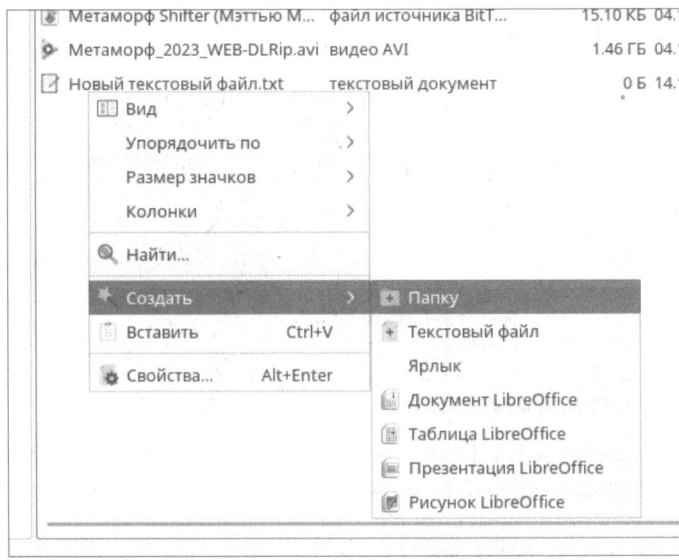

Рис. 15.4. Меню создания файлов и каталогов

Панель файловых операций

Для облегчения доступа к контекстному меню **Создать** включите панель файловых операций в меню **Вид**

15.4.6. Изменение прав доступа к файлам и каталогам

Для изменения прав доступа к файлам или каталогам:

1. Щелкните правой кнопкой мыши на файле или каталоге и выберите в открывшемся контекстном меню команду **Свойства**.
2. Перейдите в открывшейся панели на вкладку **Дискреционные атрибуты** (рис. 15.5).
3. Установите нужные права доступа.
4. Нажмите кнопку **Да**.

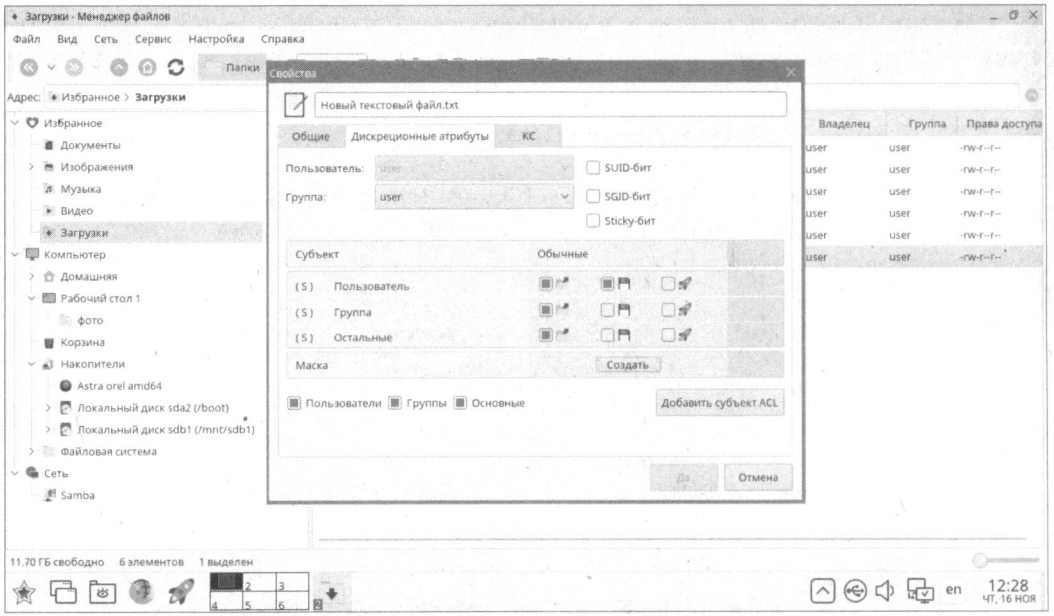

Рис. 15.5. Изменение прав доступа

15.4.7. Удаление файлов и каталогов

Выделите файлы и каталоги, которые нужно удалить, и нажмите клавишу <Delete>. В этом случае выделенные файлы и/или каталоги будут перемещены в Корзину, и вы сможете легко восстановить их в случае необходимости.

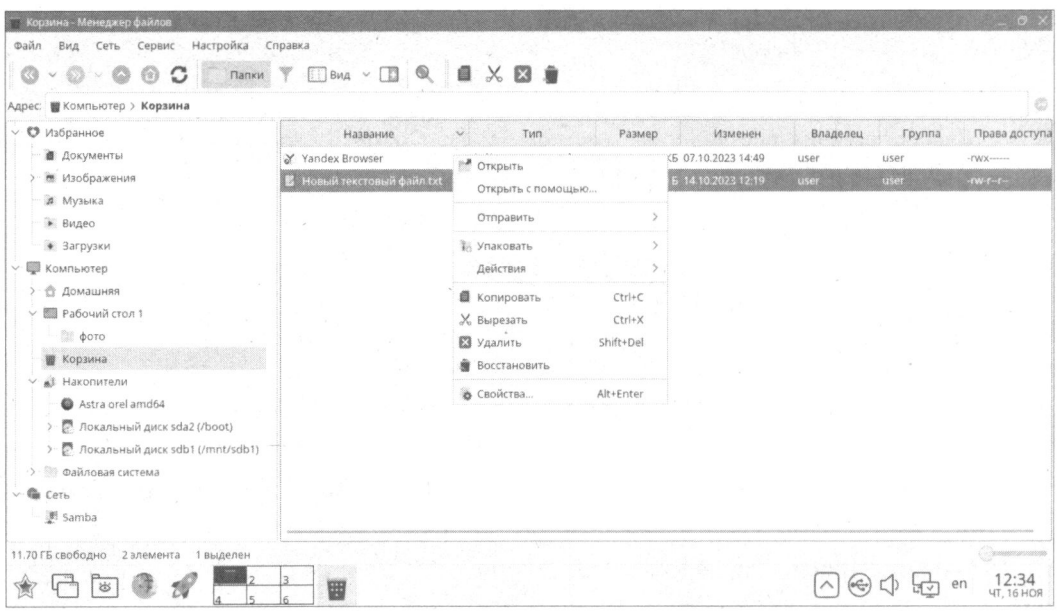

Рис. 15.6. Корзина и контекстное меню выделенного в ней файла

Если же нужно безвозвратно удалить выделенные элементы, нажмите комбинацию клавиш <Shift>+<Delete>.

Для просмотра содержимого Корзины выберите ее на левой панели файлового менеджера (рис. 15.6), а для восстановления файла — выделите в Корзине нужный файл и нажмите кнопку **Восстановить** на панели инструментов или воспользуйтесь контекстным меню.

15.5. Сжатие и распаковка файлов и каталогов

Сжатие файлов в архивы позволяет не только сэкономить место, но и облегчить их передачу по Интернету. Например, у вас есть архив документов, которыми вы пользуетесь не очень часто, но они занимают достаточно много места, а удалять их нельзя, — тогда их можно сжать.

Для сжатия используется команда `zip`. Далее приводятся два примера ее использования:

```
zip archive.zip report.doc
zip -r html.zip html
```

Первая команда сжимает файл report.doc — архив будет называться archive.zip. Вторая — сжимает подкаталог html, имя архива — html.zip.

Для распаковки архива служит команда `unzip`:

```
unzip html.zip
```

Распаковка осуществляется в текущий каталог.

Подробные сведения о командах `zip` и `unzip` вы можете получить в справочной системе, выполнив команды `man zip` и `man unzip` соответсвенно.

Для сжатия файлов/каталогов и распаковки архивов вы также можете использовать файловый менеджер — выделите файлы и/или каталоги, которые вы хотите сжать/распаковать, щелкните на выделении правой кнопкой мыши и выберите из контекстного меню команду **Упаковать** или **Распаковать** соответственно.

ASTRA LINUX®

ГЛАВА 16

Файловая система Linux. Монтирование

- Корневая файловая система
- Стандартные каталоги файловой системы
- Монтирование и размонтирование
- Файлы устройств и их монтирование
- Параметры монтирования файловых систем
- Монтирование разделов при загрузке
- Учет сменных накопителей в Astra Linux
- Обработка подключения устройств

16.1. Корневая файловая система

Вспомним, как устроена файловая система в Windows. Откройте Проводник Windows. Скорее всего, вы увидите значок гибкого диска (имя устройства **A:**), значки разделов жесткого диска (в нашем случае имеется один раздел — **C:**), значок привода CD/DVD (**D:**). Таким способом — с помощью буквенных обозначений A:, C:, D: и т. п. — в Windows обозначаются корневые каталоги разделов жесткого диска и сменных носителей.

В Linux существует понятие *корневой файловой системы*. Допустим, вы установили Linux в раздел с именем /dev/sda3 — в этом разделе и будет развернута корневая файловая система вашей Linux-системы. Корневой каталог обозначается прямым слешем — /, т. е. для перехода в корневой каталог в терминале (или в консоли) нужно ввести команду `cd /`.

Понятно, что на вашем жестком диске есть еще и другие разделы. Чтобы получить к ним доступ, вам нужно *подмонтировать* их к корневой файловой системе. После монтирования вы сможете обратиться к содержимому разделов через точку монтирования — назначенный вами при монтировании специальный каталог, например: /mnt/cdrom.

Обращение ко всем сменным носителям и другим накопителям, установленным в Linux-системе, осуществляется через корневую файловую систему.

16.2. Стандартные каталоги файловой системы

Файловая система любого дистрибутива Linux содержит следующие каталоги:

- / — корневой каталог;
- /bin — стандартные программы Linux (cat, cp, ls, login и т. д.);
- /boot — каталог загрузчика, содержит образы ядра и Initrd, может содержать конфигурационные и вспомогательные файлы загрузчика;
- /dev — файлы устройств;
- /etc — конфигурационные файлы системы;
- /home — домашние каталоги пользователей;
- /lib — библиотеки и модули;
- /lost+found — восстановленные после некорректного размонтирования файловой системы файлы и каталоги;
- /misc — может содержать все что угодно, равно как и каталог /opt;
- /mnt — обычно содержит точки монтирования;
- /proc — каталог псевдофайловой системы procfs, предоставляющей информацию о процессах;
- /root — каталог суперпользователя root;
- /sbin — каталог системных утилит, выполнять которые имеет право пользователь root;
- /tmp — каталог для временных файлов;
- /usr — пользовательские программы, документация, исходные коды программ и ядра;
- /var — постоянно изменяющиеся данные системы, например очереди системы печати, почтовые ящики, протоколы, замки и т. д.

16.3. Монтирование и размонтирование

Чтобы работать с какой-либо файловой системой, необходимо *примонтировать* ее к корневой файловой системе. Например, вставив в разъем USB флешку, нужно подмонтировать файловую систему флешки к корневой файловой системе — только так мы сможем получить доступ к файлам и каталогам, которые на этой флешке записаны. Аналогичная ситуация с жесткими, оптическими дисками и другими носителями данных.

Если вы хотите заменить сменный носитель данных (флешку, компакт-диск), вам нужно сначала *размонтировать* файловую систему, затем извлечь носитель дан-

ных, установить новый и заново смонтировать файловую систему. В случае с флешкой о размонтировании должны помнить вы сами, поскольку при этом выполняется синхронизация буферов ввода/вывода и файловой системы, т. е. данные физически записываются на носитель, если это еще не было сделано. А компакт-диск система не разрешит вам извлечь, если он не размонтирован. В свою очередь, размонтировать файловую систему можно только тогда, когда ни один процесс ее не использует.

При завершении работы системы (перезагрузке, выключении компьютера) размонтирование всех файловых систем выполняется автоматически.

Команда монтирования (ее нужно выполнять с привилегиями root) выглядит так:

```
# mount [опции] <устройство> <точка монтирования>
```

Здесь *точка монтирования* — это каталог, через который будет осуществляться доступ к монтируемой файловой системе. Например, если вы подмонтировали компакт-диск к каталогу /mnt/cdrom, то получить доступ к файлам и каталогам, записанным на компакт-диске, можно будет через точку монтирования (именно этот каталог: /mnt/cdrom). Точкой монтирования может быть любой каталог корневой файловой системы, хоть /aaa-111. Главное, чтобы этот каталог существовал на момент монтирования файловой системы.

В некоторых современных дистрибутивах запрещен вход в систему под именем суперпользователя — root. Поэтому для выполнения команд с привилегиями root вам нужно использовать команду `sudo`. Например, чтобы выполнить команду монтирования привода компакт-диска, вам нужно ввести команду:

```
sudo mount /dev/scd0 /mnt/cdrom
```

Перед выполнением команды `mount` команда `sudo` попросит вас ввести пароль root. Если введенный пароль правильный, то будет выполнена команда `mount`.

Для размонтирования файловой системы служит команда `umount`:

```
# umount <устройство или точка монтирования>
```

16.4. Файлы устройств и их монтирование

Для Linux нет разницы между устройством и файлом. Все устройства системы представлены в корневой файловой системе как обычные файлы. Например, /dev/fd0 — это ваш дисковод для гибких дисков (ведь вы все еще помните, что это за устройство?), /dev/sda — жесткий диск. Файлы устройств хранятся в каталоге /dev.

16.4.1. Жесткие диски и SSD

Все дисковые устройства, вне зависимости от интерфейса подключения (PATA, SATA, SCSI), называются /dev/sdx, где *x* — буква. Все современные дистрибутивы поддерживают udev (систему управления устройствами для новых версий ядра

Linux) и UUID (стандарт идентификации, используемый в создании программного обеспечения). Так что не удивляйтесь, если вдруг ваш старенький IDE-винчестер будет назван /dev/sda. С одной стороны, это вносит некоторую путаницу. С другой — все современные компьютеры оснащены именно SATA-дисками (т. к. PATA-диски уже устарели, а SCSI — дорогие), а на современных материнских платах только один контроллер IDE (PATA), потому многие пользователи даже ничего не заметят.

Рассмотрим ситуацию с жесткими дисками чуть подробнее. Пусть у нас есть устройство /dev/sda. На жестком диске, понятное дело, может быть несколько разделов. Предположим, что на диске имеются три раздела (логических диска), которые в Windows называются C:, D: и E:. Диск C: обычно является загрузочным (активным), поэтому этот раздел будет записан в самом начале диска. Нумерация разделов жесткого диска в Linux начинается с 1, и в большинстве случаев диску C: будет соответствовать имя /dev/sda1 — первый раздел на первом жестком диске.

Резонно предположить, что двум оставшимся разделам (D: и E:) будут присвоены имена /dev/sda2 и /dev/sda3. Это может быть и так и не так. Как известно, на жестком диске могут существовать или четыре первичных раздела, или три первичных и один расширенный. В расширенном разделе могут разместиться до 11 логических дисков (разделов). Таким образом, раздел может быть первичным (primary partition), расширенным (extended partition) или логическим (logical partition).

Для возможных четырех первичных разделов диска в Linux зарезервированы номера 1, 2, 3, 4. Если разделы D: и E: нашего диска первичные, то, да — им будут присвоены имена /dev/sda2 и /dev/sda3. Но в большинстве случаев эти разделы являются логическими и содержатся в расширенном разделе. Логические разделы именуются, начиная с 5, а это означает, что если разделы D: и E: — логические, им будут присвоены имена /dev/sda5 и /dev/sda6 соответственно.

Расширенный раздел Windows

В Windows расширенному разделу не присваивается буква, потому что этот раздел не содержит данных пользователя, а только информацию о логических разделах.

Имена устройств для SSD-накопителей, если они подключаются посредством интерфейса SATA (стационарные компьютеры и не очень новые ноутбуки), будут такими же, как в случае с SATA-дисками, — например, первый SSD-диск будет доступен как /dev/sda. А вот если SSD-накопитель подключен по интерфейсу NVMe, тогда имя устройства у него будет иметь вид /dev/nvme*X*n*Y*, и первый такой накопитель станет доступен по имени /dev/nvme0n1 (обратите внимание, что нумерация начинается с единицы — n1 — вместо букв a, b, c и т. д.).

Узнать номер раздела очень просто — достаточно запустить утилиту, работающую с таблицей разделов диска, — fdisk или популярный графический аналог GParted. Посмотрите на рис. 16.1. Здесь видно, что на диске созданы три первичных раздела, а расширенный раздел не создан — в нем нет необходимости в нашем случае. Корневая файловая система находится на устройстве /dev/sda1, файлы загрузчика — на устройстве /dev/sda2. Устройство /dev/sda3 используется как раздел подкачки.

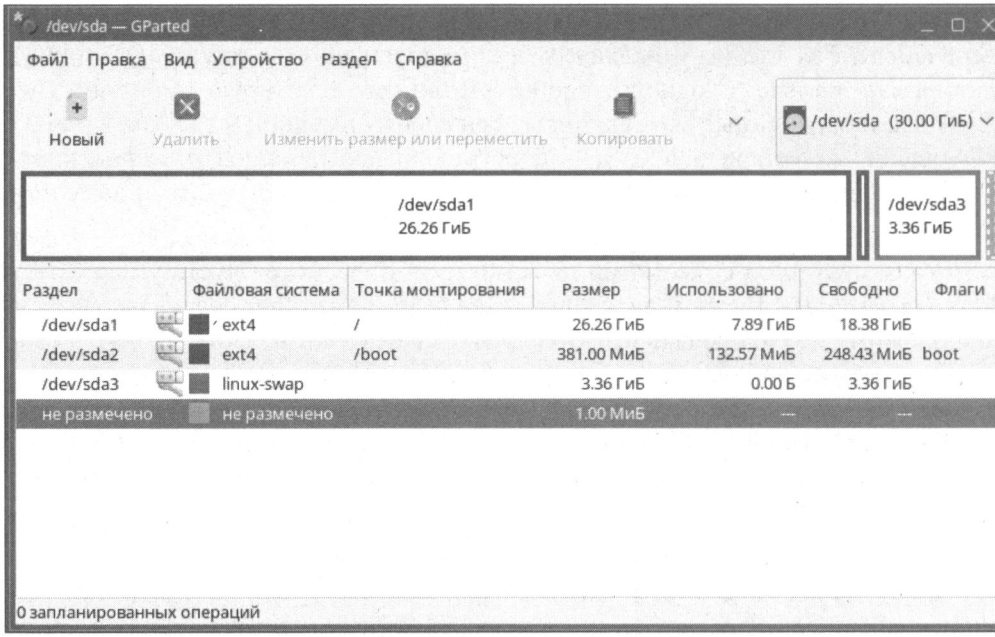

Рис. 16.1. Утилита Gparted: на диске созданы три первичных раздела

16.4.2. Приводы оптических дисков

Приводы для чтения/записи CD/DVD называются /dev/scd*N*, где *N* — номер устройства. Если у вас только один привод CD/DVD, то его имя будет /dev/scd0.

Монтирование привода для чтения оптических дисков осуществляется командой:

```
# mount /dev/scd0 /mnt/cdrom
```

После этого обратиться к файлам, записанным на диске, можно будет через каталог /mnt/cdrom. Напомню, что этот каталог должен существовать.

> ***Автоматическое монтирование***
>
> Как правило, в Astra Linux монтирование оптических приводов осуществляется автоматически. Как только вы вставите диск в привод, он будет подмонтирован автоматически к каталогу */media/cdrom*.

16.4.3. Флешки и внешние жесткие диски

Флешки (флеш-память) и внешние USB-диски определяются системой как обычные жесткие диски. Предположим, что в компьютере установлен всего один жесткий диск — тогда ему соответствует имя устройства /dev/sda.

Когда вы подключите флешку или внешний жесткий диск, этому устройству будет присвоено имя /dev/sdb. Обычно на флешке или USB-диске всего один раздел, поэтому подмонтировать устройство можно командой:

```
# mount /dev/sdb1 /mnt/usbdisk
```

Автоматическое монтирование

Флешки и внешние жесткие диски монтируются автоматически к каталогу */media/<ID-устройства>*. После подключения соответствующего носителя вы сможете обратиться к его файлам через этот каталог.

Небольшие проблемы возникли с автоматическим монтированием SD-карты, установленной в кардридер ноутбука. Система увидела устройство, но отказалась его автоматически монтировать (рис. 16.2).

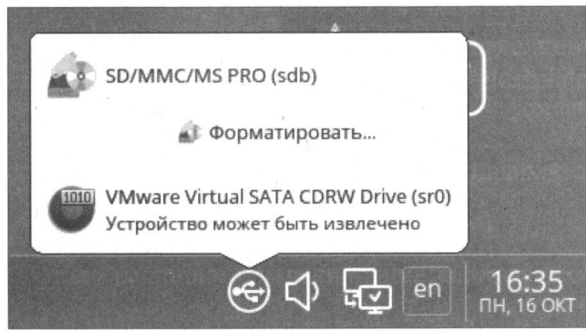

Рис. 16.2. Система видит SD-карту

Получить доступ к файлам на SD-карте можно так:

```
sudo mkdir /media/sd
sudo mount /dev/sdb1 /media/sd
ls -l /media/sd
```

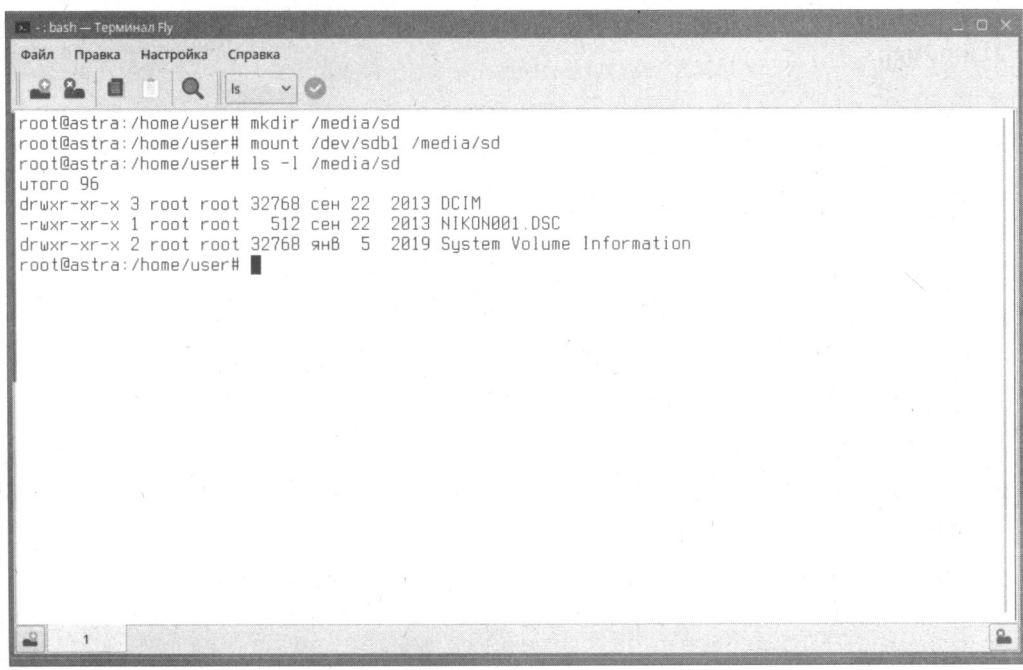

Рис. 16.3. Просмотр содержимого SD-карты

Первая команда создает каталог /media/sd, к которому мы будем монтировать SD-карту. Эту команду нужно вводить только однажды, затем вы просто будете использовать каталог /media/sd при монтировании.

Вторая команда монтирует устройство /dev/sdb1 (конкретное имя устройства показано на рис. 16.2) к ранее созданному каталогу /media/sd. Третья команда выводит содержимое SD-карты (рис. 16.3).

16.5. Параметры монтирования файловых систем

Теперь, когда мы знаем номер раздела, можно подмонтировать его файловую систему. Делается это так:

```
# mount <раздел> <точка монтирования>
```

Например:

```
# mount /dev/sda5 /mnt/win_d
```

У команды `mount` довольно много опций, но на практике наиболее часто используются только некоторые из них: `-t`, `-r`, `-w`, `-a`.

- Параметр `-t` позволяет задать тип файловой системы. Обычно программа сама определяет файловую систему, но иногда это у нее не получается. Тогда мы должны ей помочь. Формат этого параметра следующий:

  ```
  # mount -t <файловая система> <устройство> <точка монтирования>
  ```

  Например,

  ```
  # mount -t iso9660 /dev/sdc /mnt/cdrom
  ```

  Вот опции для указания наиболее популярных монтируемых файловых систем:

  - `ext2`, `ext3`, `ext4` — файловая система Linux;
  - `iso9660` — указывается при монтировании CD-ROM;
  - `vfat` — FAT, FAT32 (поддерживаются Windows 9*x*, ME, XP);
  - `ntfs` — NT File System (поддерживаются Windows NT, XP, 7, 8, 10/11). Будет использована стандартная поддержка NTFS, при которой NTFS-раздел доступен только для чтения;
  - `ntfs-3g` — будет запущен модуль ntfs-3g, входящий в большинство современных дистрибутивов. Этот модуль позволяет производить запись информации на NTFS-разделы.

    Модуль ntfs-3g

    Если в вашем дистрибутиве нет модуля ntfs-3g, то при попытке указания файловой системы NTFS будет выведено сообщение об ошибке. В таком случае вы можете скачать его с сайта **www.ntfs-3g.org**. На этом сайте доступны как исходные коды, так и уже откомпилированные для разных дистрибутивов пакеты.

Если вы не можете смонтировать NTFS-раздел с помощью опции `ntfs-3g`, то, вероятнее всего, он был неправильно размонтирован (например, работа Windows не была завершена корректно). В этом случае для монтирования раздела нужно использовать опцию `-o force`, например:

```
sudo mount -t ntfs-3g /dev/sdb1 /media/usb -o force
```

- Параметр `-r` монтирует указанную файловую систему в режиме «только чтение».
- Параметр `-w` монтирует файловую систему в режиме «чтение/запись». Этот параметр используется по умолчанию для файловых систем, поддерживающих запись (например, NTFS по умолчанию запись не поддерживает, как и файловые системы CD/DVD-дисков).
- Параметр `-a` служит для монтирования всех файловых систем, указанных в файле /etc/fstab (кроме тех, для которых указано `noauto`, — такие файловые системы нужно монтировать вручную). При загрузке системы тогда вызывается команда `mount` с параметром `-a`.

16.6. Монтирование разделов при загрузке

Если вы не хотите при каждой загрузке монтировать постоянные файловые системы (например, Windows-разделы), то нужно прописать их в файле /etc/fstab. Обратите внимание — в этом файле не следует прописывать файловые системы сменных носителей (дисковода, CD/DVD-привода, флешки). Следует отметить, что программы установки некоторых дистрибутивов читают таблицу разделов и автоматически заполняют файл /etc/fstab — в результате все ваши Windows-разделы оказываются доступны сразу после установки системы. К сожалению, не все дистрибутивы могут похвастаться такой интеллектуальностью, поэтому вам нужно знать формат файла fstab:

```
устройство точка_монтирования тип_ФС опции флаг_РК флаг_проверки
```

Здесь: *тип_ФС* — это тип файловой системы, а *флаг_РК* — флаг резервного копирования. Если он установлен (`1`), то программа dump заархивирует эту файловую систему при создании резервной копии. Если не установлен (`0`), то резервная копия ее создаваться не будет. *Флаг_проверки* устанавливает, будет ли эта файловая система проверяться на наличие ошибок программой fsck. Проверка производится в двух случаях:

- если файловая система размонтирована некорректно;
- если достигнуто максимальное число операций монтирования для этой файловой системы.

Поле *опции* содержит важные параметры файловой системы. Некоторые из них представлены в табл. 16.1.

Таблица 16.1. *Опции монтирования файловой системы в файле /etc/fstab*

Опция	Описание
auto	Файловая система должна монтироваться автоматически при загрузке. Опция используется по умолчанию, поэтому ее указывать не обязательно
noauto	Файловая система не монтируется при загрузке системы (при выполнении команды `mount -a`), но ее можно смонтировать вручную с помощью все той же команды `mount`
defaults	Используется стандартный набор опций, установленных по умолчанию
exec	Разрешает запуск выполняемых файлов для этой файловой системы. Опция используется по умолчанию
noexec	Запрещает запуск выполняемых файлов для этой файловой системы
ro	Монтирование в режиме «только чтение»
rw	Монтирование в режиме «чтение/запись». Используется по умолчанию для файловых систем, поддерживающих запись
user	Эту файловую систему разрешается монтировать/размонтировать обычному пользователю (не root)
nouser	Файловую систему может монтировать только пользователь root. Используется по умолчанию
umask	Определяет маску прав доступа при создании файлов. Для не-Linux файловых систем маску нужно установить так: `umask=0`
utf8	Применяется только на дистрибутивах, которые используют кодировку UTF8 в качестве кодировки локали. В старых дистрибутивах (где используется KOI8-R) для корректного отображения русских имен файлов на Windows-разделах нужно задать параметры `iocharset=koi8-u`, `codepage=866`

Редактирование файла /etc/fstab

Редактировать файл */etc/fstab*, как и любой другой файл из каталога */etc*, можно в любом текстовом редакторе (например, gedit, kate), но перед этим нужно получить права root (командами `su` или `sudo`).

Рассмотрим небольшой пример:

```
/dev/sdc /mnt/cdrom auto umask=0,user,noauto,ro,exec 0 0
/dev/sda1 /mnt/win_c vfat umask=0,utf8 0 0
```

Первая строка — это строка монтирования файловой системы компакт-диска, а вторая — строка монтирования диска C:.

- Начнем с первой строки. `/dev/sdc` — это имя устройства CD-ROM. Точка монтирования — `/mnt/cdrom`. Понятно, что этот каталог должен существовать. Обратите внимание — в качестве файловой системы не указывается жестко iso9660, поскольку компакт-диск может быть записан в другой файловой системе, поэтому в качестве типа файловой системы задано `auto`, т. е. автоматическое определение. Далее идет достаточно длинный набор опций. Ясно, что `umask` установлен в ноль, поскольку файловая система компакт-диска не поддерживает права доступа Linux. Параметр `user` говорит о том, что эту файловую систему можно монтировать

обычному пользователю. Параметр `noauto` запрещает автоматическое монтирование этой файловой системы, что правильно — ведь на момент монтирования в приводе может и не быть компакт-диска. Опция `ro` разрешает монтирование в режиме «только чтение», а `exec` разрешает запускать исполнимые файлы. Понятно, что компакт-диск не нуждается ни в проверке, ни в создании резервной копии, поэтому два последних флага равны нулю.

- Вторая строка проще. Первые два поля — это устройство и точка монтирования. Третье — тип файловой системы. Файловая система постоянна, поэтому можно явно указать тип файловой системы (`vfat`), а не `auto`. Опция `umask`, как и в предыдущем случае, равна нулю. Указание опции `utf8` позволяет корректно отображать русскоязычные имена файлов и каталогов.

16.7. Учет сменных накопителей в Astra Linux

Версия Astra Linux Special Edition оснащена механизмом учета сменных накопителей, с помощью которого администратор может запретить определенным пользователям доступ к определенным носителям (или же запретить определенные операции — например, запись).

Посмотрим, как работает этот механизм. Откройте конфигуратор **Политика безопасности** и перейдите в раздел **Устройства и правила | Устройства**. Нажмите кнопку **+** для добавления нового устройства. Вы увидите приглашение подключить устройство (рис. 16.4). Подключать устройство нужно именно в этот момент — не раньше. Если оно уже подключено, отключите его и подключите снова.

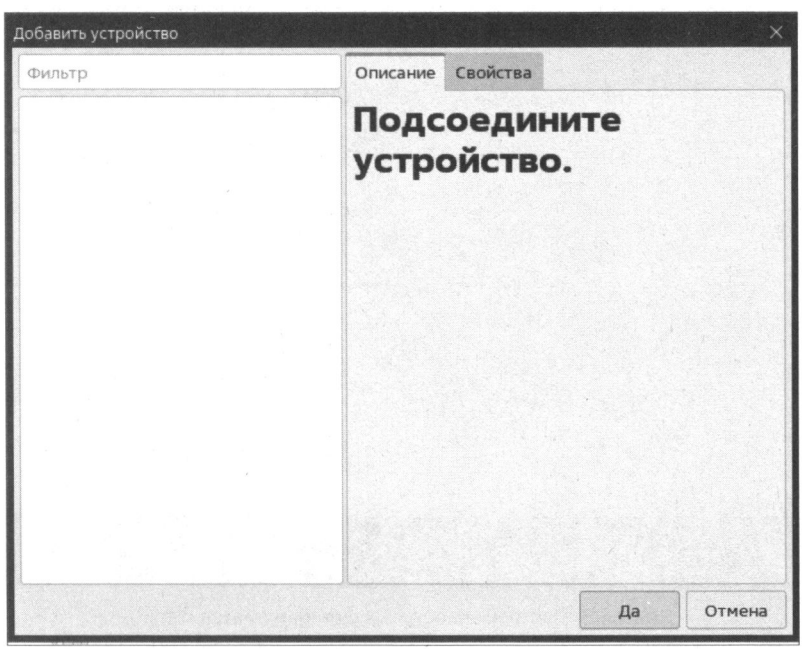

Рис. 16.4. Приглашение подключить устройство

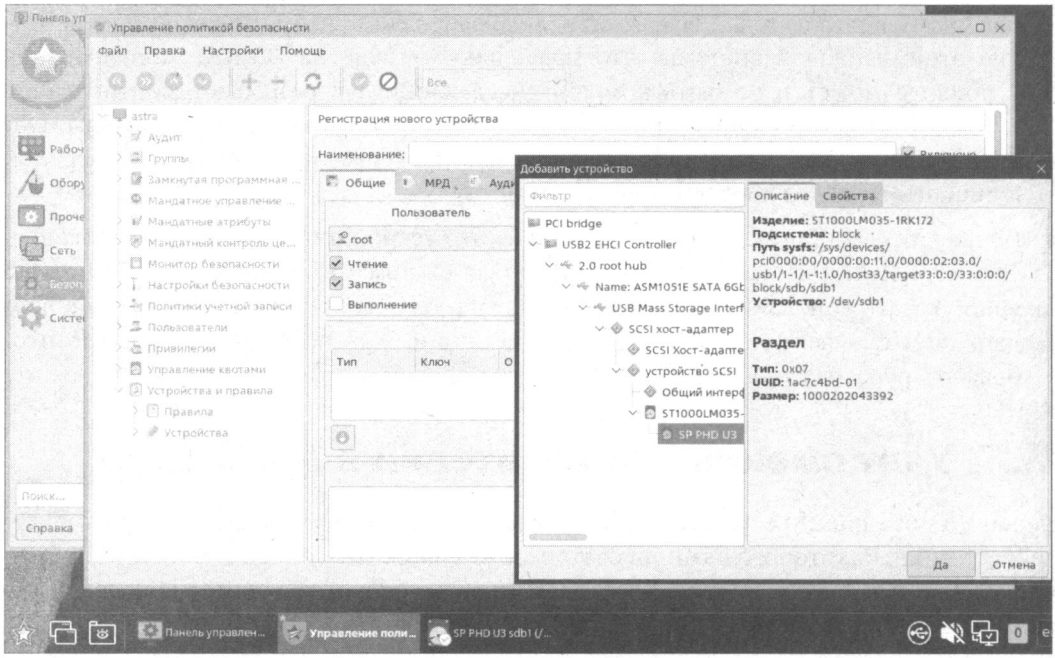

Рис. 16.5. Добавление устройства

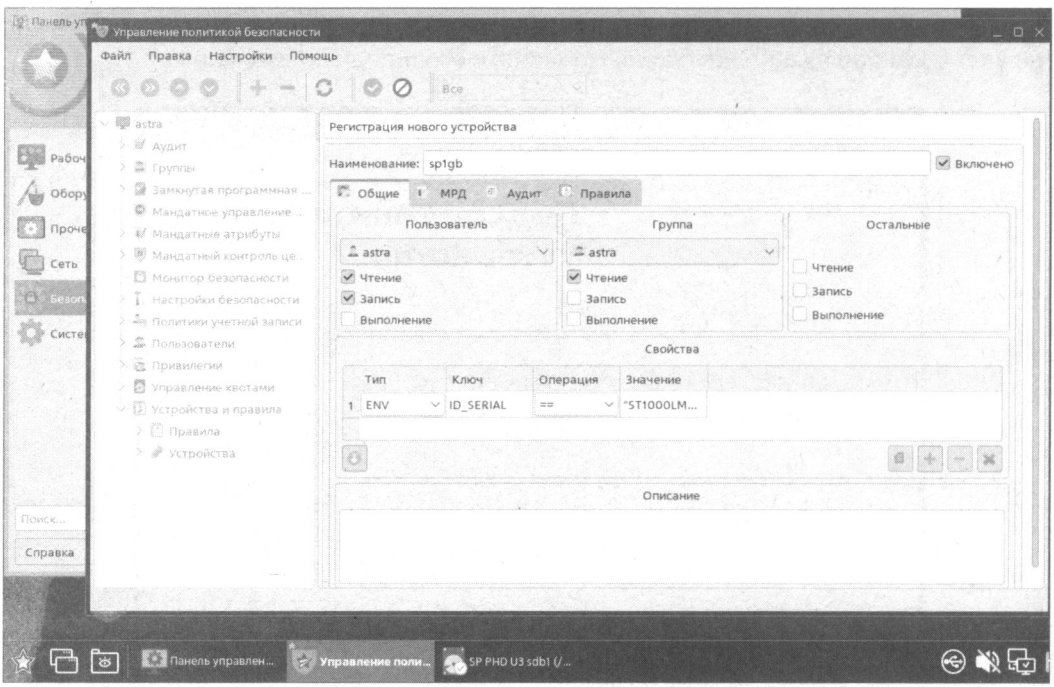

Рис. 16.6. Настройка доступа к сменному накопителю

Теперь нужно выбрать само устройство — как правило, оно будет последним в списке (рис. 16.5). Выберите его и нажмите кнопку **Да**.

Далее нужно ввести наименование устройства (все, что вам хочется), а также установить, кто будет обладать доступом к устройству. На рис. 16.6 разрешено чтение/запись для пользователя **astra**, а для членов группы **astra** — только чтение Всем остальным любой доступ запрещен.

Когда все будет готово, нажмите зеленую галку для сохранения конфигурации. Далее можете войти под другим пользователем и протестировать — доступа к этому устройству у вас не будет.

16.8. Обработка подключения устройств

Когда вы подключаете сменное устройство, то можете выбрать один из вариантов предлагаемых системой действий (рис. 16.7).

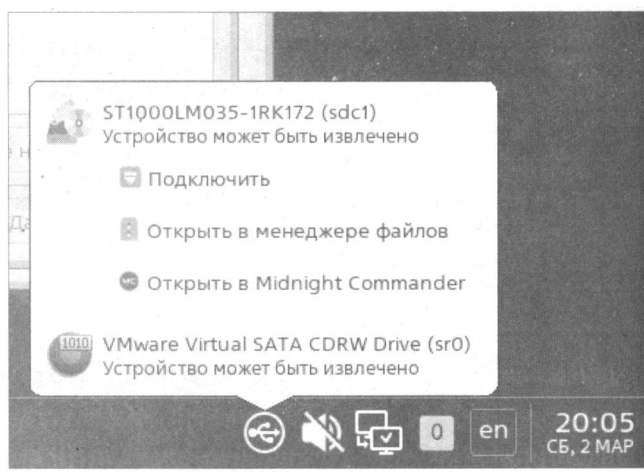

Рис. 16.7. Выберите действие

В нашем случае был подключен внешний жесткий диск (**sdc1**), и система предложила подключить его, открыть в менеджере файлов или в Midnight Commaner. Отредактировать эти действия можно с помощью конфигуратора **Обработка подключения устройств** (через группу **Оборудование** Панели управления). На рис. 16.8 показан этот конфигуратор и предлагаемые им действия.

Щелкнув на действии, вы можете отредактировать его — например, задать тип устройств, для которого оно должно показываться, или вовсе отключить его, если оно вам не нужно (рис. 16.9).

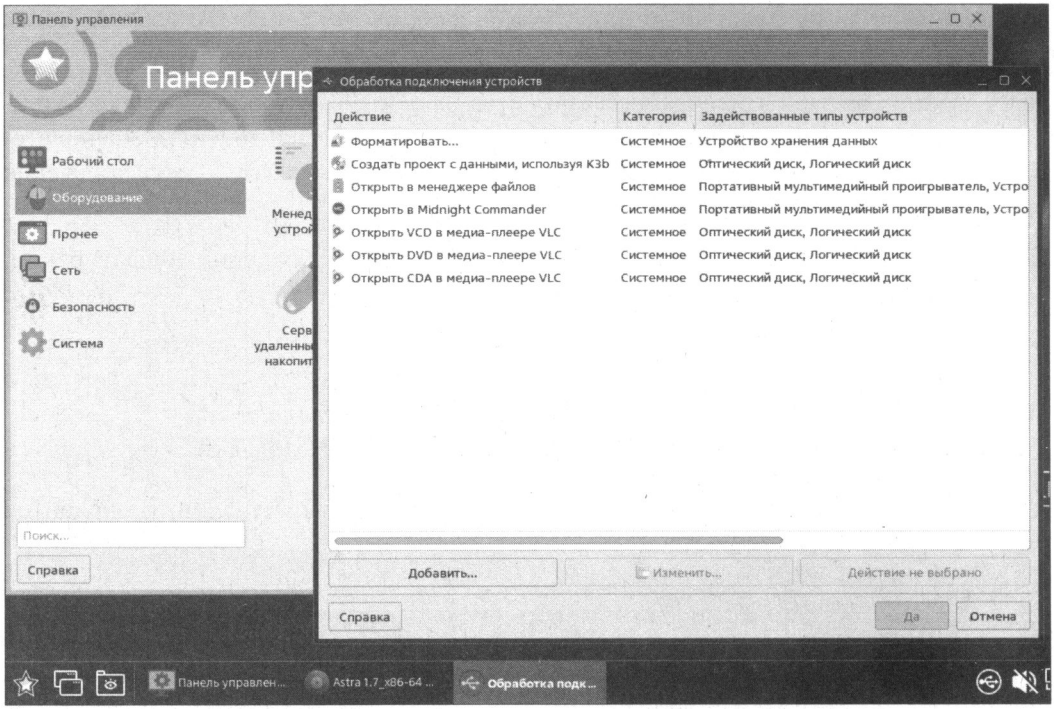

Рис. 16.8. Обработка подключения устройств

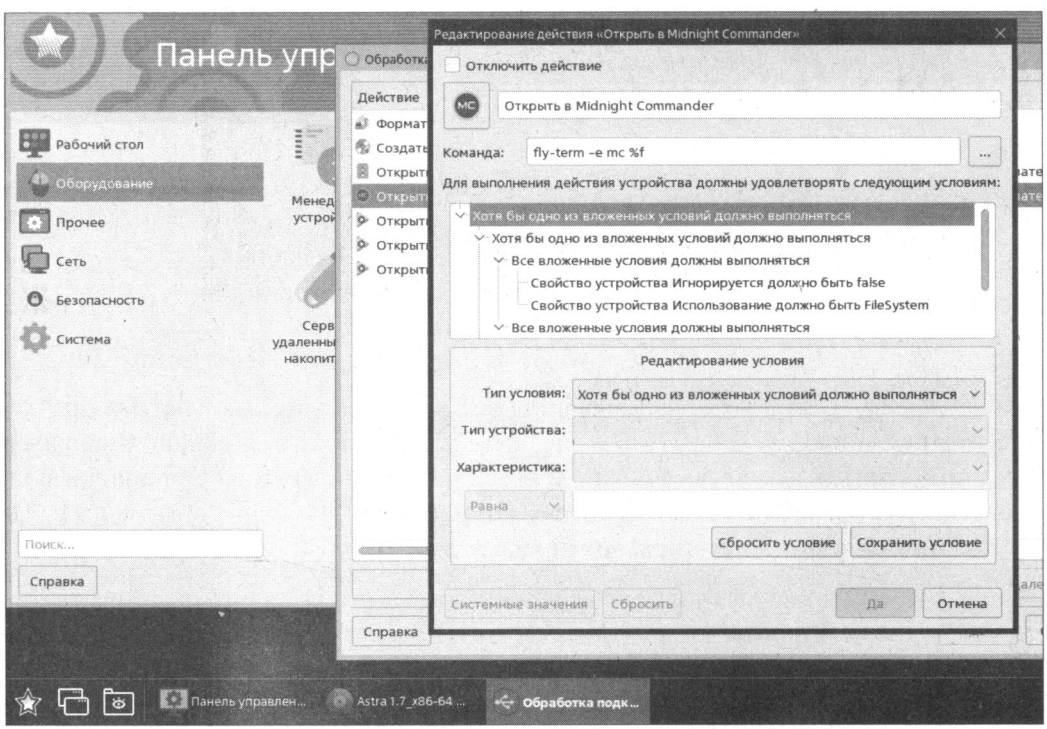

Рис. 16.9. Редактирование действия

ASTRA LINUX®

ГЛАВА 17

Особые операции с файловой системой

- Монтирование ISO-образа
- Создание файловой системы
- Проверка и восстановление файловой системы
- Смена корневой файловой системы
- Монтирование каталога к каталогу
- Команды поиска файлов
- Примеры использования команды `dd`

17.1. Монтирование ISO-образа

Некоторое ПО распространяется в виде ISO-файлов. Но не записывать же образ на диск? Особенно если DVD-дисковода давно нет. Linux, в отличие от Windows, позволяет смонтировать ISO-образ безо всякого стороннего ПО. Просто с помощью команды `mount`:

```
sudo mount -o loop образ.iso каталог
```

Опция `-o loop` как раз и обеспечивает монтирование ISO-образа образ.iso к указанному каталогу. Разумеется, каталог должен существовать на момент вызова этой команды, поэтому не забудьте его создать.

Размонтировать образ можно командой:

```
umount каталог
```

Пример:

```
mkdir ~/debian
sudo mount -o loop debian9.iso ~/debian
ls -l ~/debian
umount ~/debian
```

Мы создали каталог debian, подмонтировали к нему образ debian9.iso, вывели содержимое корневого каталога образа ISO и размонтировали образ.

17.2. Создание файловой системы

С помощью команды `mkfs` мы можем создать файловую систему на разделе жесткого диска (говоря терминологией Windows — отформатировать диск), например, так:

```
mkfs.ext4 /dev/sdb1
```

Вообще, создать файловую систему нужного типа (если эта файловая система поддерживается ядром вашей системы) можно с помощью команды:

```
mkfs.<имя_файловой_системы>
```

Например, так:

```
mkfs.ext4
mkfs.vfat
mkfs.reiserfs
```

Подробнее прочитать об этом можно, введя команду:

```
man mkfs.<имя_файловой_системы>
```

17.3. Проверка и восстановление файловой системы

Для проверки файловой системы служит команда `fsck`. Использовать ее нужно так:

```
fsck <раздел>
```

Например:

```
fsck /dev/sda5
```

> ***Внимание!***
>
> Перед использованием этой команды нужно размонтировать проверяемую файловую систему. Если надо проверить корневую файловую систему, то следует загрузиться с LiveCD и запустить `fsck` для проверки необходимого раздела.

Если же жесткий диск «посыпался», т. е. на нем появились «плохие» блоки, нужно, не дожидаясь полной потери данных, выполнить следующие действия:

1. Выполнить команду (она пометит «плохие» блоки):

   ```
   fsck -c <раздел>
   ```

2. Сделать резервную копию всех важных данных.
3. Отправиться в магазин за новым жестким диском и перенести данные со старого жесткого диска на новый. Проверить жесткий диск на наличие плохих секторов можно программой `badblocks`.

Команда fsck

Команда `fsck` может проверять не только файловые системы ext*. Для проверки, например, файловой системы vfat можно использовать команду:

```
fsck.vfat <раздел>
```

Для восстановления «упавшей» таблицы разделов можно использовать программу `gpart`. Только применяйте ее осторожно и внимательно читайте все сообщения, выводимые программой.

17.4. Смена корневой файловой системы

Предположим, мы установили Windows после установки Linux, и программа установки Windows перезаписала начальный загрузчик. Теперь Windows загружается, а Linux — нет. Что делать? Нужно загрузиться с LiveCD и выполнить команду:

```
# chroot <раздел, содержащий корневую файловую систему>
```

Например, если Linux был установлен в раздел /dev/sda5, то нужно ввести команду:

```
# chroot /dev/sda5
```

Эта команда сменит корневую файловую систему, т. е. вы загрузите ядро Linux с LiveCD, а затем сделаете подмену корневой файловой системы. Вам останется только ввести команду записи загрузчика (например, `grub-install`) для восстановления начального загрузчика.

17.5. Установка скорости CD/DVD

Утилита `hdparm` позволяет ограничить скорость оптического привода CDROM/DVDROM. Иногда нужно ограничить эту скорость, чтобы информация была считана без ошибок (как правило, если поверхность носителя информации немного повреждена). Рассмотрим команду ограничения скорости:

```
# hdparm -q -E<множитель> <устройство>
```

Здесь *множитель* — это и есть скорость. Например, `1` соответствует скорости 150 Кбайт/с для CD, 1385 Кбайт/с для DVD. Чтобы установить вторую скорость чтения для CD (2, 300 Кбайт/с), используется команда:

```
# hdparm -q -E2 /dev/cdrom
```

Для ограничения скорости DVD можно использовать следующую команду:

```
# hdparm -q -E1 /dev/dvd
```

17.6. Монтирование каталога к каталогу

В Linux можно подмонтировать каталог к каталогу, а не только каталог к устройству. Делается это с помощью все той же команды `mount`, запущенной с параметром `--bind`:

```
# mount --bind исходный_каталог каталог_назначения
```

17.7. Команды поиска файлов

Для поиска файлов в Linux служит команда `find`. Это довольно мощная утилита со сложным синтаксисом, и далеко не всегда нужная обычному пользователю. Обычному пользователю намного проще установить файловый менеджер `mc` и использовать встроенную в него функцию поиска.

Но команду `find` мы все же рассмотрим, по крайней мере, ее основы. Синтаксис команды следующий:

```
find список_поиска выражение
```

Мощность программы `find` заключается в множестве самых разных параметров поиска, которые не так легко запомнить, — их слишком много. К тому же `find` может выполнять команды для найденных файлов. Например, вы можете найти временные файлы и сразу удалить их.

Подробно опции команды `find` мы рассматривать не будем — это вы можете сделать самостоятельно с помощью команды `man find`. Зато мы рассмотрим несколько примеров использования этой команды.

Найти файлы с именем a.out (точнее, в имени которых содержится строка a.out), а поиск начать с корневого каталога (/):

```
find / -name a.out
```

Найти файлы по маске `*.txt`:

```
find / -name '*.txt'
```

Найти файлы нулевого размера, поиск начать с текущего каталога (.):

```
find . -size 0c
```

Хотя для поиска пустых файлов намного проще использовать параметр `-empty`:

```
find . -empty
```

Найти файлы, размер которых от 100 до 150 Мбайт, поиск производить в домашнем каталоге и всех его подкаталогах:

```
find ~ -size +100M -size -150M
```

Найти все временные файлы и удалить их (для каждого найденного файла будет запущена команда `rm`):

```
# find / -name *.tmp -ok rm {} \;
```

Вместо параметра `-ok` можно использовать параметр `-exec`, который также запускает указанную после него команду, но не запрашивает подтверждение выполнения этой команды для каждого файла.

Кроме команды `find`, можно использовать команды `which` и `locate`. Первая выводит полный путь к программе или к сценарию, если программа или сценарий находится в списке каталогов, заданном в переменной окружения `PATH`:

```
which sendmail
```

Программа `locate` ищет в базе данных демона `located` файлы, соответствующие заданному образцу. Недостаток этой команды в том, что `located` имеется далеко не во всех дистрибутивах, поэтому команды `locate` у вас может и не быть. Зато если `located` имеется и запущен, поиск файлов будет осуществляться быстрее, чем с помощью `find`.

17.8. Примеры использования команды *dd*

Команда `dd` в Linux особенная. Ее синтаксис не похож на синтаксис других команд, а благодаря ее широкой функциональности эта команда заслужила особое внимание. Именно поэтому ей посвящен отдельный раздел в этой главе.

Команда `dd` очень старая, но тем не менее весьма полезная. Ее история начинается вместе с историей UNIX — 1 января 1970 года. Тогда эта команда использовалась для работы с ленточными накопителями. Почему эту команду назвали именно так — `dd`, — теперь уже никто не вспомнит. Есть различные варианты (вроде Data Definition, определение данных), но точного определения названия не сохранилось.

Об этой команде можно написать даже целую главу, в которой точно представить ее синтаксис, историю и даже придумать несколько десятков вариантов названий для нее. Но я не стану это делать. Во-первых, синтаксис команды `dd` описан в man (обязательно прочитайте страницу руководства), а во-вторых, особого интереса в его досконально изучении нет. А интересны практические примеры использования этой команды, которые пригодятся всем пользователям! На них, собственно, мы и остановимся. Но прежде вы должны знать, что команда очень опасна, и ее неправильное использование может полностью уничтожить данные на вашем жестком диске — и не только на Linux-разделе! Поэтому я снимаю с себя всю ответственность за дальнейшее описание ее возможностей, а вы должны принять, что используете эту команду на свой страх и риск. Если вас что-то в таком подходе не устраивает, тогда забудьте о ней и перейдите к чтению следующего раздела.

Да, команда `dd` очень опасна. Но если использовать ее осторожно, можно извлечь для себя много пользы. Далее мы рассмотрим типичные примеры использования этой команды.

17.8.1. Копирование файлов с помощью *dd*

Обычно никто не копирует файлы командой `dd` — для этого есть команда `cp`. Но при желании можно выполнить копирование и командой `dd`. Мы же рассматриваем копирование файлов, чтобы немного разобраться с синтаксисом команды, который проще изучить на безобидных командах, которые не повредят ваши данные. Сразу скажу, что для выполнения команды `dd` нужны права root, поэтому здесь и далее приглашение командной строки будет иметь вид # (в некоторых дистрибутивах вы не можете зарегистрироваться как root, поэтому придется выполнить эту команду или через команду `sudo`, или через команду `su`).

Итак, чтобы скопировать файл /home/user/example.txt в файл /home/user/example.bak, выполните следующую команду:

```
# dd if=/home/user/example.txt of=/home/user/example.bak
```

Параметр `if` (input file) задает входной файл, параметр `of` (output file) — выходной. Как видите, синтаксис немного не обычный для Linux и имеет вид «опция=значение». Была бы `dd` обычной командой, пример ее использования выглядел бы так:

```
-if /home/user/example.txt -of /home/user/example.bak
```

Ничего интересного, как видите, мы не сделали: просто скопировали один файл в другой, что можно было бы достичь и средствами команды `cp`. Но интересное только начинается. Команде `dd` все равно, что будет входным, а что — выходным файлом, она работает с необработанными данными (raw) на самом низком уровне — на уровне секторов жесткого диска. Именно поэтому эта команда так опасна. Ведь она может запросто взять секторы одного диска и заменить ими секторы другого диска. В результате вместо данных на диске останется нечитаемая «каша».

Параметры `if` и `of` правильнее назвать входным и выходным буферами. Величина буфера задается параметром `bs` (block size). По умолчанию размер блока равен 512 байтам, что подходит для большинства задач.

Параметр `count` задает число блоков, которое должно быть считано. Давайте сейчас попробуем считать главную загрузочную запись (MBR) вашего жесткого диска в файл:

```
# dd if=/dev/sda of=mbr.bak bs=512 count=1
```

Эта команда скопировала всего 512 байтов первого жесткого диска (/dev/sda) вашего компьютера и поместила их в файл mbr.bak. Теперь скопируйте этот файл на флешку и храните в безопасном месте. Если что-то повредит ваш MBR, вы всегда сможете восстановить его так (здесь /dev/sdb1 — это имя флешки):

```
# dd if=/dev/sdb1/mbr.bak of=/dev/sda bs=512 count=1
```

Вот вам и первый практически полезный пример использования этой команды!

17.8.2. Разделение файла на несколько частей

Бывает так, что большой файл не полностью помещается на носитель и его нужно разбить на несколько частей. Следующие две команды разделят файл размером 1000 Мбайт на два файла по 500 Мбайт (`part1` и `part2`):

```
dd if=file1000 of=part1 bs=1M count=500
dd if=file1000 of=part2 bs=1M skip=500
```

Параметр `skip` задает смещение, т. е. команда должна пропустить первые 500 частей по 1 Мбайт, т. к. они уже вошли в первую часть — `part1`.

В качестве домашнего задания вам будет нужно соединить файлы `part1` и `part2` в один исходный файл.

17.8.3. Создание резервной копии жесткого диска

Представим, что у нас есть два одинаковых жестких диска (/dev/sda и /dev/sdb) и вам нужно создать резервную копию жесткого диска /dev/sda на диске /dev/sdb. Это делается следующей командой:

```
# dd if=/dev/sda of=/dev/sdb conv=noerror,sync bs=4k
```

Обратите внимание, что мы установили размер блока в `4k` (4096 байт), чтобы повысить производительность копирования. Если не задать размер блока, то будет использовано значение по умолчанию (512 байтов), в итоге число операций чтения/записи будет увеличено в 8 раз, что негативно отразится на производительности.

17.8.4. Создание архива с резервной копией всего жесткого диска

Обычно у нас нет двух одинаковых жестких дисков, но зато все еще есть необходимость создать резервную копию. Сейчас мы создадим архив с резервной копией жесткого диска /dev/sda. Сама команда `dd` не умеет создавать архивы, но мы можем перенаправить ее вывод команде `gzip`, которая и сделает основную работу.

Итак, для резервного копирования выполните следующую команду:

```
# dd if=/dev/sda | gzip > /mnt/sdb1/backups/sda.img.gz
```

Созданный архив будет записан в файл /mnt/sdb1/backups/sda.img.gz. В случае необходимости архив можно развернуть обратно на диск /dev/sda с помощью следующей команды (только выполнять ее нужно, предварительно загрузившись с LiveCD):

```
# gzip -dc /mnt/sdb1/sda.img.gz | dd of=/dev/sda
```

17.8.5. Уничтожение всех данных раздела жесткого диска

Командой `dd` можно легко нечаянно уничтожить данные на всем жестком диске. Но сейчас мы разберемся с тем, как уничтожить намеренно и целенаправленно данные определенного раздела жесткого диска. Ведь не секрет, что при удалении файлов и даже при форматировании диска данные все еще можно восстановить. А вот с помощью `dd` можно перезаписать все секторы жесткого диска нулями или случайными числами, после чего уже никто и никогда не восстановит старые данные, поскольку они будут безвозвратно перезаписаны:

```
# dd if=/dev/zero of=/dev/sdb2 bs=1M
# dd if=/dev/urandom of=/dev/sdb2 bs=1M
```

Первая команда перезаписывает все секторы раздела sdb2 нулями, а вторая — случайными числами. На момент ввода одной из этих команд устройство /dev/sdb2 должно быть размонтировано, а после ввода подмонтировать устройство будет невозможно. Чтобы снова использовать этот раздел, нужно его отформатировать, т. е. заново создать файловую систему (как это сделать командой `mkfs`, было показано ранее), а после создания файловой системы раздел можно будет снова подмонтировать.

ГЛАВА 18

Права доступа

- Концепция прав доступа
- Смена владельца файла
- Групповое изменение прав доступа
- Специальные права доступа: SUID и SGID
- Атрибуты файла. Запрет изменения файла

18.1. Концепция прав доступа

Операционная система Linux, независимо от применяемого дистрибутива, является многопользовательской. И поскольку пользователей может быть в системе несколько, все они могут создавать файлы. Поэтому вводится понятие владельца файла/каталога. *Владелец* — это пользователь, создавший файл/каталог.

Чтобы не было беспорядка, пользователь может создать файлы не в любом месте файловой системы, а, как правило, только в своем домашнем каталоге или в любом другом, если так определено *правами доступа* к каталогу. Исключение — пользователь root, у которого максимальные права доступа: он может создавать, редактировать и удалять файлы/каталоги в любом месте файловой системы.

При создании файла права доступа к нему назначаются по умолчанию: как правило, владельцу разрешается полный доступ к файлу, а всем остальным — только чтение.

> ***ПРАВА ДОСТУПА ПО УМОЛЧАНИЮ***
>
> Сменить права доступа по умолчанию можно командой `umask`. Как правило, на домашнем компьютере сменой прав доступа никто не занимается, но эта команда может пригодиться администратору сервера. Узнать подробности о команде `umask` можно командой `man umask`.

Изменить права доступа к файлу/каталогу может владелец, а также пользователь root. Для этого используется команда `chmod`.

Зачем изменять права доступа?

Например, вам нужно, чтобы к вашему файлу-отчету смогли получить доступ пользователи — члены вашей группы. Или вы создали обычный текстовый файл, содержащий инструкции командного интерпретатора. Чтобы этот файл стал сценарием, надо установить право на выполнение для этого файла. На сервере неправильно установленные права доступа могут стать настоящей проблемой — сервер могут попросту взломать.

Существуют три права доступа: чтение (`r`), запись (`w`), выполнение (`x`). Для каталога право на выполнение означает право на просмотр содержимого каталога.

Вы можете установить разные права доступа для владельца (т. е. для себя), для группы владельца (т. е. для всех пользователей, входящих в одну с владельцем группу) и для прочих пользователей. Пользователь root может получить доступ к любому файлу или каталогу вне зависимости от прав, которые вы установили.

Чтобы просмотреть текущие права доступа, введите команду:

```
ls -l <имя файла/каталога>
```

Например:

```
ls -l contacts.txt
```

В ответ программа выведет следующую строку:

```
-r--r----- 1 user group 300 Apr 11 11:11 contacts.txt
```

В этой строке фрагмент: `-r--r-----` описывает права доступа:

- первый символ (`-`) — это признак каталога. Сейчас перед нами файл. Если бы перед нами был каталог, то первым символом шел бы **`d`** (от directory);
- последующие три символа (**`r--`**) определяют *права доступа владельца файла или каталога*. Первый символ — это чтение, второй — запись, третий — выполнение. Как можно видеть, владельцу разрешено только чтение этого файла, а запись и выполнение запрещены, поскольку в правах доступа режимы **`w`** и **`x`** не определены;
- следующие три символа (**`r--`**) задают *права доступа для членов группы владельца*. Права здесь такие же, как и у владельца: можно читать файл, но нельзя изменять или запускать;
- последние три символа (`---`) задают *права доступа для прочих пользователей*. Прочие пользователи не имеют права ни читать, ни изменять, ни выполнять этот файл. При попытке получить доступ к нему они увидят сообщение: **`Отказано в доступе`** (рис. 18.1). В графическом интерфейсе сообщение об ошибке зависит от приложения, но суть остается той же — не хватает прав доступа (рис. 18.2).

Полный вывод команды ls

Как можно видеть, после символов прав доступа команда `ls` выводит имя владельца файла, имя группы владельца, размер файла, дату и время создания, а также имя файла.

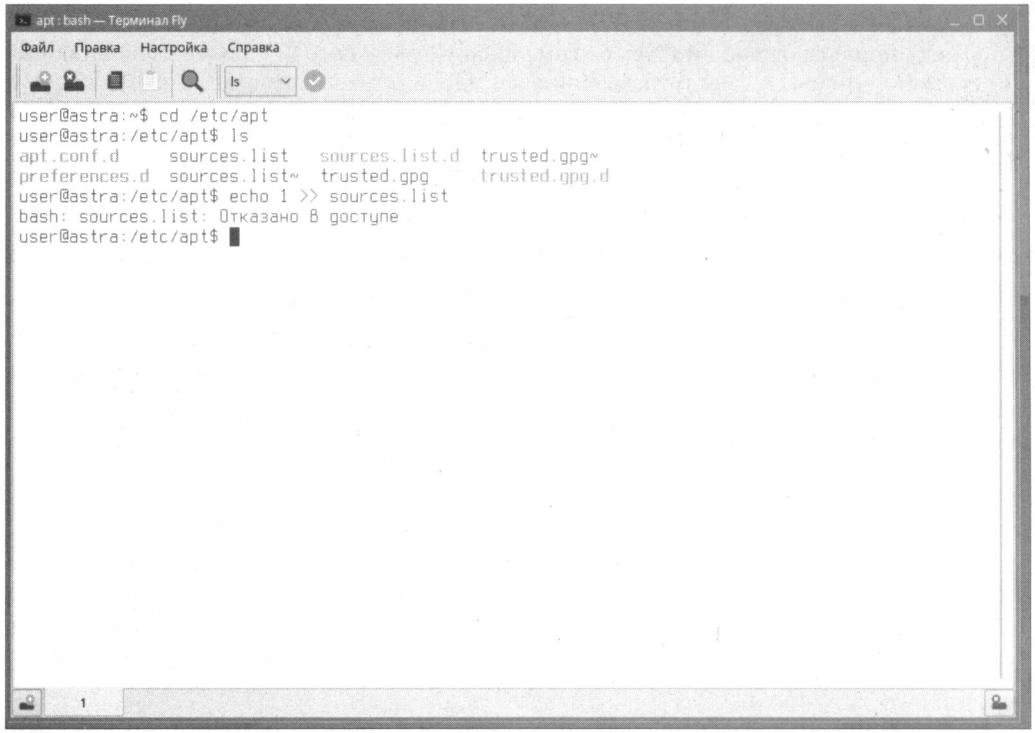

Рис. 18.1. Сообщение об отказе в доступе (командная строка)

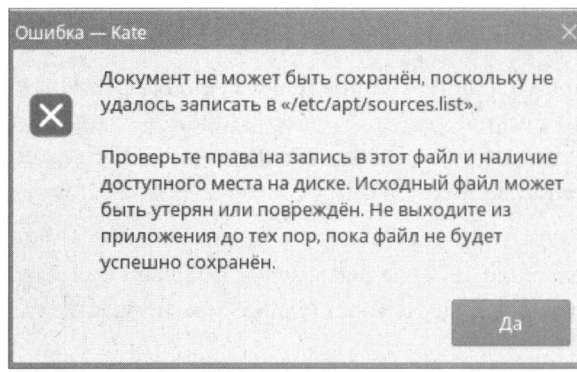

Рис. 18.2. Сообщение об отказе в доступе (графический интерфейс)

Права доступа задаются командой `chmod`. Существуют два способа указания прав доступа: *символьный* (когда указываются символы, задающие право доступа: `r`, `w`, `x`) и *абсолютный*.

Так уж заведено, что в мире UNIX-подобных систем чаще пользуются абсолютным методом. Разберемся, в чем он заключается, и рассмотрим для этого следующий набор прав доступа:

```
rw-r-----
```

Этот набор предоставляет владельцу право чтения и модификации файла (`rw-`), запускать файл владелец не может. Члены группы владельца могут только просматривать файл (`r--`), а все остальные пользователи не имеют вообще никакого доступа к файлу.

Возьмем отдельный набор прав — например, для владельца: `rw-`.

Чтение разрешено — мысленно записываем 1, запись разрешена — запоминаем еще 1, а вот выполнение запрещено, поэтому запоминаем 0. Получается число 110. Если перевести число 110 из двоичной системы в восьмеричную, получится число 6. Для перевода можно воспользоваться табл. 18.1.

Таблица 18.1. *Преобразование чисел из двоичной системы в восьмеричную*

Двоичная система	Восьмеричная система	Двоичная система	Восьмеричная система
000	0	100	4
001	1	101	5
010	2	110	6
011	3	111	7

Аналогично произведем разбор прав для членов группы владельца. Получится двоичное 100, т. е. восьмеричное 4. С третьим набором (`---`) все вообще просто — это 000, т. е. 0.

Записываем полученные числа в восьмеричной системе в порядке владелец-группа-остальные. Получится число 640 — это и есть права доступа. Для того чтобы установить эти права доступа, выполните команду:

```
chmod 640 <имя_файла>
```

Наиболее популярные права доступа:

- 644 — владельцу можно читать и изменять файл, остальным пользователям — только читать;
- 666 — читать и изменять файл можно всем пользователям;
- 777 — всем можно читать, изменять и выполнять файл.

> ***Право выполнения для каталога***
>
> Напомню, что для каталога право выполнения — это право просмотра оглавления каталога.

Иногда символьный метод оказывается проще. Например, чтобы файл script сделать исполнимым, можно отдать команду:

```
chmod +x script
```

Для того чтобы снять право выполнения, указывается параметр `-x`:

```
chmod -x script
```

Подробнее о символьном методе вы сможете прочитать в руководстве по команде `chmod` (выполнив команду `man chmod`).

Изменить права доступа в Astra Linux можно и с помощью графического интерфейса Fly. Для этого выполните следующие действия:

1. Откройте **Менеджер файлов** и перейдите в папку **Домашняя** — там проще найти файлы, владельцем которых вы являетесь (для остальных права может изменить только root).
2. Щелкните правой кнопкой мыши на интересующем вас файле/каталоге и выберите команду **Свойства**.
3. Перейдите на вкладку **Дискреционные атрибуты** (рис. 18.3). Там вы увидите табличку, позволяющую назначить права доступа для пользователя, группы и остальных. Флажок рядом со значком папки — это право просмотра (открытия) файла (read), флажок рядом со значком дискеты — право записи (write), а флажок рядом со значком ракеты — право запуска для файла и право просмотра содержимого для каталога. О том, что такое биты SGID/SUID/Sticky, вы узнаете в следующих разделах главы.
4. Нажмите кнопку **Да** для сохранения изменений.

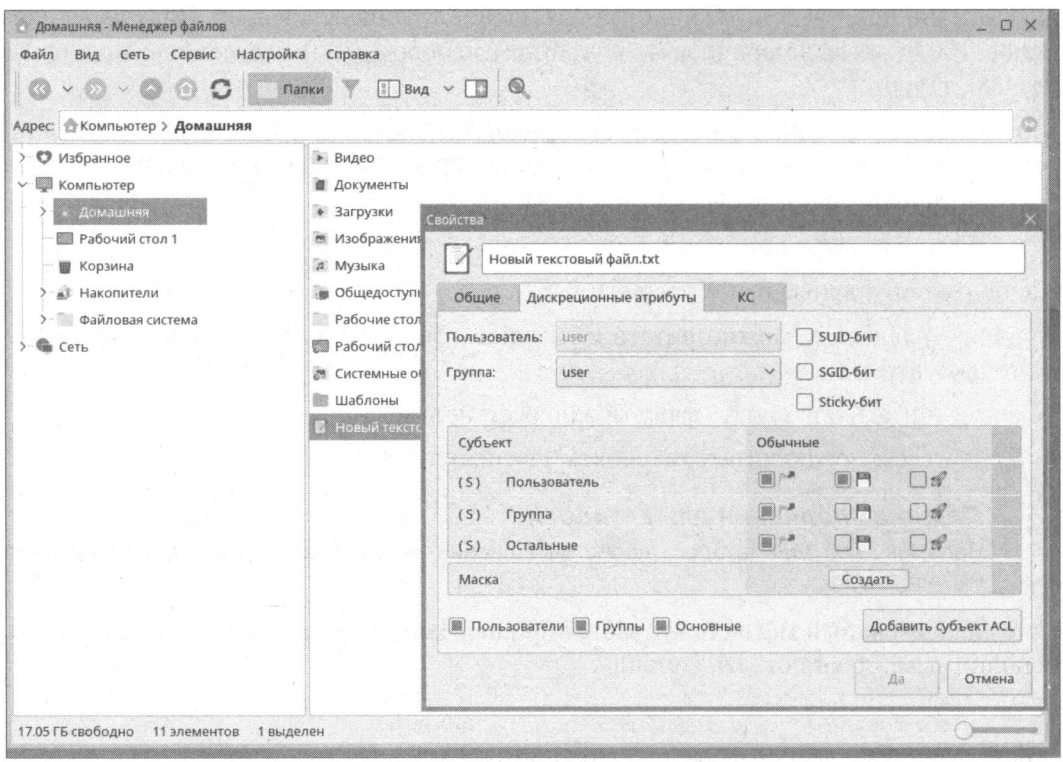

Рис. 18.3. Изменение прав доступа с помощью графического интерфейса

18.2. Смена владельца файла

Если вы хотите «подарить» кому-то файл, т. е. сделать какого-либо пользователя его владельцем, вам нужно использовать команду `chown`:

```
chown пользователь файл
```

> ***Последствия изменения владельца файла***
>
> Возможно, что после изменения владельца файла вы сами не сможете получить к нему доступ, ведь владельцем будете уже не вы. Чтобы этого не произошло, попробуйте сначала назначить этому файлу максимальные права доступа с помощью команды `chmod 666 <имя_файла>`, а затем уже измените его владельца командой `chown`.

18.3. Групповое изменение прав доступа

Иногда нужно изменить права доступа (либо владельца) не для одного файла, а сразу для многих. Если эти файлы/каталоги находятся в текущем каталоге, можно использовать маску `*` — например:

```
chown www-data:www-data *
chmod 644 *
```

Если же нужно изменить права доступа также и для файлов, которые находятся в подкаталогах, удобно использовать опцию `-R` (рекурсия). Например:

```
chown -R www-data:www-data *
```

> ***Внимание!***
>
> При групповом изменении прав доступа нужно быть предельно аккуратным. Помните, что право запуска для файла — это право просмотра для каталога. Если вы выполните команду `chmod 666 *` для предоставления полного доступа к файлам в текущем каталоге, и в этом каталоге будут подкаталоги, то для них также будут установлены права доступа 666, и вы потеряете возможность просматривать содержимое этих каталогов. Если же установить для всех файлов/каталогов права 777, то вы обеспечите полный доступ к каталогам, однако также и предоставите право выполнения для всех файлов, — очевидно, это не то, что бы вы хотели. В этом случае правильнее использовать команду `find`, где вы можете разграничить права доступа для файлов и каталогов. Например:
>
> ```
> sudo find . -type f -exec chmod 666 {} \;
> sudo find . -type d -exec chmod 777 {} \;
> ```
>
> Первая команда назначает права 666 всем файлам (параметр `-type f`) в текущем каталоге, а вторая — права 777 всем подкаталогам в текущем каталоге.

18.4. Специальные права доступа: SUID и SGID

Мы рассмотрели обычные права доступа к файлам, но в Linux есть еще так называемые *специальные права доступа*: SUID (Set User ID root) и SGID (Set Group ID root). Именно эти биты можно было установить на вкладке **Дискреционные атрибуты** (см. рис. 18.3).

Специальные права доступа позволяют обычным пользователям запускать программы, требующие для своего запуска привилегий пользователя root. Например, демон pppd требует привилегий root, но чтобы каждый раз при установке PPP-соединения (модемное или ADSL-соединение) не входить в систему под именем root, достаточно установить специальные права доступа для демона pppd. Делается это так:

```
chmod u+s /usr/sbin/pppd
```

Однако не нужно увлекаться такими решениями, поскольку каждая программа, для которой установлен бит SUID, является потенциальной «дырой» в безопасности системы. Для выполнения программ, требующих прав root, намного рациональнее использовать команды `sudo` и `su`, описание которых можно получить по командам справочной системы `man sudo` и `man su`.

18.5. Атрибуты файла. Запрет изменения файла

С помощью команды `chattr` можно изменить атрибуты файла. Параметр (+) устанавливает атрибут, а параметр (-) — атрибут снимает. Например:

```
# chattr +i /etc/apt/sources.lst
```

Эта команда устанавливает атрибут `i`, запрещающий любое изменение, переименование и удаление файла (так называемый Sticky-бит). Установить этот атрибут, равно как и снять его, имеет право только суперпользователь или процесс с возможностью `CAP_LINUX_IMMUTABLE`. Чтобы изменить файл, нужно очистить атрибут с помощью команды:

```
# chattr -i /etc/apt/sources.lst
```

Если установить атрибут `j`, то все данные, прежде чем быть записанными непосредственно в файл, будут сохранены в журнал файловой системы. Этот атрибут имеет смысл, только если файловая система смонтирована с опциями `data=ordered` или `data=writeback`. Когда файловая система смонтирована с опцией `data=journal`, установка атрибута `j` не имеет смысла, поскольку все данные файла и так уже журналируются.

Рассмотрим еще несколько атрибутов:

- когда для файла установлен атрибут `A` (прописная буква!), тогда не происходит обновление записи `atime` (в ней хранится время доступа к файлу). Это позволяет избежать лишних дисковых операций ввода/вывода, что полезно для медленных компьютеров;
- если для файла установлен атрибут `a`, в файл можно только добавлять данные. Этот атрибут имеет право установить (или очистить) суперпользователь или процесс с возможностью `CAP_LINUX_IMMUTABLE`;
- атрибут `c` заставляет систему упаковывать (сжимать) содержимое файла, что позволяет сэкономить место на диске. При записи в файл информация автомати-

чески сжимается и записывается на диск в уже сжатом виде, при чтении из этого файла возвращаются несжатые данные;

- когда изменяется каталог с установленным атрибутом `D`, изменения сразу же записываются на диск. Это эквивалентно применению опции монтирования `dirsync`;
- если для файла установлен атрибут `d`, для него не будет выполнено резервное копирование командой `dump`;
- при изменении файла с установленным атрибутом `S` его данные синхронно записываются на диск. Это аналогично опции монтирования `sync` к подмножеству файлов;
- когда удаляется файл с установленным атрибутом `s`, система выполняет обнуление его блоков и запись их обратно на диск;
- при удалении файла с атрибутом `u` его содержимое сохраняется на диске, что позволяет впоследствии легко восстановить этот файл;
- атрибуты `X` и `Z` используются экспериментальными заплатками сжатия для служебных целей.

Установить любой атрибут можно командой `chattr`, а просмотреть — командой `lsattr`. Об остальных атрибутах вы сможете прочитать в справочной системе, выполнив команду: `man chattr`.

ЧАСТЬ V

Аппаратные средства

ASTRA LINUX®

ГЛАВА 19

Получение информации о ПК. Псевдофайловые системы

- Что такое псевдофайловые системы?
- Виртуальная файловая система sysfs
- Виртуальная файловая система proc
- Сохранение произведенных изменений

19.1. Что такое псевдофайловые системы?

Особую роль в Linux играют так называемые *псевдофайловые* системы. Слово «псевдо», как мы знаем, означает «почти», т. е. псевдофайловая система — не совсем файловая система в прямом смысле этого слова. Псевдофайловые системы также называются *виртуальными*, поскольку работают они на уровне виртуальной файловой системы (Virtual File System layer). Для большинства пользователей виртуальная файловая система выглядит как обычная файловая система — можно открыть тот или иной файл и посмотреть, что в нем записано, можно записать в него ту или иную информацию. Ради интереса зайдите в каталог /proc (это каталог псевдофайловой системы proc) и посмотрите на размер любого файла — например, /proc/filesystems. Его размер будет равен 0, как и остальных файлов этой файловой системы, но если открыть сам файл, то вы увидите, что информация в нем есть. Это объясняется тем, что содержимое файла формируется при обращении к нему, т. е. на лету. Другими словами, виртуальная файловая система находится в оперативной памяти, а не на жестком диске. Информация попадает в файл на основании сведений, полученных от ядра.

В большинстве современных дистрибутивов используются виртуальные файловые системы sysfs и proc. Откройте файл /etc/fstab, и вы увидите строки монтирования этих файловых систем:

```
sysfs    /sys      sysfs   defaults        0 0
proc     /proc     proc    defaults        0 0
```

19.2. Виртуальная файловая система sysfs

Виртуальная (псевдофайловая) система sysfs экспортирует в пространство пользователя информацию о ядре Linux, об имеющихся в системе устройствах и их драйверах. Впервые sysfs появилась в ядре версии 2.6. Зайдите в каталог /sys — названия подкаталогов говорят сами за себя:

- block — содержит каталоги всех блочных устройств, имеющихся в системе в текущее время (под устройством подразумевается совокупность физического устройства и его драйвера). Когда вы подключаете Flash-диск, то в любом случае в каталоге /sys/devices/ появляется новое устройство, но в каталоге /sys/block это устройство появится только при наличии соответствующих драйверов (в указанном случае — драйвера usb-storage);
- bus — перечень шин, поддерживаемых ядром (точнее, зарегистрированных в ядре). В каждом каталоге шины есть подкаталоги devices и drivers. В каталоге devices находятся ссылки на каталоги всех устройств, которые описаны в системе (т. е. находящихся в каталоге /sys/devices);
- class — по этому каталогу можно понять, как устройства формируются в классы. Для каждого устройства в каталоге class есть свой отдельный каталог (под устройством, как и в случае с каталогом block, подразумевается совокупность устройства и его драйвера);
- devices — содержит файлы и каталоги, которые полностью соответствуют внутреннему дереву устройств ядра;
- drivers — каталоги драйверов для загруженных устройств. Подкаталог drivers каталога шины содержит драйверы устройств, работающих на этой шине.

Теперь немного практики. Перейдите в каталог /sys/block. В нем вы найдете список доступных блочных устройств. На рис. 19.1 видно, что, кроме устройств loop*, в этом каталоге есть устройства sda, sdb и sr0. Диск sda — это первый накопитель, подключенный к SATA-контроллеру, sdb — второй и т. д. Устройство sr0 — это привод CD/DVD. Если зайти в папку, принадлежащую накопителю, вы увидите, помимо всего прочего, — папки, соответствующие разделам этого диска: sda1, sda2, sda3.

Вывести список разделов конкретного диска можно так:

```
ls /sys/block/sda | grep sda
```

Обратите внимание, что команда `ls -l /sys/block/sda/sda*` выводит не список файлов (рис. 19.2), а системную информацию, поэтому если вам нужен просто список разделов, то это проще сделать через `grep`.

Файл /sys/block/*<устройство>*/*<раздел>*/size содержит размер (в килобайтах) этого раздела, а размер всего накопителя находится в файле /sys/block/*<устройство>*. На рис. 19.3 показаны размер накопителя и каждого из его разделов.

Каталог /sys/bus/cpu/devices позволяет быстро узнать, сколько ядер у вашего процессора. Просто перейдите в этот каталог и посчитайте количество подкаталогов cpu*. На рис. 19.4 показано, что у процессора два ядра: cpu0 и cpu1.

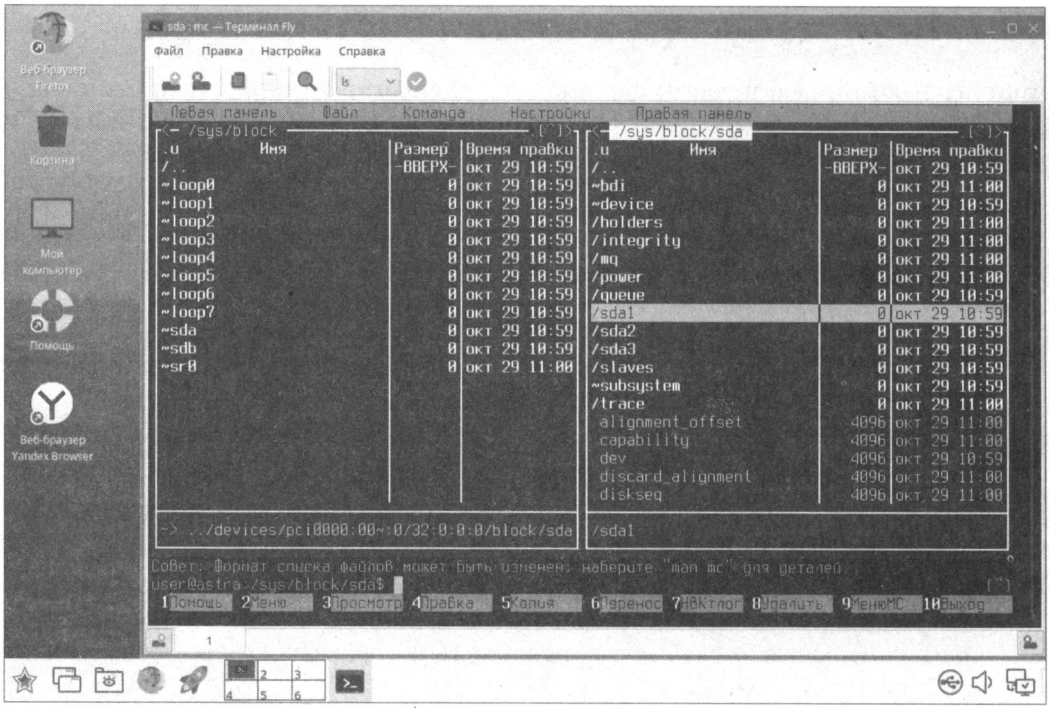

Рис. 19.1. Каталог /sys/block

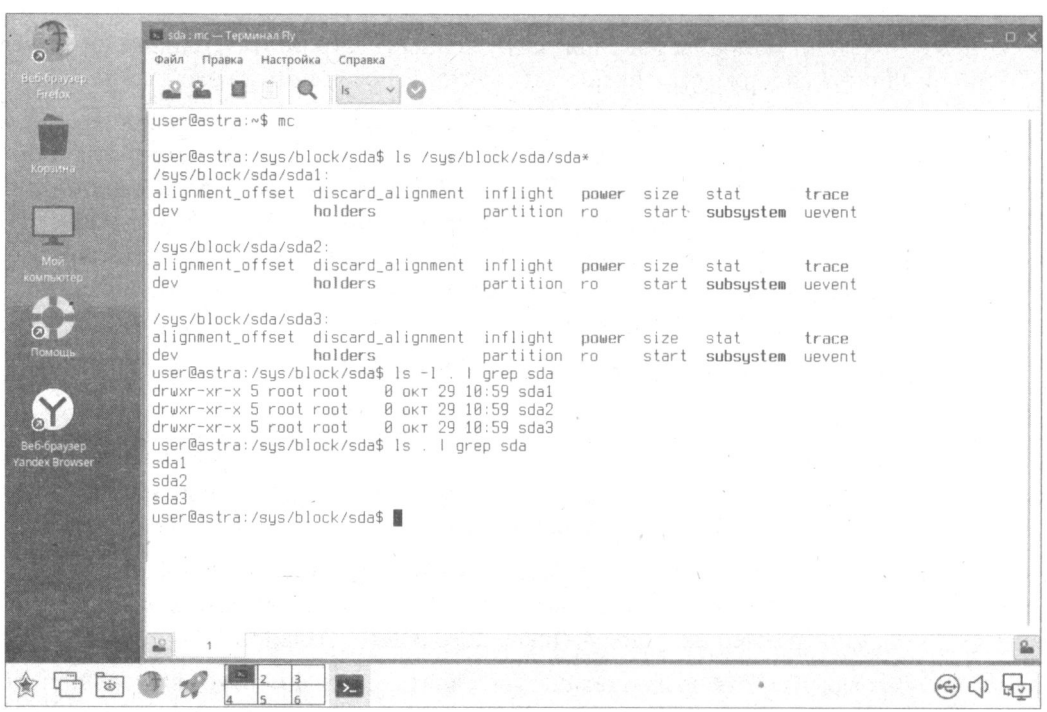

Рис. 19.2. Получение информации о разделах накопителя sda

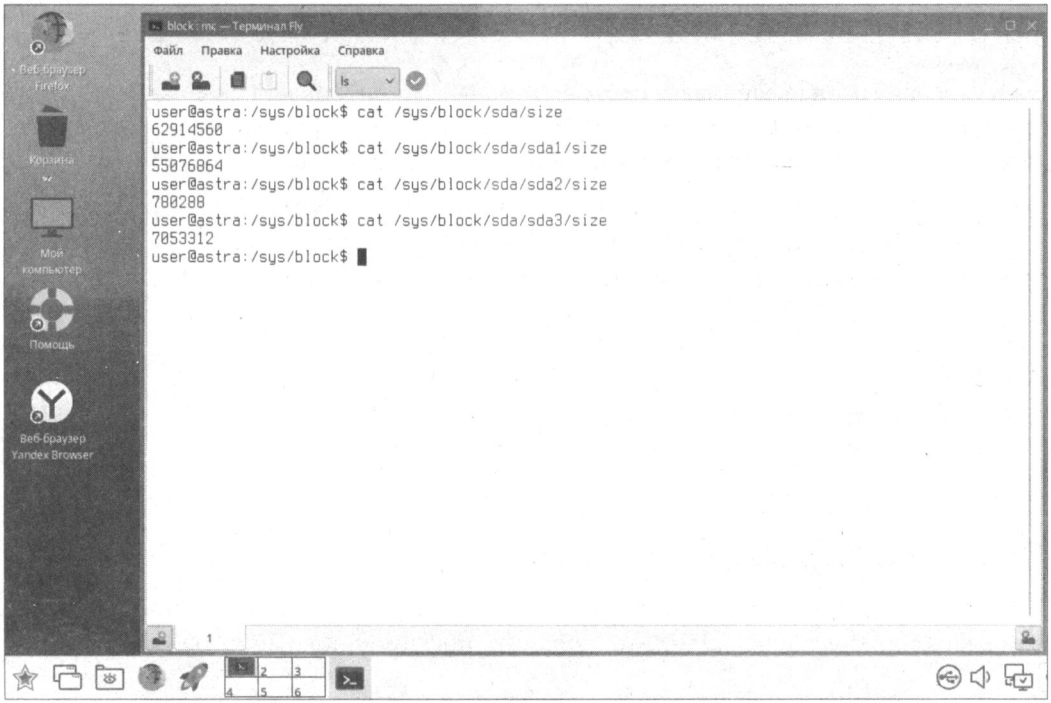

Рис. 19.3. Получение информации о размере диска и каждого из разделов

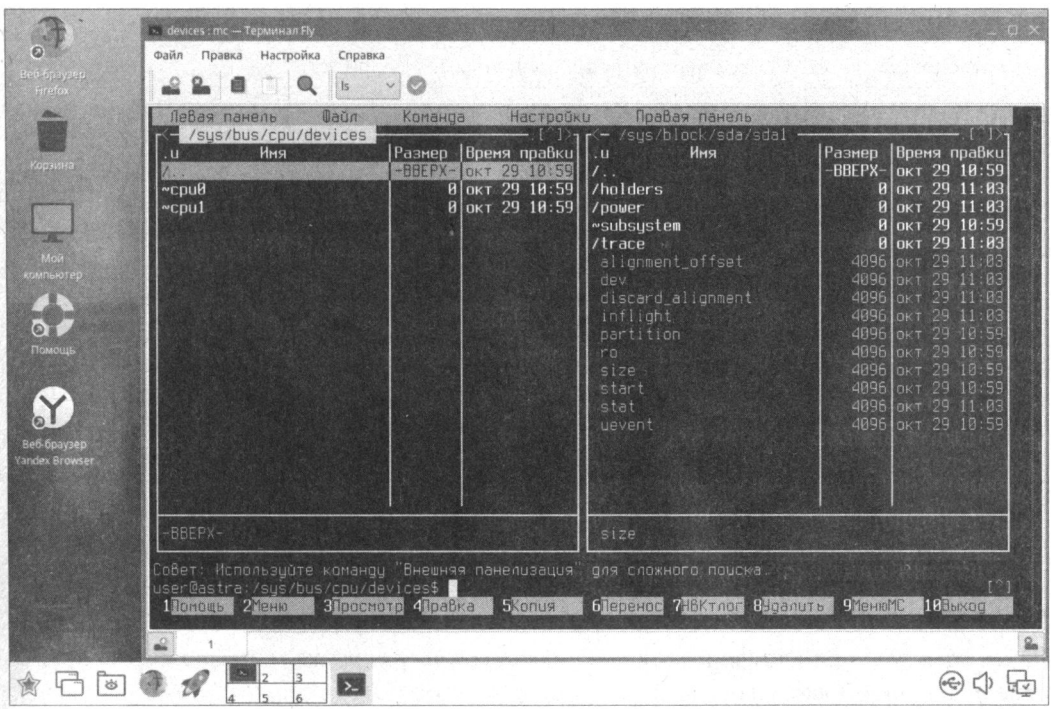

Рис. 19.4. Информация о количестве ядер процессора

19.3. Виртуальная файловая система proc

Виртуальная (псевдофайловая) система proc — это специальный механизм, позволяющий получать информацию о системе, ядре или процессе и изменять параметры ядра и его модулей на лету (кстати, ее название — это сокращение от process).

Файлы, позволяющие получать информацию, мы можем только просмотреть, а файлы, с помощью которых можно изменять некоторые параметры системы, — просмотреть и, если нужно, изменить.

Просмотреть информационный файл можно командой `cat`:

```
cat /proc/путь/<название_файла>
```

Записать значение в один из файлов proc можно так:

```
echo "данные" > /proc/путь/название_файла
```

19.3.1. Информационные файлы

В табл. 19.1 представлены некоторые (самые полезные) информационные proc-файлы — с их помощью вы можете получить информацию о системе.

Таблица 19.1. *Информационные proc-файлы*

Файл	Описание
/proc/version	Содержит версию ядра
/proc/cmdline	Список параметров, переданных ядру при загрузке
/proc/cpuinfo	Информация о процессоре
/proc/meminfo	Информация об использовании оперативной памяти (почти то же, что и команда `free`)
/proc/devices	Список устройств
/proc/filesystems	Файловые системы, которые поддерживаются системой
/proc/mounts	Список подмонтированных файловых систем
/proc/partitions	Список разделов (блочных устройств)
/proc/modules	Список загруженных модулей
/proc/swaps	Список разделов и файлов подкачки, которые активны в текущий момент

Файл /proc/cpuinfo (рис. 19.5) предоставляет исчерпывающую информацию о процессоре: вендор, поколение, модель, частота (cpu MHz), размер кеша (cache size), количество ядер (cpu cores).

Ранее мы получали информацию об имеющихся разделах и их размерах посредством файловой системы /sys, но гораздо проще это делать с помощью файла /proc/partitions (рис. 19.6).

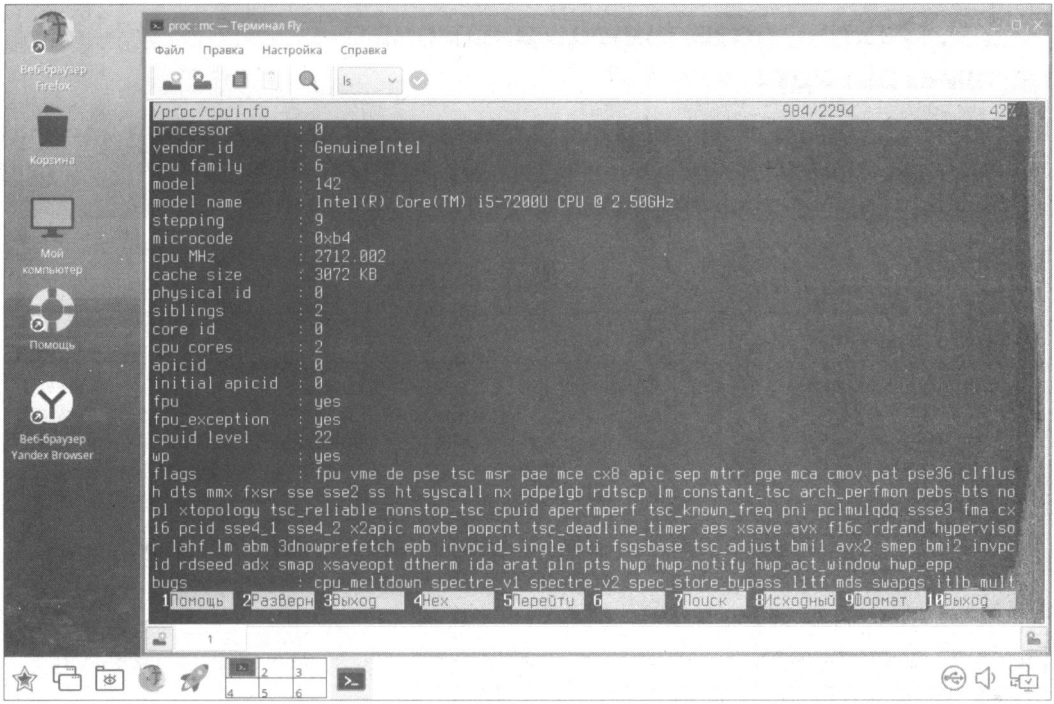

Рис. 19.5. Информация о процессоре

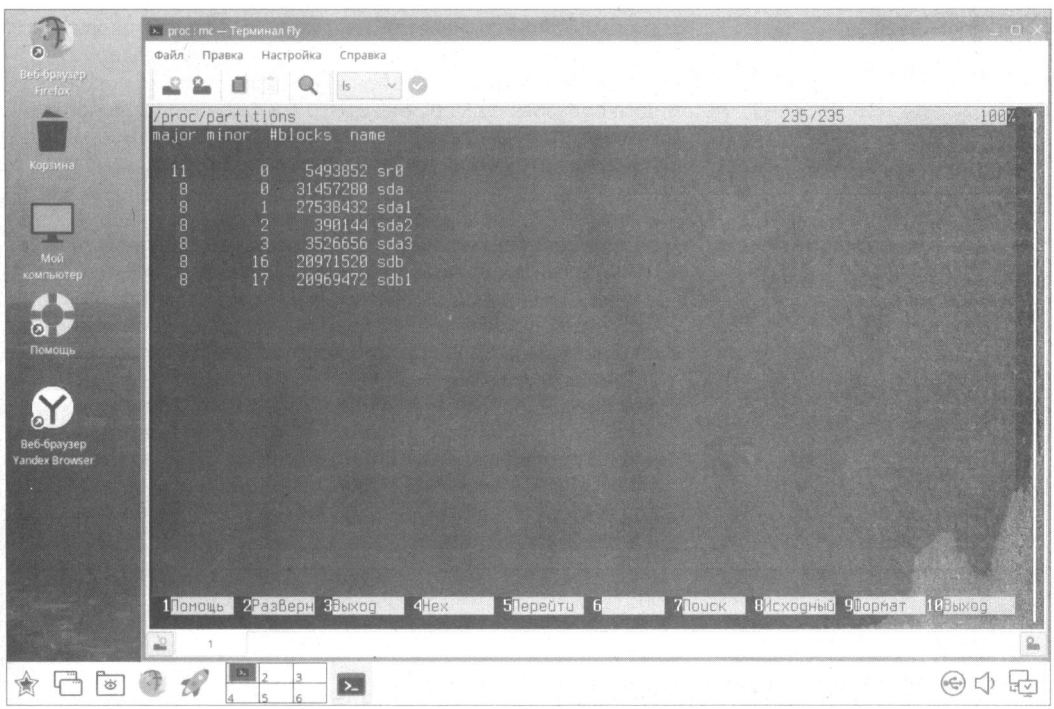

Рис. 19.6. Информация об имеющихся разделах

19.2.2. Файлы, позволяющие изменять параметры ядра

Каталог /proc/sys/kernel содержит файлы, с помощью которых вы можете изменять важные параметры ядра. Конечно, все файлы мы рассматривать здесь не будем, а остановимся лишь на тех, которые используются на практике (табл. 19.2).

Таблица 19.2. *Файлы каталога /proc/sys/kernel*

Файл	Каталог
/proc/sys/kernel/ctrl-alt-del	Если этот файл содержит значение 0, то при нажатии клавиатурной комбинации <Ctrl>+<Alt>+<Del> будет выполнена так называемая мягкая перезагрузка, когда управление передается программе системы инициализации, и последняя «разгружает» систему, как при вводе команды `reboot`. Если этот файл содержит значение 1, то нажатие <Ctrl>+<Alt>+<Del> равносильно нажатию кнопки Reset. Сами понимаете, значение 1 устанавливать не рекомендуется
/proc/sys/kernel/domainname	Здесь находится имя домена — например, **example.com**
/proc/sys/kernel/hostname	Содержит имя компьютера, например server
/proc/sys/kernel/panic	При критической ошибке ядро «впадает в панику» — работа системы останавливается, а на экран выводится надпись **`kernel panic`** и текст ошибки. Указанный файл содержит значение в секундах, в течение которого система подождет, пока пользователь прочитает это сообщение, после чего компьютер будет перезагружен. Значение 0 (по умолчанию) означает, что перезагружать компьютер вообще не нужно
/proc/sys/kernel/printk	Этот файл позволяет определить важность сообщения об ошибках. По умолчанию файл содержит значения 6, 4, 1 и 7. Это означает, что сообщения с уровнем приоритета 6 и ниже (чем ниже уровень, тем выше важность сообщения) будут выводиться на консоль. Для некоторых сообщений об ошибках уровень приоритета не задается. Тогда нужно установить уровень по умолчанию. Это как раз и есть второе значение — 4. Третье значение — это номер самого максимального приоритета, а последнее значение задает значение по умолчанию для первого значения. Обычно изменяют только первое значение, чтобы определить, какие значения должны быть выведены на консоль, а какие — попасть в журнал демона syslog

19.2.3. Файлы, изменяющие параметры сети

В каталоге /proc/sys/net вы найдете файлы, изменяющие параметры сети (табл. 19.3).

Таблица 19.3. Файлы каталога /proc/sys/net

Файл	Описание
/proc/sys/net/core/message_burst	Опытные системные администраторы используют этот файл для защиты от атак на отказ (DoS). Один из примеров DoS-атаки — когда система заваливается сообщениями атакующего, а полезные сообщения системой игнорируются, потому что она не успевает реагировать на сообщения злоумышленника. В указанном файле содержится значение времени (в десятых долях секунды), необходимое для принятия следующего сообщения. Значение по умолчанию — 50 (5 секунд). Сообщение, попавшее в «перерыв» (в эти 5 секунд), будет проигнорировано
/proc/sys/net/core/message_cost	Чем выше значение в этом файле, тем больше сообщений будут проигнорированы в перерыв, заданный файлом message_burst
/proc/sys/net/core/netdev_max_backlog	Задает максимальное число пакетов в очереди. По умолчанию 300. Используется, если сетевой интерфейс передает пакеты быстрее, чем система может их обработать
/proc/sys/net/core/optmem_max	Задает максимальный размер буфера для одного сокета

19.3.4. Файлы, изменяющие параметры виртуальной памяти

В каталоге /proc/sys/vm вы найдете файлы, с помощью которых можно изменить параметры виртуальной памяти:

- в файле buffermem находятся три значения (разделяются пробелами): минимальный, средний и максимальный объем памяти, которую система может использовать для буфера. Значения по умолчанию: 2 10 60;
- в файле kswapd тоже есть три значения, которые можно использовать для управления подкачкой:
  - первое значение задает максимальное количество страниц, которые ядро будет пытаться переместить на жесткий диск за один раз;
  - второе значение — минимальное количество попыток освобождения той или иной страницы памяти;
  - третье значение задает количество страниц, которые можно записать за один раз. Значения по умолчанию: 512 32 8.

19.3.5. Файлы, позволяющие изменить параметры файловых систем

Каталог /proc/sys/fs содержит файлы, изменяющие параметры файловых систем. В частности:

- файл file-max задает максимальное количество одновременно открытых файлов (по умолчанию 4096);
- в файле inode-max содержится максимальное количество одновременно открытых индексных дескрипторов — *инодов* (максимальное значение также равно 4096);
- в файле super-max находится максимальное количество используемых суперблоков;

> ***Ограничение на количество подмонтируемых файловых систем***
>
> Поскольку каждая файловая система имеет свой суперблок, легко догадаться, что количество подмонтируемых файловых систем не может превысить значение из файла *super-max*, которое по умолчанию равно 256, чего в большинстве случаев вполне достаточно. Наоборот, можно уменьшить это значение, чтобы никто не мог подмонтировать больше файловых систем, чем нужно (если монтирование файловых систем разрешено обычным пользователям).

- в файле super-nr находится количество открытых суперблоков в текущий момент. В этот файл нельзя записывать, его можно только читать.

19.4. Сохранение произведенных изменений

Итак, вы изменили некоторые параметры системы с помощью файлов каталога /proc. Чтобы сохранить измененные параметры, их следует прописать в файле /etc/sysctl.conf. Вот только формат этого файла следующий: надо отбросить /proc/sys/ в начале имени файла, все, что останется, записать через точку, а затем через знак равенства указать значение параметра. Например, для изменения параметра /proc/sys/vm/buffermem нужно в файле /etc/sysctl.conf прописать строку:

```
vm.buffermem = 2 11 60
```

Если в вашем дистрибутиве нет файла /etc/sysctl.conf, тогда пропишите команды вида `echo "значение" > файл` в сценарий инициализации системы.

ГЛАВА 20

ASTRA LINUX®

Управление устройствами

- Команды для получения информации об устройствах
- Управление модулями ядра
- Подключение принтера

20.1. Команды для получения информации об устройствах

Следующие команды позволяют получить информацию о подключенных устройствах:

- `lsusb` — выводит список USB-устройств;
- `lspci` — выводит список PCI-устройств;
- `lsscsi` — возвращает список SCSI-устройств;
- `lsblk` — выводит список блочных устройств (разделы накопителей, флешки, CD/DVD-приводы);
- `fdisk` — может использоваться, как для изменения таблицы разделов, так и для вывода информации о ней (`fdisk -l`);
- `lscpu` — выводит информацию о процессоре и его функциях;
- `lshw` — выводит информацию о некоторых аппаратных компонентах (процессоре, памяти, контроллерах USB, сетевых адаптерах);
- `hwinfo` — чем-то похожа на `lshw`, но выводит больше информации;
- `hdparm` — выводит информацию о жестком диске.

Подключение устройства происходит примерно так же, как и в Windows:

1. Физическое подключение устройства.

   Если это не USB-устройство, то необходимо выключить компьютер перед его подключением, подключить устройство и затем уже включить компьютер и

устройство. С USB-устройствами все просто — нужно только подключить их к свободному разъему.

2. Проверить, распознала ли система устройство, можно командами из приведенного ранее списка.

   Например, увидеть список подключенных USB-устройств можно командой `lsusb`.

3. Если устройство распознано, можно приступить к его использованию.

   Так, если вы подключили веб-камеру, можно выбрать ее в настройках программы для работы с веб-камерой — например, в Skype.

4. Если устройство не распознано, вероятнее всего, причина в отсутствии модуля ядра для работы с этим устройством.

   Найдите информацию от вендора о подключении устройства — там будет указано, как называется нужный модуль ядра. Добавить модуль ядра можно командой `modprobe`. Если нужного модуля в вашей системе нет, необходимо скачать его с сайта вендора устройства.

Для получения информации об устройствах в Astra Linux можно также задействовать **Менеджер устройств** из группы **Оборудование** Панели управления (рис. 20.1). Вы можете использовать эту утилиту вместо команд `ls*` — кому как удобнее.

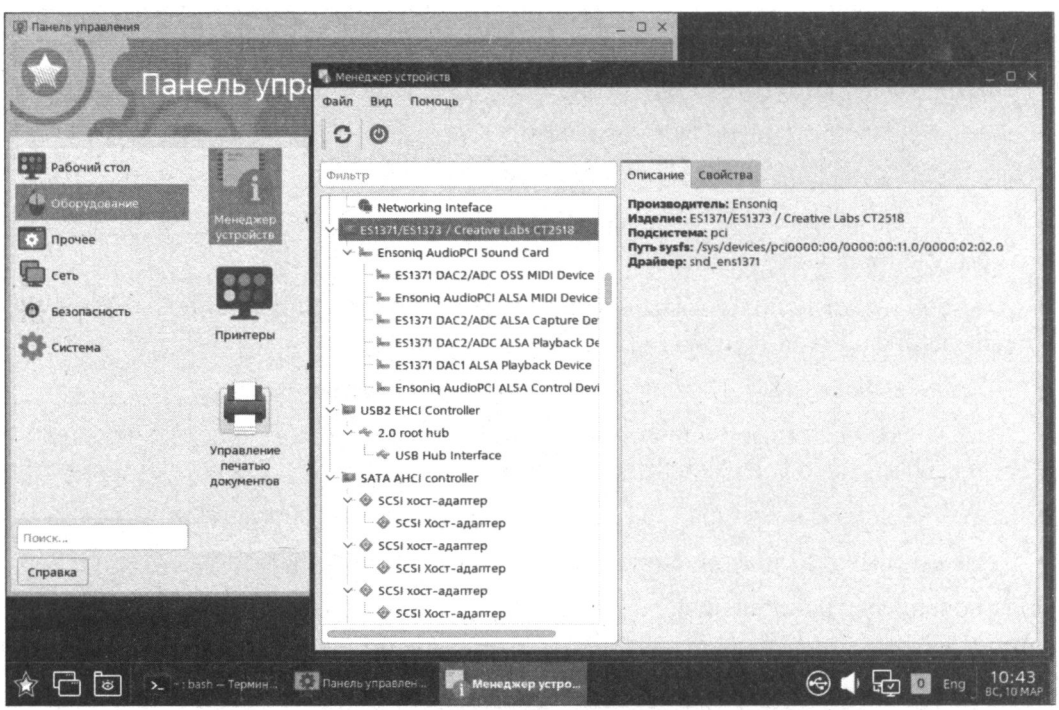

Рис. 20.1. Менеджер устройств

20.2. Управление модулями ядра

Просмотреть доступные модули ядра можно командой:

```
find /lib/modules/`uname -r` -name '*.ko'
```

Модули ядра хранятся в каталоге /lib/modules/<*версия ядра*>, а команда `uname -r` как раз возвращает версию ядра — чтобы вам не пришлось выяснять версию ядра, прежде чем просматривать каталог с модулями. Файлы модулей имеют «расширение» .ko. Список модулей довольно длинный (рис. 20.2), поэтому его можно фильтровать командой `grep`. Например, так можно вывести только модули звуковых устройств:

```
find /lib/modules/`uname -r` -name '*.ko' | grep sound
```

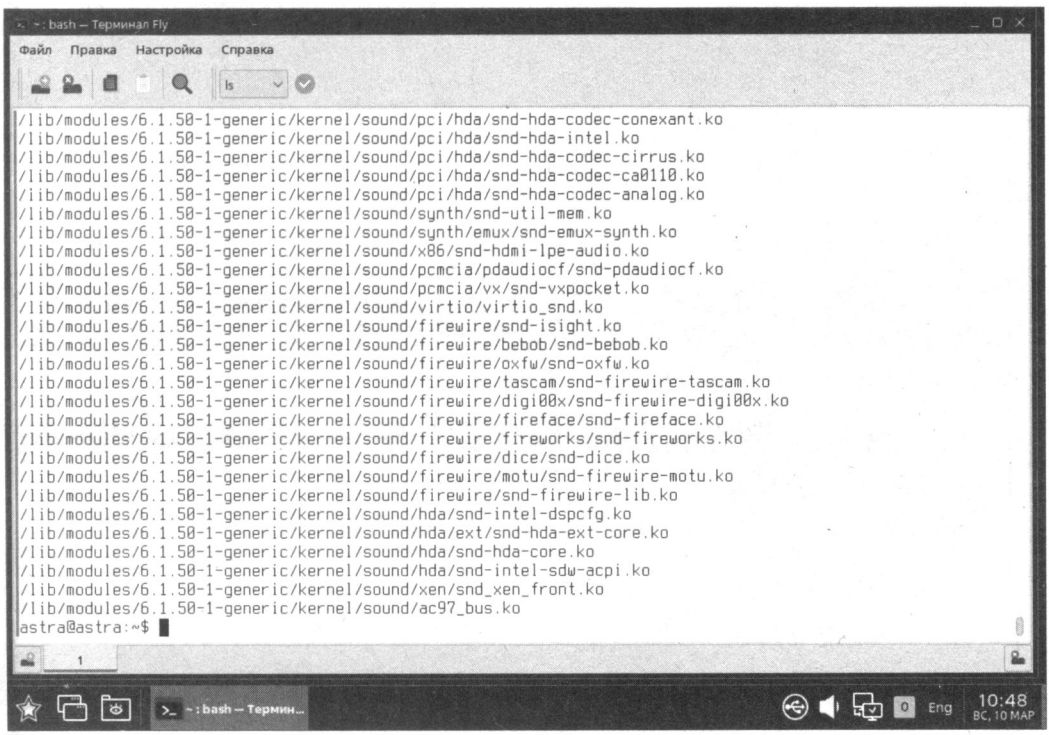

Рис. 20.2. Список модулей ядра

Получить информацию о модуле ядра можно командой `modinfo` (напомню, что для работы утилиты `modinfo` нужны права root):

```
sudo modinfo <название модуля без "расширения">
```

Посмотрите на вывод, показанный на рис. 20.3. Поле `depends` пусто, но для некоторых модулей здесь выводится список модулей, от которых зависит рассматриваемый модуль. Все эти модули должны быть также загружены.

Для загрузки модулей можно использовать команду `modprobe`:

```
sudo modprobe ac97_bus
```

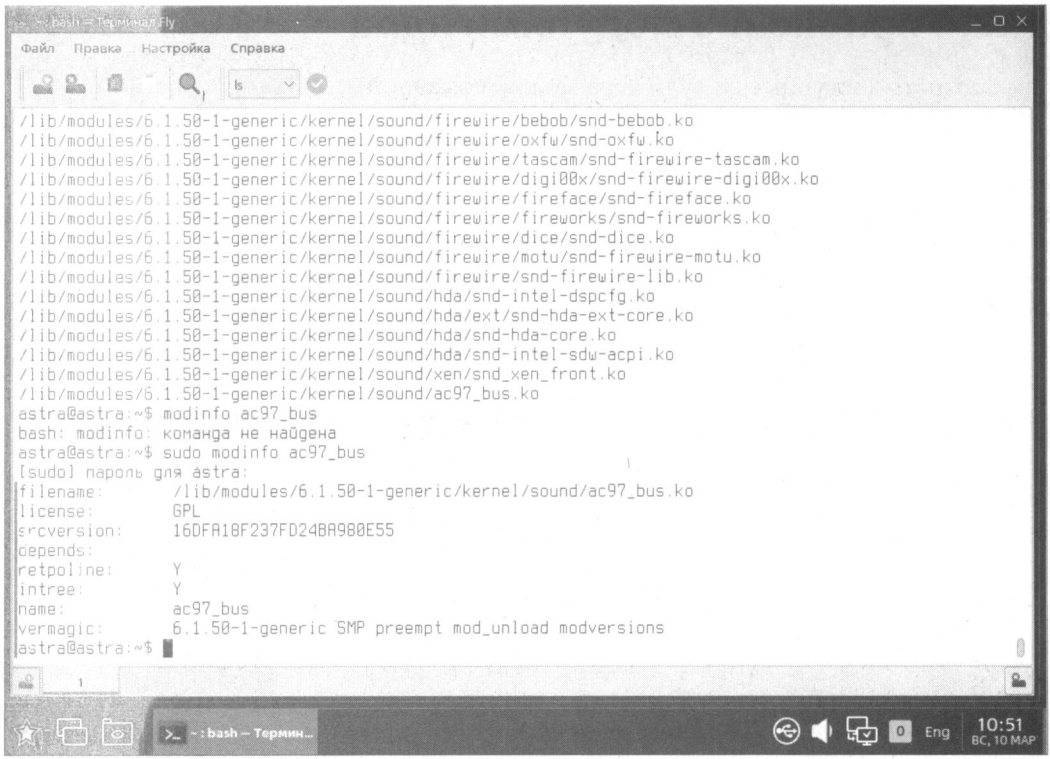

Рис. 20.3. Информация о модуле

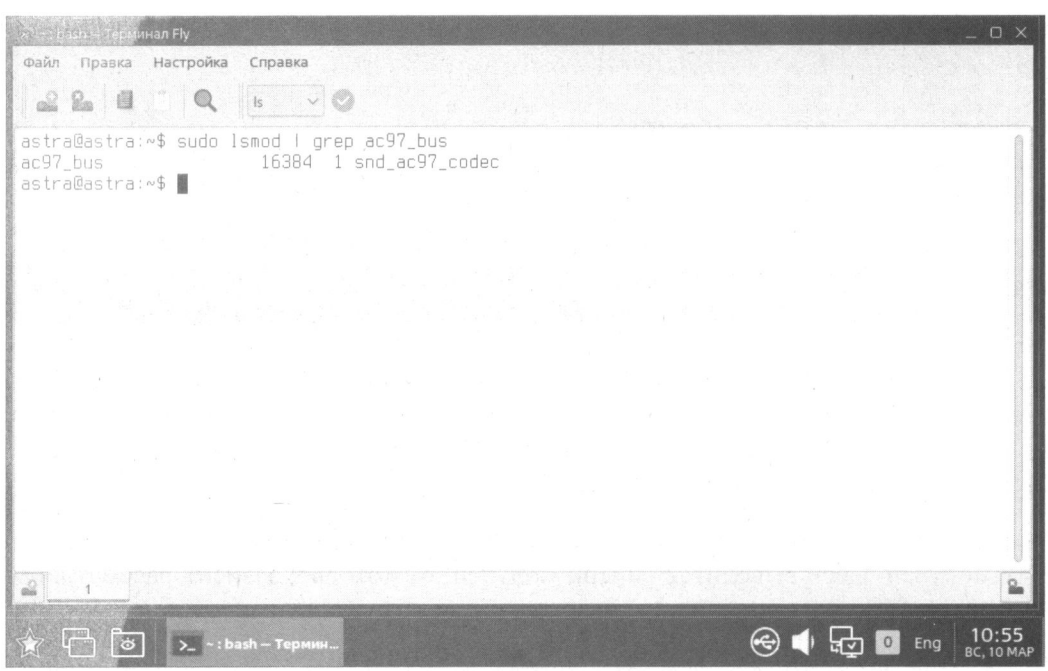

Рис. 20.4. Список загруженных модулей, отфильтрованный по строке `ac97_bus`

Просмотреть список загруженных модулей ядра можно командой `lsmod`:

```
sudo lsmod | grep <имя_модуля>
```

Например (рис. 20.4):

```
sudo lsmod | grep ac97_bus
```

Кроме команды `modpobe` можно также воспользоваться и командой `insmod`. Если `modprobe` позволяет добавить модуль, находящийся в соответствующем вашему ядру каталоге с модулями, то `insmod` позволяет вставить модуль, находящийся в любом другом каталоге. В качестве параметра ей нужно передать имя ko-файла:

```
sudo insmod /root/module.ko
```

20.3. Подключение принтера

20.3.1. Добавление принтера

Подключить принтер в Astra Linux несложно, но процесс полностью не автоматизирован, как в некоторых других дистрибутивах или в ОС Windows. Рассмотрим действия, необходимые для того, чтобы все заработало:

1. Откройте Панель управления, перейдите в раздел **Оборудование** (рис. 20.5) и выполните двойной щелчок на значке **Принтеры**.

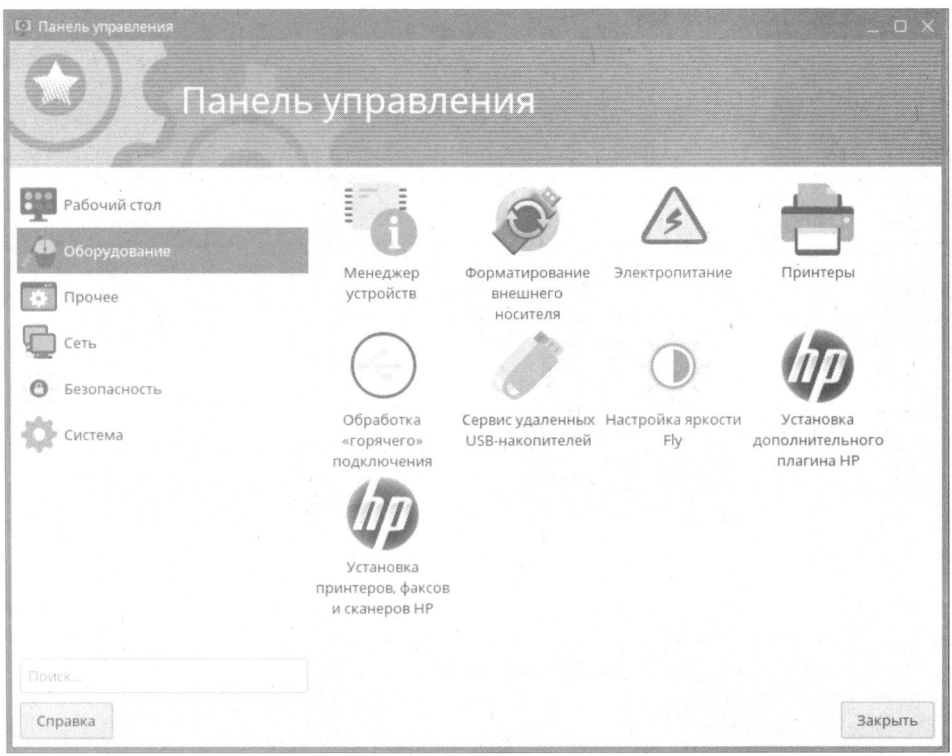

Рис. 20.5. Раздел **Панель управления | Оборудование**

2. В панели менеджера печати Fly (рис. 20.6) нажмите кнопку **Добавить** .
3. В открывшемся окне (рис. 20.7) выберите **Принтер**. Убедитесь, что принтер включен и подключен к компьютеру.
4. Система определит принтер, выберите его из списка (рис. 20.8) и нажмите кнопку **Далее**.
5. Рядом с полем **Драйвер** открывшейся панели мастера установки (рис. 20.9) нажмите на ссылку с троеточием и выберите драйвер принтера (рис. 20.10). Пока вы это не сделаете, добавить принтер будет невозможно.
6. Выбрав драйвер принтера (рис. 20.11), нажмите кнопку **Завершить**.
7. Добавленный принтер будет отображен в дереве устройств менеджера печати Fly. Если вы ошиблись с драйвером, его можно сменить на вкладке **Параметры** (рис. 20.12).

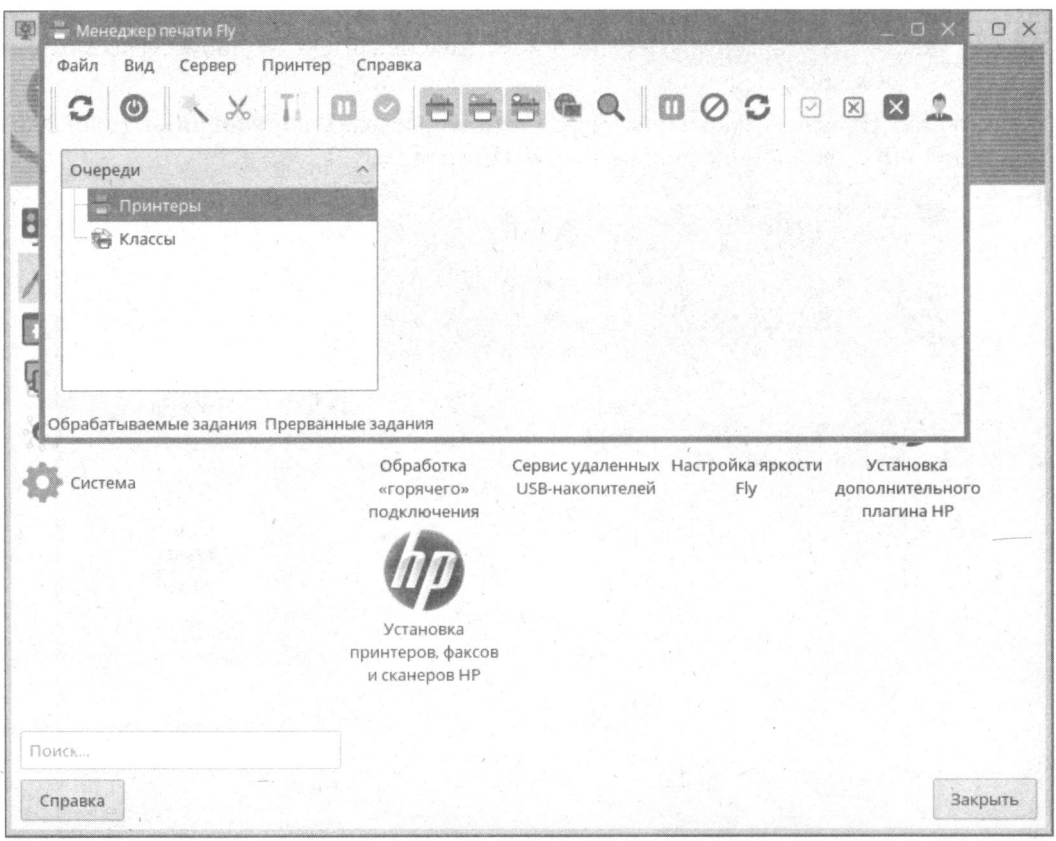

Рис. 20.6. Менеджер печати Fly

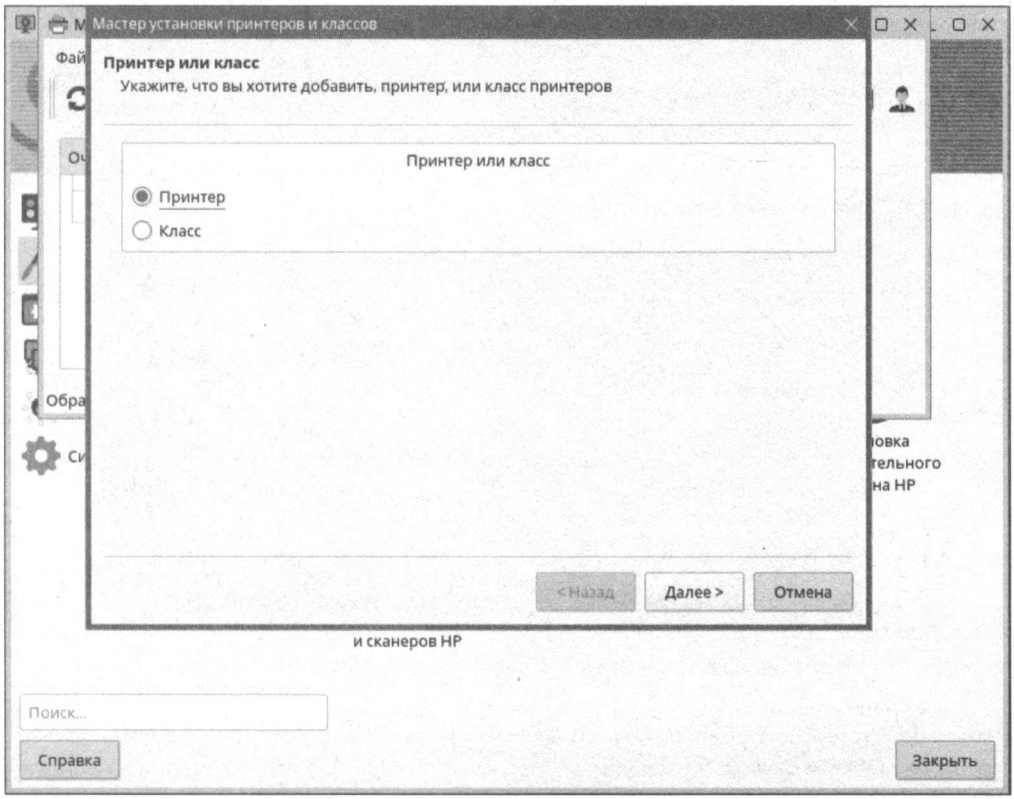

Рис. 20.7. Выберите Принтер

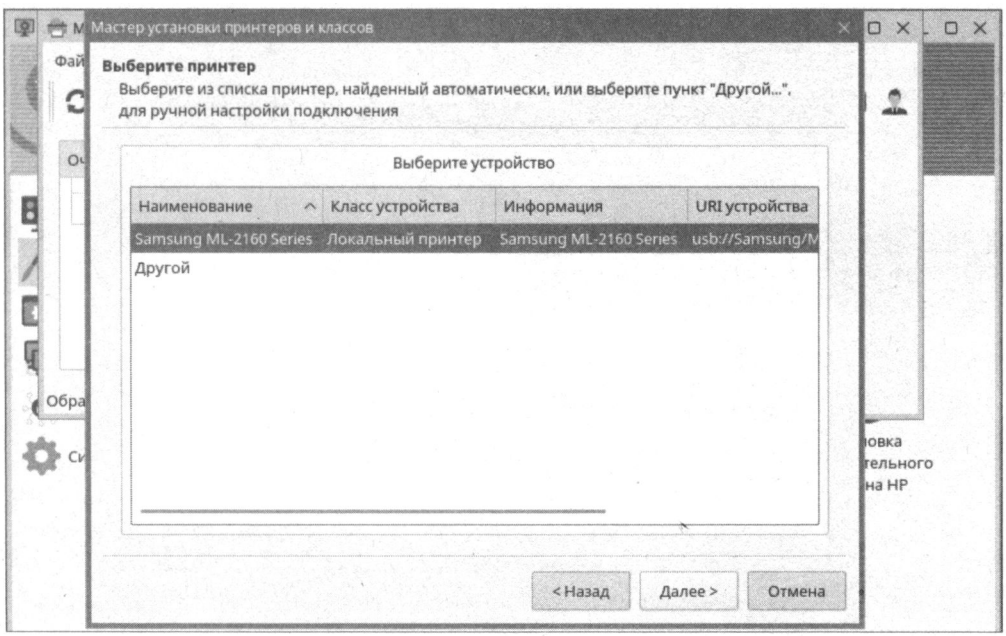

Рис. 20.8. Выберите принтер и нажмите кнопку **Далее**

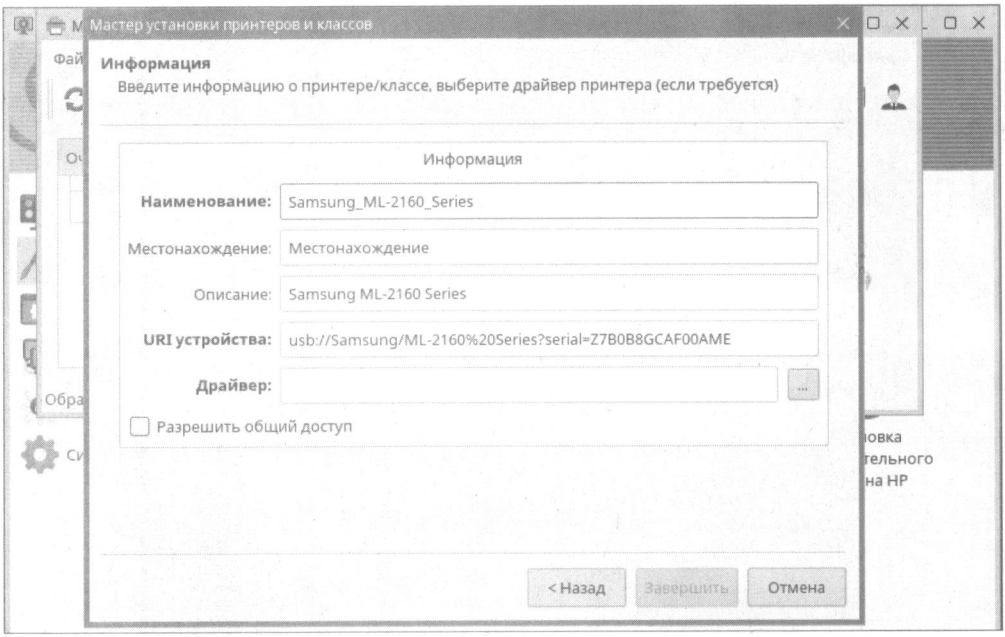

Рис. 20.9. Нажмите на ссылку с троеточием рядом с полем **Драйвер**

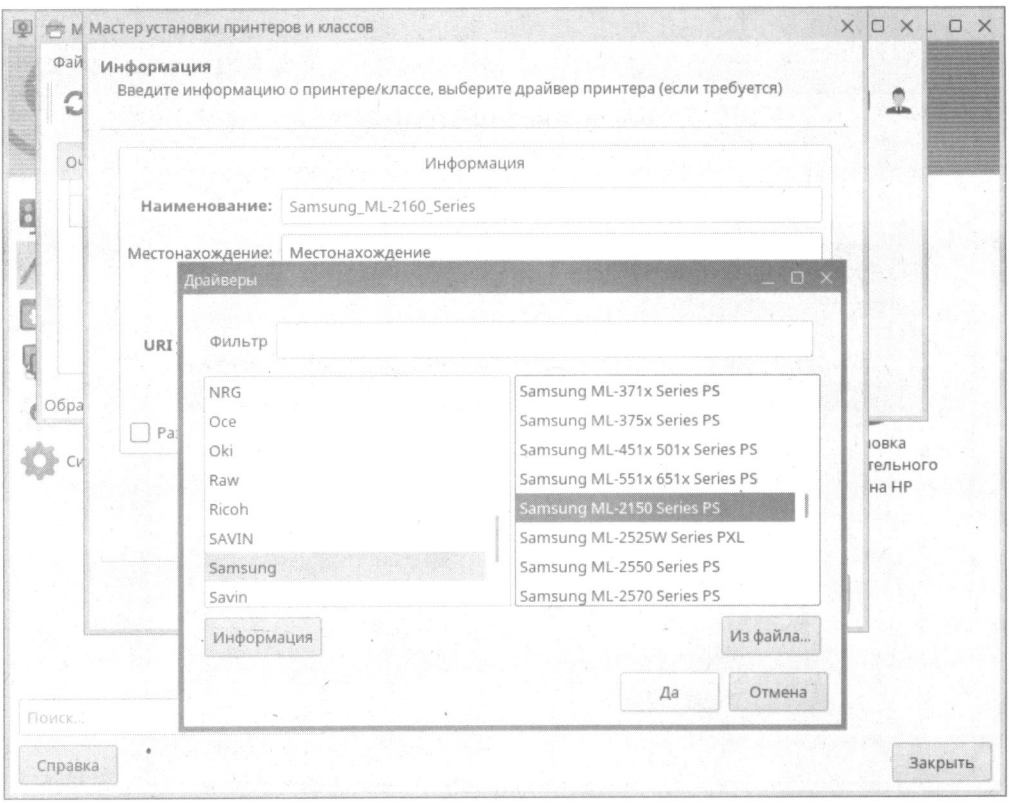

Рис. 20.10. Выберите драйвер принтера

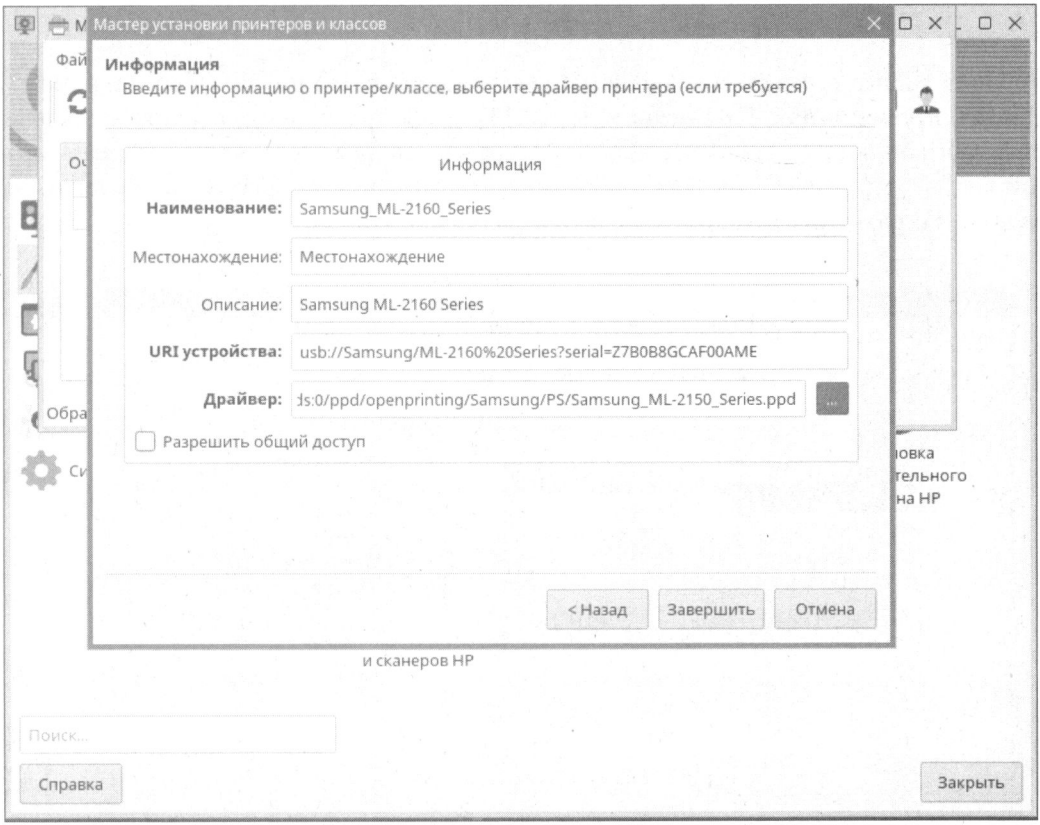

Рис. 20.11. Нажмите кнопку **Завершить**

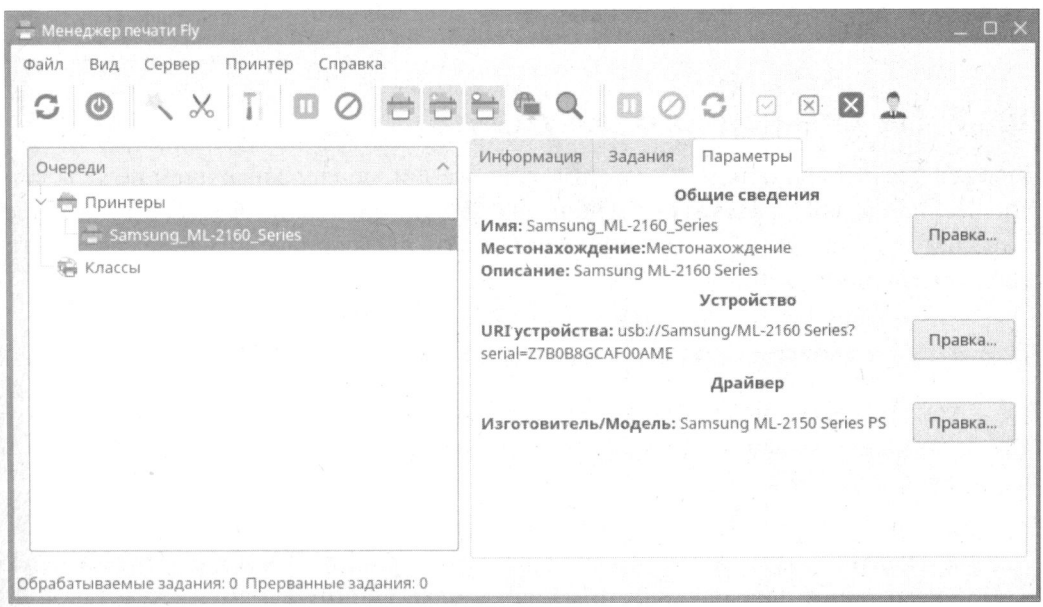

Рис. 20.12. Принтер добавлен

20.3.2. Печать пробной страницы

Добавив принтер, щелкните на нем правой кнопкой мыши и выберите команду **Печать проверочной страницы** (рис. 20.13). Если проверочная страница распечаталась без ошибок — тогда все хорошо. Если нет, скорее всего, нужно выбрать другой драйвер принтера. Это, как отмечено в предыдущем разделе, можно сделать на вкладке **Параметры** менеджера печати Fly (см. рис. 20.12).

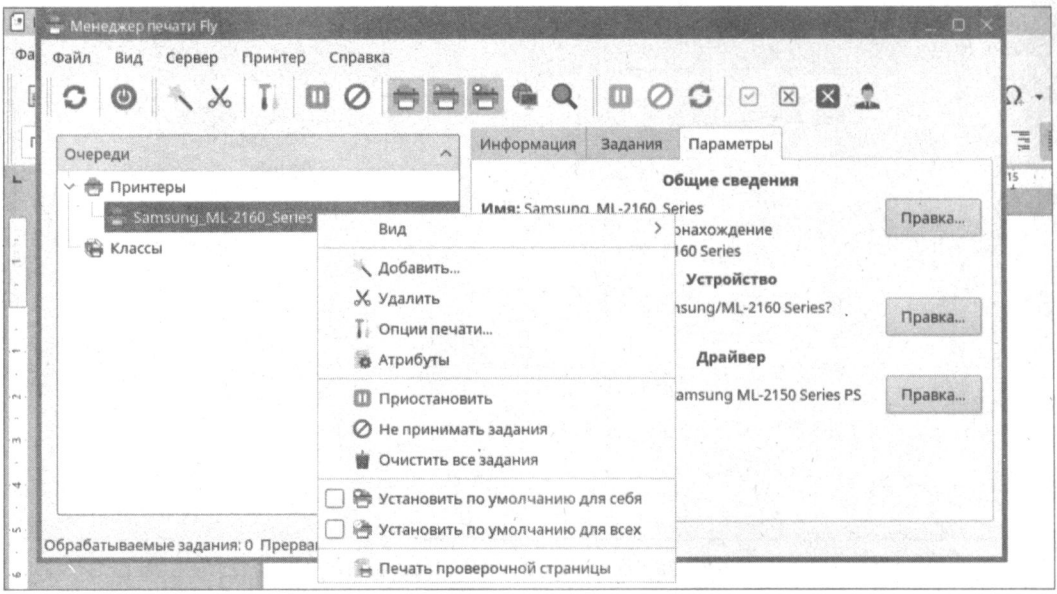

Рис. 20.13. Печать проверочной страницы

20.3.3. Печать первого документа

Здесь все достаточно просто. В каждом приложении, поддерживающем печать, есть одноименная команда **Печать**, обычно она находится в меню **Файл**. Эта команда открывает окно печати, в котором нужно выбрать свой принтер из списка и нажать кнопку **Печать** (рис. 20.14).

20.3.4. Установка опций печати

При желании можно включить различные параметры печати — например, включить двустороннюю печать, если принтер ее поддерживает, задать размер страницы (по умолчанию — A4) и т. д.

Для редактирования опций печати надо выбрать свой принтер из списка в окне менеджера печати Fly (см. рис. 20.13) и выполнить команду **Принтер | Опции печати**. В открывшейся панели (рис. 20.15) на вкладке **Общие** вы сможете установить опции печати: задать ориентацию страницы, определить отступы и пр.

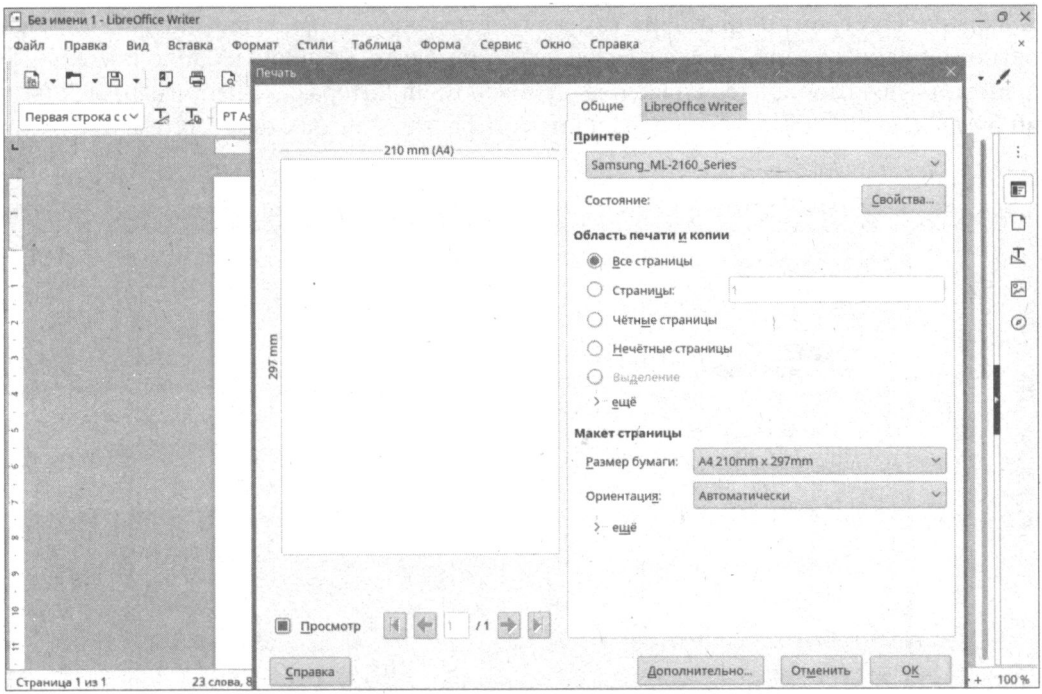

Рис. 20.14. Окно печати в LibreOffice Writer

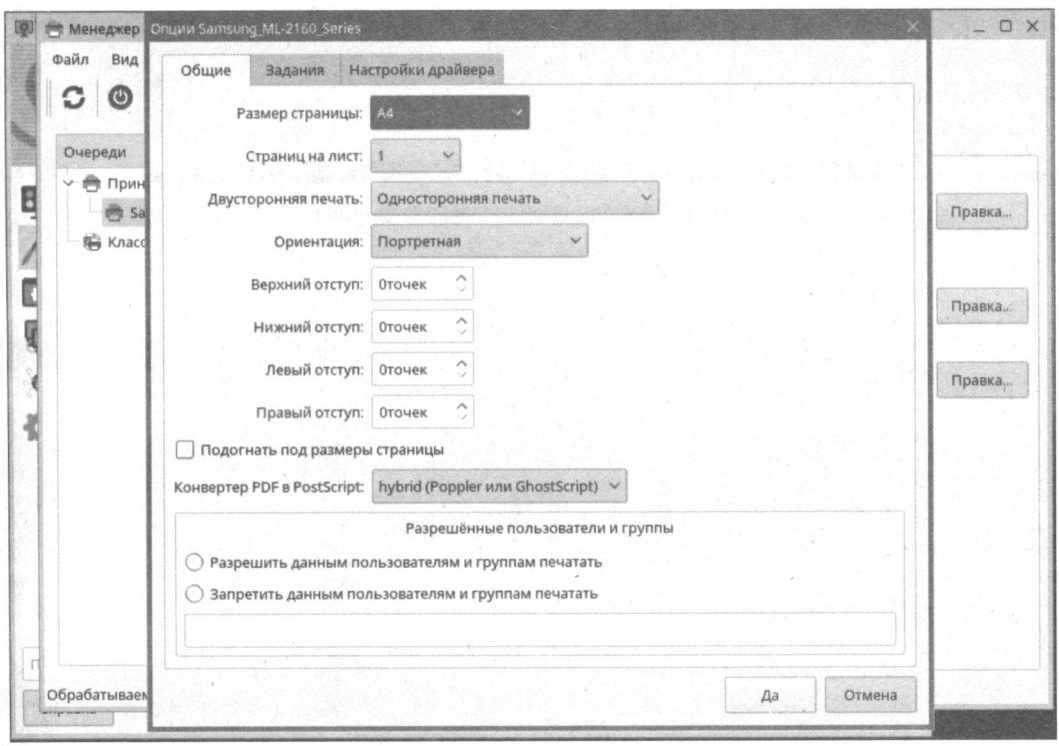

Рис. 20.15. Опции печати

На вкладке **Настройки драйвера** той же панели (рис. 20.16), в зависимости от конкретного драйвера принтера, вы можете установить различные дополнительные параметры печати, поддерживаемые драйвером принтера, — например, выбрать тип бумаги, включить режим сохранения тонера, задать качество печати и др.

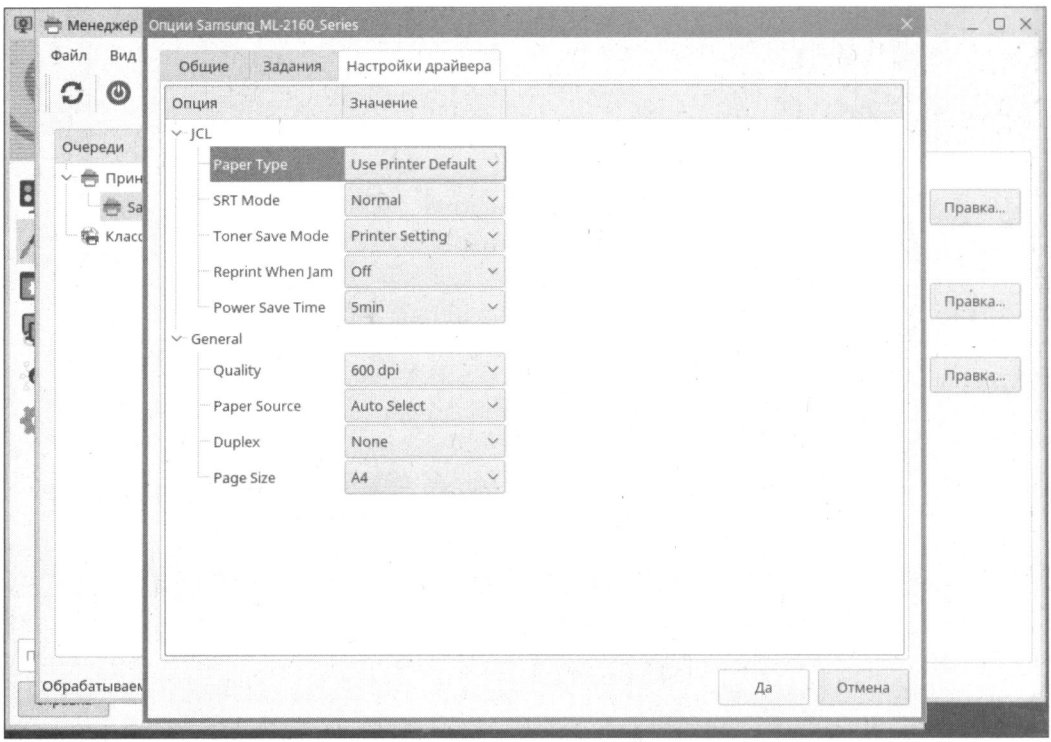

Рис. 20.16. Настройки драйвера принтера

ASTRA LINUX®

ГЛАВА 21

Настройка жесткого диска

- Физическое подключение жесткого диска
- Разметка жесткого диска
- Монтирование новых разделов

21.1. Физическое подключение жесткого диска

Давно прошли те времени, когда вам нужно было не только подключить жесткий диск к материнской плате, но еще и правильно выставить перемычки на нем, чтобы все заработало. Интерфейс IDE давно не используется. Все современные жесткие диски подключаются посредством интерфейса SATA. Исключение составляют только серверные диски и диски для ноутбуков. Но в случае с ноутбуком все немного не так — и о нем отдельно чуть позже.

> ***SSD — ТОЖЕ «ЖЕСТКИЙ ДИСК»***
>
> Под «жестким диском» следует также понимать и SSD — последовательность действий по его подключению аналогичная.

Последовательность действий по физическому подключению дополнительного жесткого диска к стационарному компьютеру такова:

1. Полностью отключите питание компьютера. Не переводите его в сон, а используйте именно выключение питания. Для надежности физически отключите его из розетки.
2. Найдите свободный SATA-разъем на материнской плате — он такой же, к которому подключен уже имеющийся жесткий диск. Подключите к нему жесткий диск с помощью информационного шлейфа, а затем подключите к жесткому диску разъем питания.
3. Установите жесткий диск в корпус компьютера и надежно зафиксируйте его.
4. Закройте корпус, подключение питание и включите компьютер.

Лучше начать с подключения...

Действия 2 и 3 вы можете поменять местами — как вам будет удобно. Лично мне удобно сначала подключить жесткий диск, а затем искать место для его установки. В противном случае если место установки диска выбрано неправильно, то длины шлейфа или кабеля питания может не хватить.

С ноутбуком все не так просто. В большинстве случаев у вас не будет в нем места для установки еще одного накопителя. Исключения составляют некоторые игровые ноутбуки или уже устаревшие модели, из которых можно извлечь DVD-привод и вместо него установить «карман» для жесткого диска. Такую операцию лучше производить в сервисном центре, чтобы ничего не сломать. В книге я описывать ее не стану, поскольку процедуры разборки ноутбука сильно различаются в зависимости от производителя и модели. Но если уж вы решили все сделать самостоятельно, найдите инструкции в Интернете — для распространенных моделей, как правило, есть пошаговые инструкции с картинками.

Еще о разборке ноутбука

Для разборки ноутбука может понадобиться специальный набор инструментов, который стоит примерно столько же, сколько и установка «кармана» в сервисном центре. Так что если вы не собираетесь часто разбирать и собирать ваш ноутбук, проще обратиться к специалистам.

Поэтому в большинстве случаев к ноутбуку вы будете подключать внешний жесткий диск, выполненный в формате USB-устройства, и процедура такого подключения ничем не отличается от подсоединения обычной флешки. Если же вы собираетесь подключить к ноутбуку обычный жесткий диск, это можно сделать с помощью адаптера SATA-USB, который стоит сущие копейки. Понятно, что выключать ноутбук при подключении внешнего жесткого диска не нужно.

Далее все зависит от того, подключаете ли вы новый жесткий диск или же диск, на котором уже есть таблица разделов. В первом случае нужно выполнить его разметку, о чем рассказано в *разд. 21.2*, а во втором — только подмонтировать нужный раздел диска, следуя указаниям, приведенным в *разд. 21.3*.

21.2. Разметка жесткого диска

Опытные линуксоиды предпочитают использовать для разметки диска утилиту fdisk, но начинающим пользователям гораздо проще воспользоваться для этого программой GParted. Она обладает графическим интерфейсом и установлена в Astra Linux по умолчанию.

Для запуска программы GParted вызовите ее из главного меню: **Пуск | Системные | Редактор разделов GParted**. Программа отобразит таблицу разделов первого (имеющегося) диска — /dev/sda. Второй накопитель, который вы только что подключили, будет называться /dev/sdb — выберите его из списка в верхнем правом углу окна программы (рис. 21.1). Как можно здесь видеть, таблица разделов диска пуста — подключенный жесткий диск еще не был в использовании.

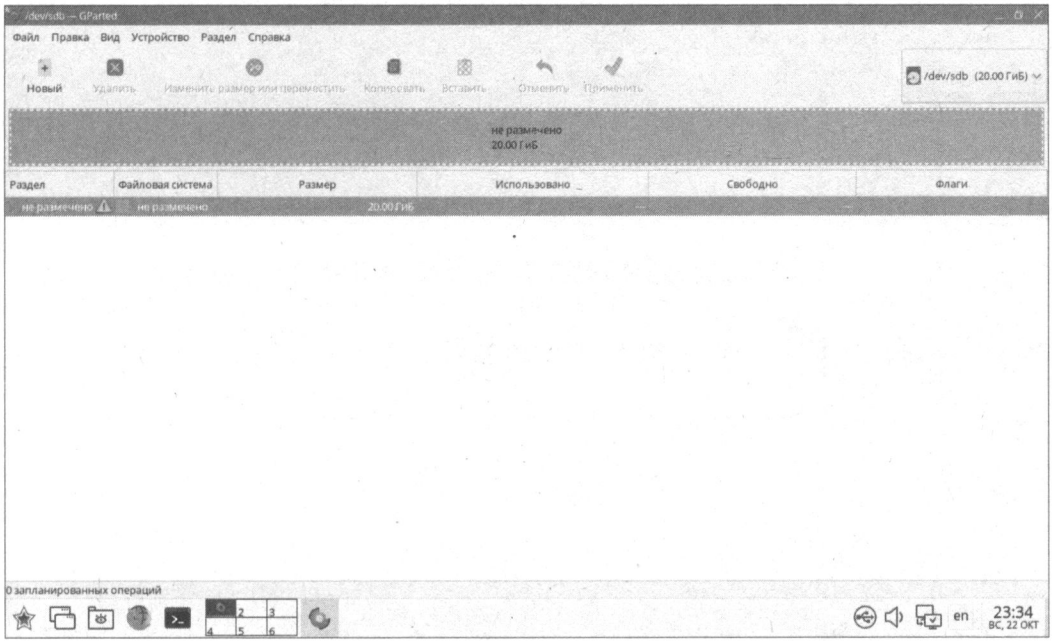

Рис. 21.1. Редактор разделов GParted

Теперь для разметки жесткого диска выполните следующие действия:

1. Выберите нужное устройство из списка в верхнем правом углу, если вы этого еще не сделали.
2. Выполните команду **Устройство | Создать таблицу разделов**.
3. В открывшемся окне (рис. 21.2) выберите тип таблицы разделов и нажмите кнопку **Применить**.

> ***Тип таблицы разделов: msdos или gpt?***
>
> Если не вникать в подробности, то таблица разделов msdos (она же mbr) не поддерживает файлы размером более 2,2 Тбайт, но зато она более совместима с имеющимся парком компьютеров, и все старые компьютеры смогут работать с таким диском. Тип разделов gpt более предпочтителен, но если вы собираетесь читать такой диск на Windows-компьютере, то для этого вам понадобится версия Windows не ниже Vista — предшествующие версии Windows не смогут работать с такой таблицей разделов. С другой стороны, если размер диска не превышает 2,2 Тбайт, нет смысла использовать gpt — этим вы только сузите круг использования диска, но преимуществ gpt не получите.

После создания таблицы разделов все доступное пространство диска будет помечено как **не размечено**.

4. Нажмите кнопку **Новый** для создания нового раздела.
5. В открывшемся окне (рис. 21.3) введите размер нового раздела и выберите тип файловой системы. По умолчанию программа предлагает использовать все доступное пространство. В принципе, если раздел диска не превышает 1 Тбайт, особого смысла создавать несколько разделов нет.

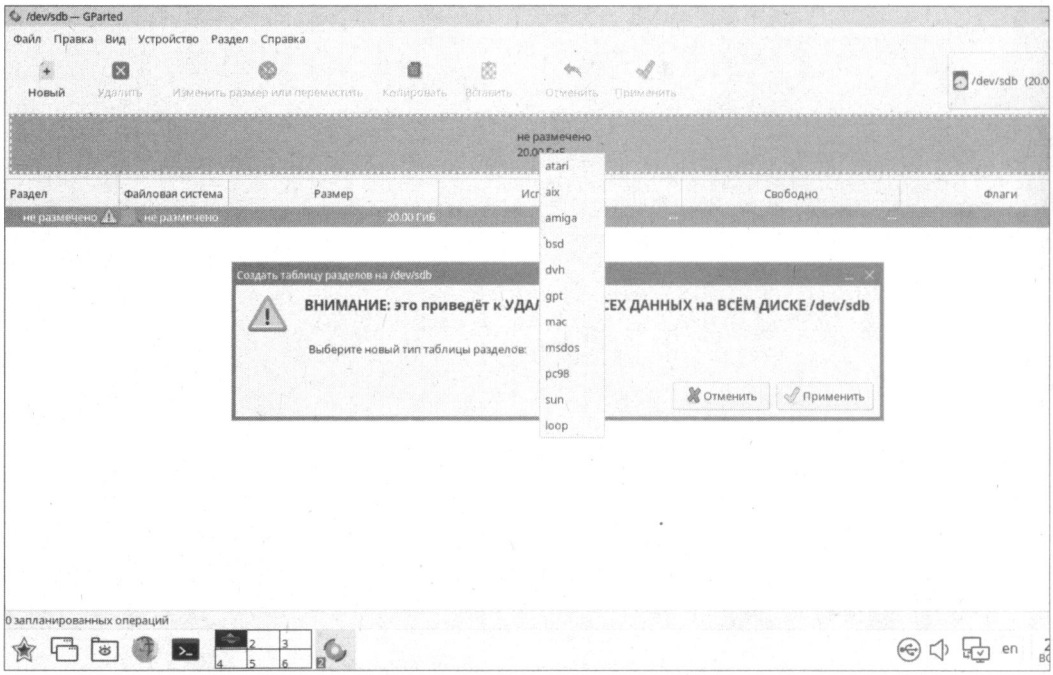

Рис. 21.2. Выбор типа таблицы разделов

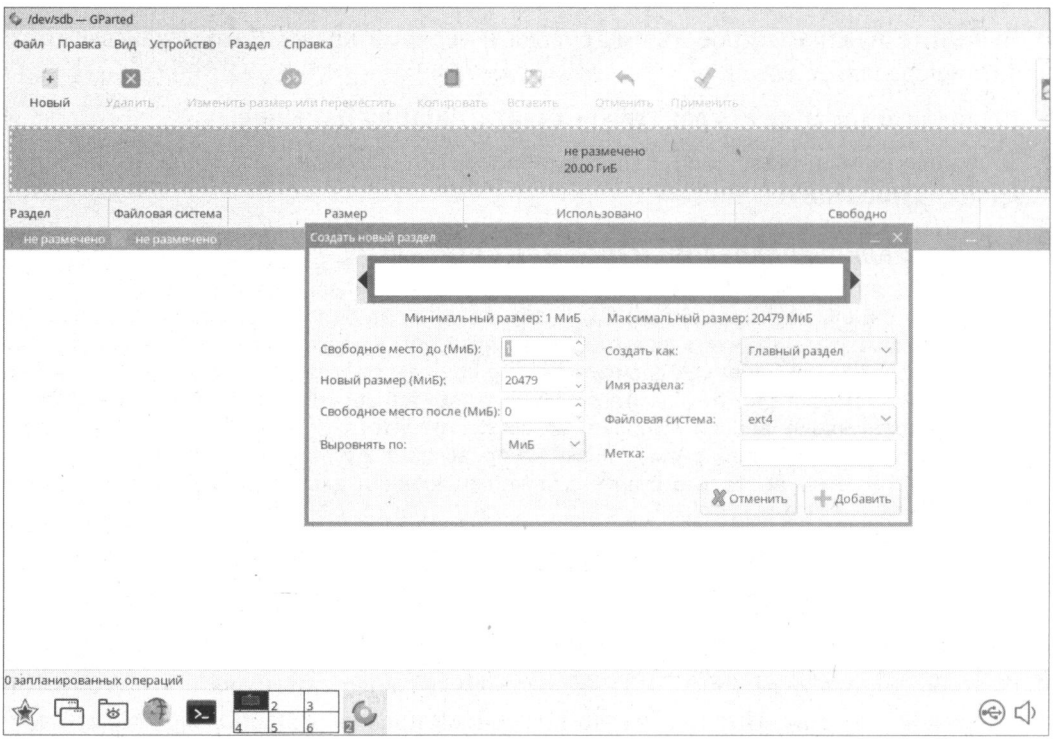

Рис. 21.3. Задание размера раздела и типа файловой системы

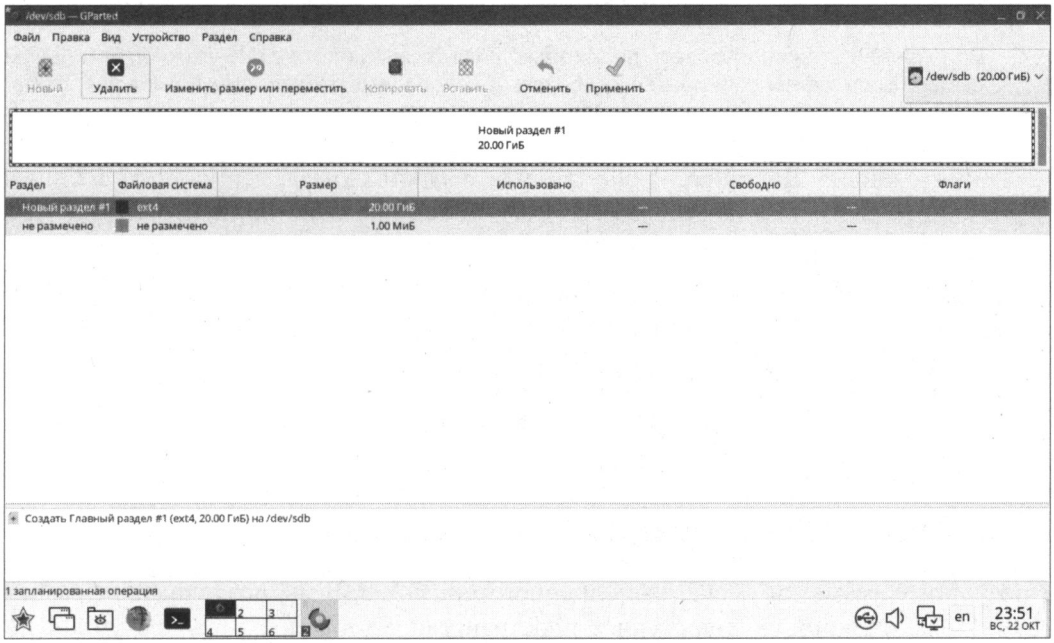

Рис. 21.4. Созданный раздел

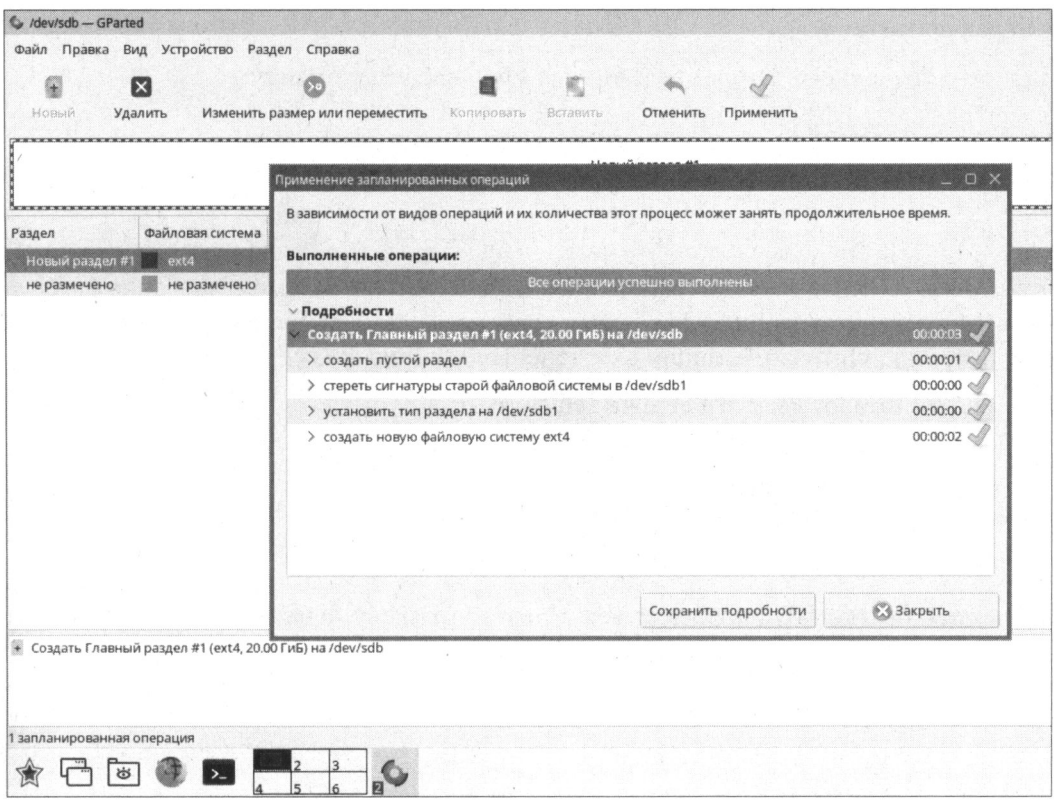

Рис. 21.5. Раздел создан — осталось нажать кнопку **Закрыть**

Тип файловой системы

Если вы будете использовать диск только в Linux, выбирайте ext4 — это оптимальный выбор. Если же вы собираетесь работать с создаваемым разделом в Windows, выберите ntfs.

6. Нажмите кнопку **Добавить** — вы увидите созданный раздел (рис. 21.4). Если вы создали раздел, не задействовав все доступное пространство, повторите шаги 5 и 6.
7. Нажмите кнопку **Применить** на панели инструментов. Для подтверждения операции нажмите кнопку **Применить** в открывшейся панели подтверждения.
8. Дождитесь окончания выполнения операций (рис. 21.5).
9. Закройте редактор разделов и перейдите к *разд. 21.3*.

21.3. Монтирование новых разделов

В результате разметки диска, выполненной в *разд. 23.2*, мы создали один раздел /dev/sdb1, занимающий все доступное пространство. Чтобы получить доступ к файловой системе этого раздела, нужно подмонтировать его к корневой файловой системе и обеспечить автоматическое монтирование (чтобы не монтировать его каждый раз при загрузке).

Начнем с последнего. Откройте терминал Fly и введите команду:

```
sudo kate /etc/fstab
```

Добавьте в открывшийся в редакторе файл etc/fstab строку:

```
/dev/sdb1      /mnt/sdb1      ext4          defaults              0       0
```

Эта строка обеспечивает автоматическое (при перезагрузке) монтирование раздела /dev/sdb1 к каталогу /mnt/sdb1 (файловая система — ext4, набор параметров — стандартный, дополнительные опции выключены: 0 0) (рис. 21.6).

Разумеется, каталог /mnt/sdb1 еще не существует, поэтому его нужно создать:

```
sudo mkdir /mnt/sdb1
```

Теперь можно или перезагрузить компьютер, или ввести команду:

```
sudo mount /dev/sdb1 /mnt/sdb1
```

После этого можно будет обратиться к новому разделу через каталог /mnt/sdb1. На рис. 21.7 видно, что диск пока пуст — на нем есть лишь каталог lost+found.

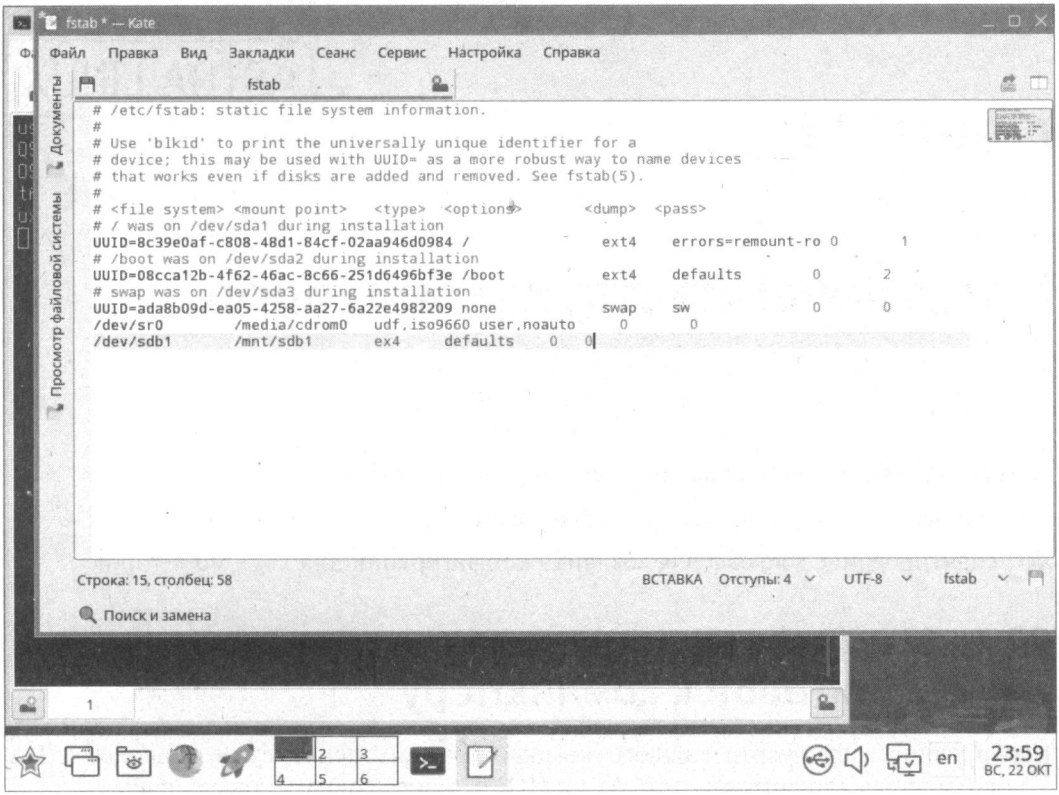

Рис. 21.6. Редактирование файла /etc/fstab

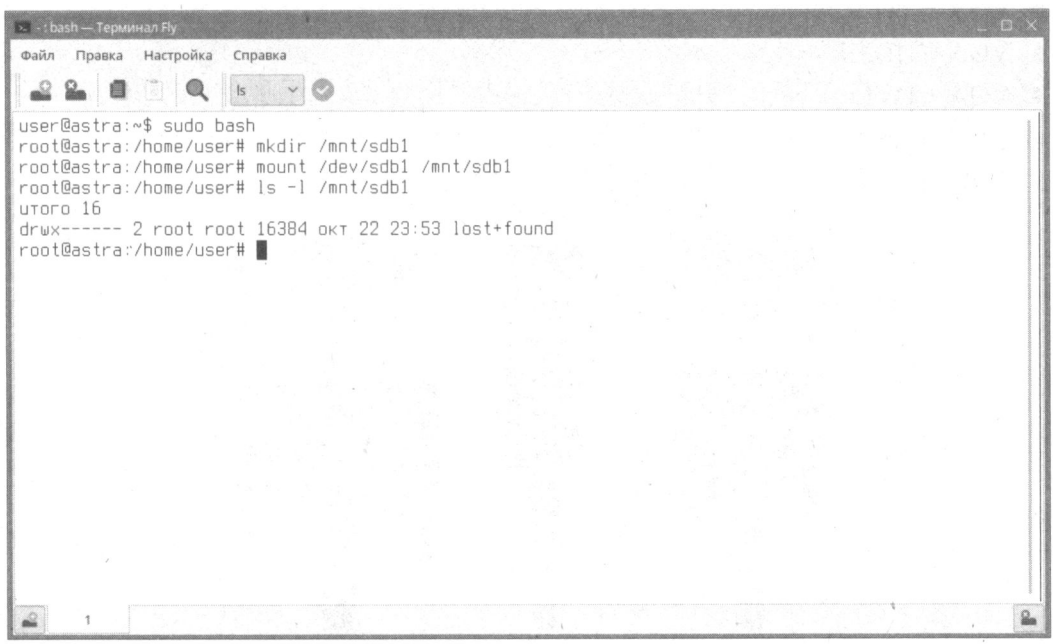

Рис. 21.7. Монтирование раздела

ASTRA LINUX®

ГЛАВА 22

Подключение двух мониторов

- Физическое подключение двух мониторов к компьютеру
- Получение эталонного файла конфигурации xorg.conf
- Редактирование xorg.conf для создания конфигурации для двух мониторов

22.1. Физическое подключение двух мониторов к компьютеру

Как правило, видеокарта каждого компьютера оснащена двумя видеовыходами. У современных вариантов — это DVI и HDMI, менее современные оснащены VGA-выходами (рис. 22.1). Комбинации могут быть разными:

- VGA + VGA;
- VGA + HDMI;
- VGA + DVI;
- DVI + DVI;
- DVI + HDMI.

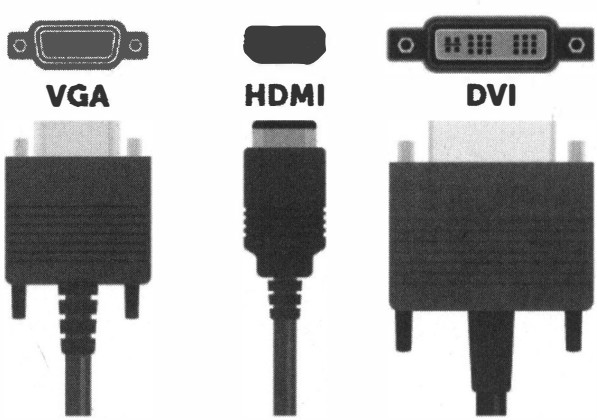

Рис. 22.1. Различные варианты видеовыходов компьютера

Конфигурация из трех мониторов

Используя ноутбук, можно с легкостью получить конфигурацию из трех мониторов — стандартный дисплей ноутбука и два монитора, подключенные к его видеовыходам (если у него их два).

Если всего один видеовыход...

Если у вашей видеокарты всего один видеовыход, то лучше всего поменять видеокарту на более современную. Другого простого решения не существует. Если вы купите какой-то разветвитель, позволяющий подключить несколько мониторов к одному видеовыходу, то получите несколько мониторов с одной и той же картинкой. Немного не то, что вы ожидали, правда? Конечно, в некоторых ситуациях (например, на демостендах) — это как раз то, что и нужно, но в большинстве случаев два монитора подключают для расширения рабочего стола.

Если ваши мониторы можно физически подключить к разъемам на видеокарте (например, у вас есть два DVI-монитора и на видеокарте имеются два DVI-выхода), тогда никаких проблем нет: подключаете мониторы, включаете компьютер и приступаете к настройке.

А как же DisplayPort?

Мы не забыли про DisplayPort. Это очень хороший интерфейс для цифровых мониторов. Он обладает вдвое большей пропускной способностью, чем DVI, и более низким напряжением питания. Но он почему-то не получил широкого распространения — до лета 2011 года в основном использовался на компьютерах Apple, а позже был заменен на интерфейс Thunderbolt — это дальнейшее развитие DisplayPort, причем Thunderbolt обратно совместим с DisplayPort. Но вряд ли кому-то придет в голову установить Astra Linux на MacBook...

А что делать, если у вас, например, два DVI-монитора, а видеовыходы видеокарты — DVI + HDMI? Тогда вам поможет переходник HDMI-DVI. Подобные переходники стоят недорого и есть в любых конфигурациях: HDMI-VGA (рис. 22.2), VGA-DVI и т. д.

Рис. 22.2. Переходник HDMI-VGA

22.2. Настройка системы

Итак, вы подключаете два монитора, загружаете компьютер, но в системе по-прежнему виден только один монитор. При этом в утилите настройки монитора имеется список, позволяющий выбрать главный видеовыход, но в нем нет вашего монитора. Придется, как в старые добрые времена, редактировать файл конфигурации xorg.conf вручную.

Все это не так просто, и если бы можно было решить проблему за пару кликов мышки, как в Windows, я бы здесь об этом сразу и написал.

Начнем с файла конфигурации графической подсистемы xorg.conf. По умолчанию он, к сожалению, в системе отсутствует. Причем вы не можете взять чей-то файл конфигурации — он просто не заработает на вашем компьютере, поэтому придется создавать свой. Как минимум вам нужна в нем секция `Device`, описывающая ваш драйвер видеокарты.

Чтобы создать файл xorg.conf, переключитесь на первую консоль (<Ctrl>+<Alt>+<F1>), войдите как root или как администратор и введите команду:

```
Xorg :1 -configure
```

```
        at http://wiki.x.org
 for help.
(EE)
root@astra:~# Xorg :1 -configure

X.Org X Server 1.20.14
X Protocol Version 11, Revision 0
Build Operating System: linux Debian
Current Operating System: Linux astra 5.15.0-70-generic #astra1+c13 SMP Fri Mar 31 15:57:55 UTC 2023 x86_64
Kernel command line: BOOT_IMAGE=/vmlinuz-5.15.0-70-generic root=UUID=8c39e0af-c808-48d1-84cf-02aa946d0984 ro quiet net.ifnames=0
Build Date: 07 February 2023  05:51:43PM
xorg-server 2:1.20.14-1ubuntu1astra17 (For technical support please see http://www.ubuntu.com/support)
Current version of pixman: 0.40.0
        Before reporting problems, check http://wiki.x.org
        to make sure that you have the latest version.
Markers: (--) probed, (**) from config file, (==) default setting,
        (++) from command line, (!!) notice, (II) informational,
        (WW) warning, (EE) error, (NI) not implemented, (??) unknown.
(==) Log file: "/var/log/Xorg.1.log", Time: Mon Oct 23 21:39:19 2023
List of video drivers:
        amdgpu
        ast
        ati
        intel
        mach64
        mga
        nouveau
        r128
        radeon
        vmware
        modesetting
        fbdev
        vesa
(++) Using config file: "/root/xorg.conf.new"
(==) Using system config directory "/usr/share/X11/xorg.conf.d"


Xorg detected your mouse at device /dev/input/mice.
Please check your config if the mouse is still not
operational, as by default Xorg tries to autodetect
the protocol.

Your xorg.conf file is /root/xorg.conf.new

To test the server, run 'X -config /root/xorg.conf.new'

(EE) Server terminated with error (2). Closing log file.
root@astra:~# _
```

Рис. 22.3. Создание файла конфигурации Xorg

Вывод этой команды показан на рис. 22.3. В нем есть два важных момента:

- список драйверов видеокарты, которые установлены в вашей системе (на рис. 22.3 видно, что установлены драйверы для популярных карточек от Ati, Intel и пр.). Это базовые драйверы, но если вы соберетесь заниматься 3D-моделированием, вам придется установить драйвер от производителя видеокарты;
- сообщение о том, что созданный файл конфигурации xorg.conf.new помещен в каталог /root/.

Откройте созданный файл конфигурации. На рис. 22.4, снятом на моей машине, видно, что используется драйвер `vmware`. На реальной машине у вас будет реальный драйвер и реальная секция `Device`.

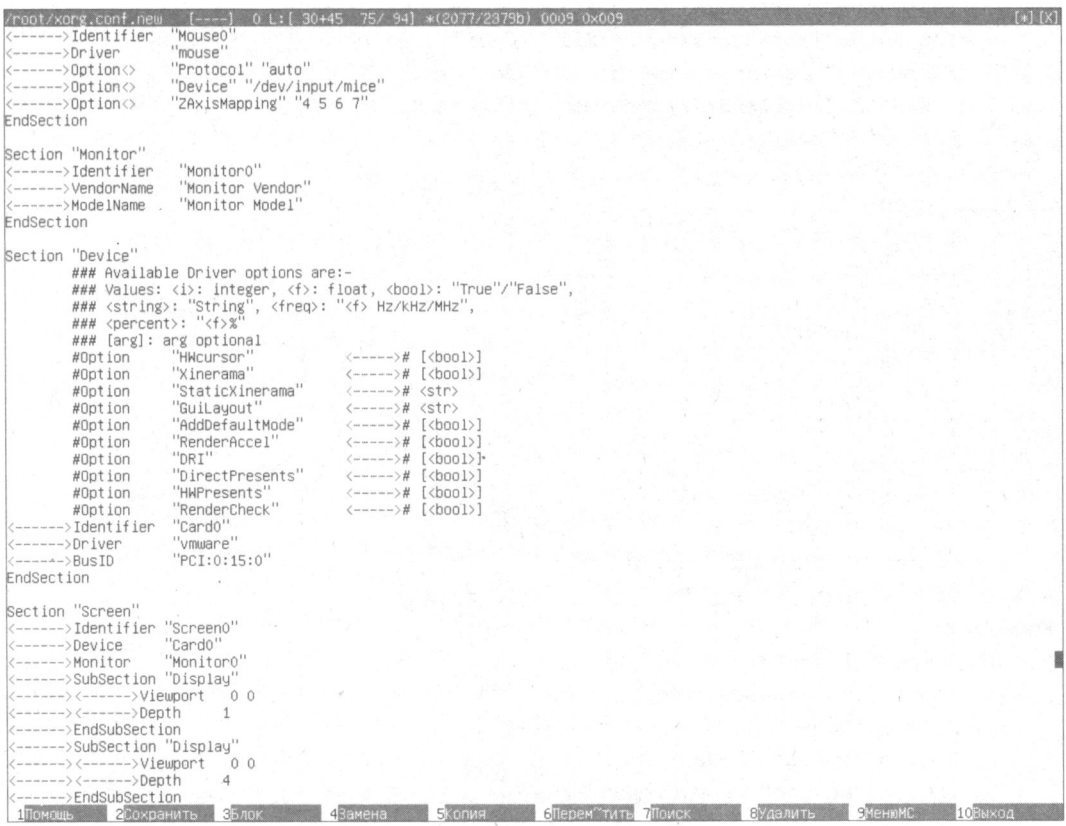

Рис. 22.4. Прототип созданного файла конфигурации

Сгенерированный файл будет содержать конфигурацию для одного монитора. Нам же нужно привести ее к конфигурации на два монитора. Весьма желательно при этом, чтобы оба монитора поддерживали одно и то же разрешение. Можно, конечно, использовать и разные разрешения, но тогда графический режим может работать некорректно. Примерное содержимое файла конфигурации на два монитора приведено в листинге 22.1.

Листинг 22.1. Файл xorg.conf

```
Section "ServerLayout"
       Identifier "X.org Configured"
       Screen 0 "Screen0" 0 0
       InputDevice "Mouse0" "CorePointer"
       InputDevice "Keyboard0" "CoreKeyboard"
EndSection
Section "Files"
       ModulePath "/usr/lib/xorg/modules"
       FontPath "/usr/share/fonts/X11/misc"
       FontPath "/usr/share/fonts/X11/100dpi/:unscaled"
       FontPath "/usr/share/fonts/X11/75dpi/:unscaled"
       FontPath "/usr/share/fonts/X11/Type1"
       FontPath "/usr/share/fonts/X11/100dpi"
       FontPath "/usr/share/fonts/X11/75dpi"
       FontPath "/usr/share/fonts/msttcorefonts"
       FontPath "built-ins"
EndSection
Section "Module"
       Load "record"
       Load "dri"
       Load "dri2"
       Load "extmod"
       Load "dbe"
       Load "glx"
EndSection
Section "InputDevice"
       Identifier "Keyboard0"
       Driver "kbd"
       Option "XkbLayout" "us,ru"
       Option "XkbOptions" "grp:ctrl_shift_toggle"
EndSection
Section "InputDevice"
       Identifier "Mouse0"
       Driver "mouse"
       Option "Protocol" "auto"
       Option "Device" "/dev/input/mice"
       # Option "ZAxisMapping" "4 5 6 7"
EndSection
Section "Monitor"
       Identifier "MonitorLVDS"
       # VendorName "Monitor Vendor"
       ModelName "Monitor Model"
       Option "DPMS" "False"
       Option "Position" "0 0"
EndSection
```

```
Section "Monitor"
        Identifier "MonitorVGA"
        # VendorName "Monitor Vendor"
        ModelName "Monitor Model"
        Option "DPMS" "False"
        Option "RightOF" "MonitorLVDS"
        Option "Position" "1280 0"
EndSection
Section "Device"
        ### Available Driver options are:-
        ### Values: <i>: integer, <f>: float, <bool>: "True"/"False",
        ### <string>: "String", <freq>: "<f> Hz/kHz/MHz"
        ### [arg]: arg optional
        #Option "ColorKey"# <i>
        #Option "Dac6Bit" # [<bool>]
        Option "DRI" "True"# [<bool>]
        #Option "NoDDC" # [<bool>]
        #Option "ShowCache" # [<bool>]
        #Option "PageFlip" # [<bool>]
        Option "Tiling" "True"
        Option "XvPreferOverlay" "True"
        Option "FallbackDebug" "False"
        Option "XvMC" "True"
        Identifier "Card0"
        Driver "intel"
        VendorName "Intel Corporation"
        BoardName "Core Processor Integrated Graphics Controller"
        BusID "PCI:0:2:0"
        Option "monitor-VGA1" "MonitorLVDS"
        Option "monitor-HDMI1" "MonitorVGA"
EndSection

Section "Screen"
        Identifier "Screen0"
        Device "Card0"
        Monitor "MonitorLVDS"
        SubSection "Display"
                Viewport 0 0
                Depth 24
                # Modes "1366x768" # разрешение вашего монитора 1
        EndSubSection
        EndSection
Section "Screen"
        Identifier "Screen0"
        Device "Card0"
        Monitor "MonitorVGA"
```

```
        SubSection "Display"
                Viewport 0 0 Depth 24
                #Modes "1280x1024" # разрешение вашего монитора 2
        EndSubSection
EndSection
```

Выводы, к которым подключены мониторы

К каким выводам подключены мониторы, можно определить по журналу */var/log/Xorg.log.0*, — возможно, вместо `"monitor-HDMI1" "Monitor-VGA1"` будут активны выходы `"monitor-DP1", "monitor-DP2", "monitor-HDMI2"`.

Этот файл не нужно слепо копировать — его надо отредактировать под вашу конфигурацию:

- секции `ServerLayout` и `Files` сверьте со своими секциями и проверьте пути к шрифтам;
- секцию `Module` приведите к виду из листинга 22.1;
- секции `InputDevice` пусть будут такими, как в вашем эталонном файле конфигурации, — это конфигурация клавиатуры и мыши, не нужно их слепо копировать.
- секцию `Device` скопируйте из своего файла конфигурации — она описывает видеокарту;
- далее вам нужно описать две секции `Monitor`. В нашем примере мы настраиваем два монитора: MonitorLVDS и MonitorVGA. Это идентификаторы мониторов, вы можете задать свои, но не забудьте тогда отредактировать и дальнейшие фрагменты кода. Первый монитор в нашем примере — это `MonitorLDVS`. Пусть его разрешение по горизонтали равно 1280 пикселов (это важно!). Мы задаем позицию изображения параметром `"Position" "0 0"`. Это и есть первый монитор. Для второго монитора мы указываем, что он находится справа от `MonitorLDVS` — `"RightOF"`, и его `"Position" "1280 0"`, т. е. позиция второго монитора начинается с 1280-го пиксела. Если разрешение первого монитора по горизонтали 1920, то и в `Position` нужно указать 1920.

```
Section "Monitor"
        Identifier "MonitorLVDS"
        # VendorName "Monitor Vendor"
        ModelName "Monitor Model"
        Option "DPMS" "False"
        Option "Position" "0 0"
EndSection
Section "Monitor"
        Identifier "MonitorVGA"
        # VendorName "Monitor Vendor"
        ModelName "Monitor Model"
        Option "DPMS" "False"
        Option "RightOF" "MonitorLVDS"
        Option "Position" "1280 0"
EndSection
```

- в секции `Device` обязательно нужно указать, к какому выходу подключены наши мониторы. Здесь монитор LVDS подключен к HDMI1, монитор MonitorVGA — к VGA1:

```
Option "monitor-VGA1" "MonitorVGA"
Option "monitor-HDMI1" "MonitorLVDS"
```

- секция `Screen` должна связать монитор (параметр `Monitor`) с видеокартой (параметр `Card`) и задать разрешение монитора. Разрешение задается параметром `Modes`:

```
Section "Screen"
        Identifier "Screen0"
        Device "Card0"
        Monitor "MonitorLVDS"
        SubSection "Display"
                Viewport 0 0
                Depth 24
                # Modes "1366x768" # разрешение вашего монитора 1
        EndSubSection
        EndSection
Section "Screen"
        Identifier "Screen0"
        Device "Card0"
        Monitor "MonitorVGA"
        SubSection "Display"
                Viewport 0 0 Depth 24
                #Modes "1280x1024" # разрешение вашего монитора 2
        EndSubSection
EndSection
```

Теперь переключитесь на консоль и попробуйте запустить X в тестовом режиме:

```
X :1 -config /путь/xorg.conf.new
```

Здесь */путь* — это путь к файлу конфигурации. Если результат вас не устроил, вернитесь к редактированию файла конфигурации. Если же все хорошо, вернитесь на консоль и введите команды:

```
sudo cp /путь/xorg.conf.new /etc/X11/xorg.conf
sudo reboot
```

Первая команда копирует ваш конфигурационный файл в каталог /etc/X11, вторая — перезагружает компьютер.

ASTRA LINUX®

ГЛАВА 23

Интеграция со смартфоном

- Необходимость интеграции
- Интеграция с Android-смартфоном
- Интеграция с iPhone

23.1. Необходимость интеграции

Прежде всего нужно понять, зачем нам интеграция со смартфоном. Если идет речь о том, чтобы использовать те же мессенджеры, которые установлены на смартфоне, — ведь на компьютере большая клавиатура и писать на ней проще, — то эта задача уже решена ранее. При желании можно установить в Linux популярные мессенджеры и общаться с друзьями с компьютера без необходимости подключения к нему смартфона.

Наиболее частая задача, которая ставится перед интеграцией со смартфоном, — это передача файлов, в частности фотографий, со смартфона на компьютер и, возможно, обратно. И это можно осуществить двумя путями: по кабелю и «по воздуху», т. е. беспроводным способом.

Далее мы разберемся, как выполнить интеграцию Linux-компьютера со смартфоном, в том числе и для передачи файлов, на двух самых популярных смартфон-платформах: Android и iOS.

23.2. Интеграция с Android-смартфоном

23.2.1. Беспроводной способ

Для беспроводного доступа с компьютера к смартфону в Play Market можно найти множество всевозможных программ. Лучшая из них, на мой взгляд, AirDroid. Эта программа предоставляет пользователю компьютера полный доступ к экрану и файловой системе смартфона. Но давайте посмотрим на нее в действии. Установите на свой смартфон приложение AirDroid и запустите его. Перейдите в раздел

AirDroid Web — там вы увидите IP-адрес и номер порта, через который осуществляется доступ к устройству (рис. 23.1).

Запустите на компьютере браузер и введите указанный адрес. В нашем случае — это **http://192.168.0.113:8888**. После этого на смартфоне нужно разрешить доступ с компьютера (рис. 23.2).

Рис. 23.1. IP-адрес и порт для доступа к смартфону

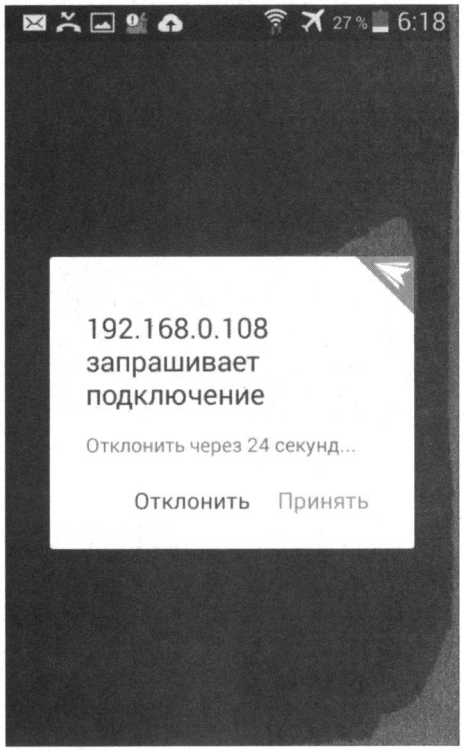

Рис. 23.2. Разрешите доступ

Как только вы разрешите доступ, в браузере отобразится интерфейс вашего смартфона (рис. 23.3). Обратите внимание: в верхней части экрана отображается количество пропущенных вызовов.

Стандартные приложения AirDroid предоставляют доступ к журналу звонков, контактам, файлам, изображениям, музыке, видео и др. В Toolbox, расположенном справа вверху на экране компьютера (рис. 23.4), выводится информация о смартфоне, а также — на вкладке **Файл** — предоставляется возможность быстрой загрузки файлов с компьютера на смартфон.

Для быстрой загрузки на смартфон файла с компьютера выполните следующие действия:

1. Перейдите на вкладку **Файл** области Toolbox.
2. Выберите папку, в которую нужно загрузить файлы. По умолчанию загрузка осуществляется в папку /sdcard/airdroid/upload.

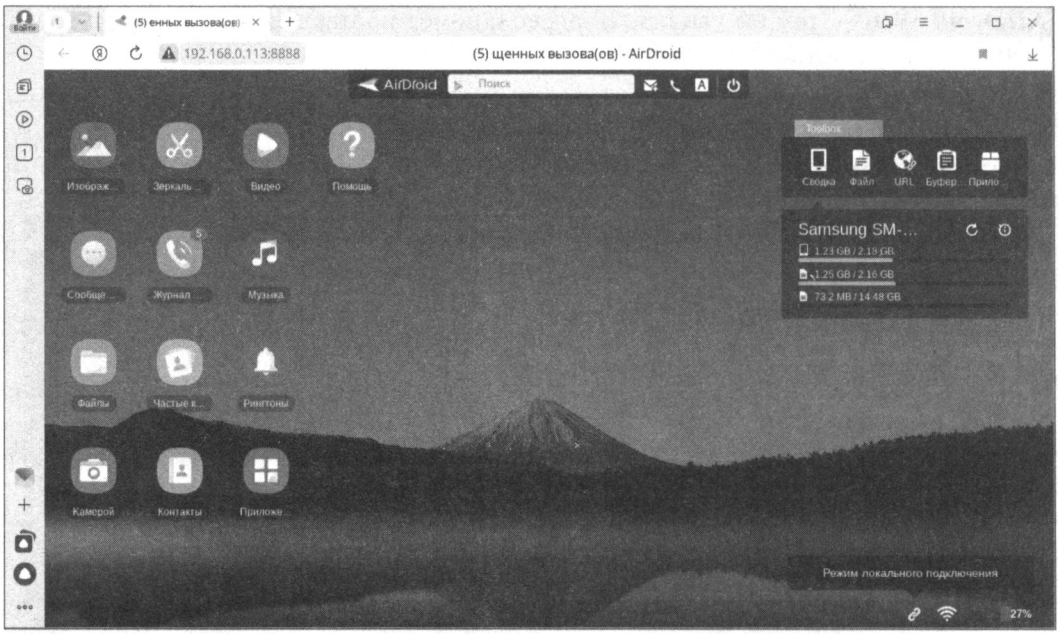

Рис. 23.3. Интерфейс управления смартфоном в браузере

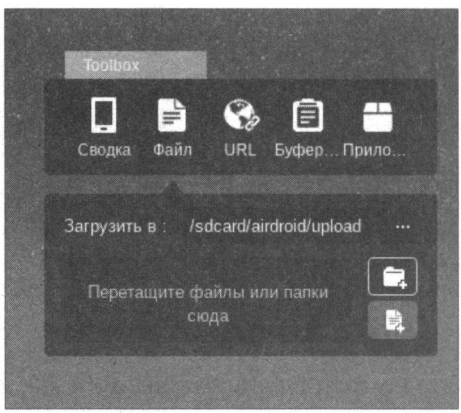

Рис. 23.4. Быстрая загрузка файла

3. Перетащите файл (файлы и/или папки) в область **Перетащите файлы или папки сюда**.
4. Дождитесь загрузки файлов на смартфон.

Чтобы скачать фото на свой компьютер со смартфона:

1. Запустите приложение **Изображения** интерфейса AirDroid, щелкнув на значке на экране компьютера.
2. Выберите изображения, которые вы хотите скачать (рис. 23.5).
3. Нажмите кнопку **Скачать.**

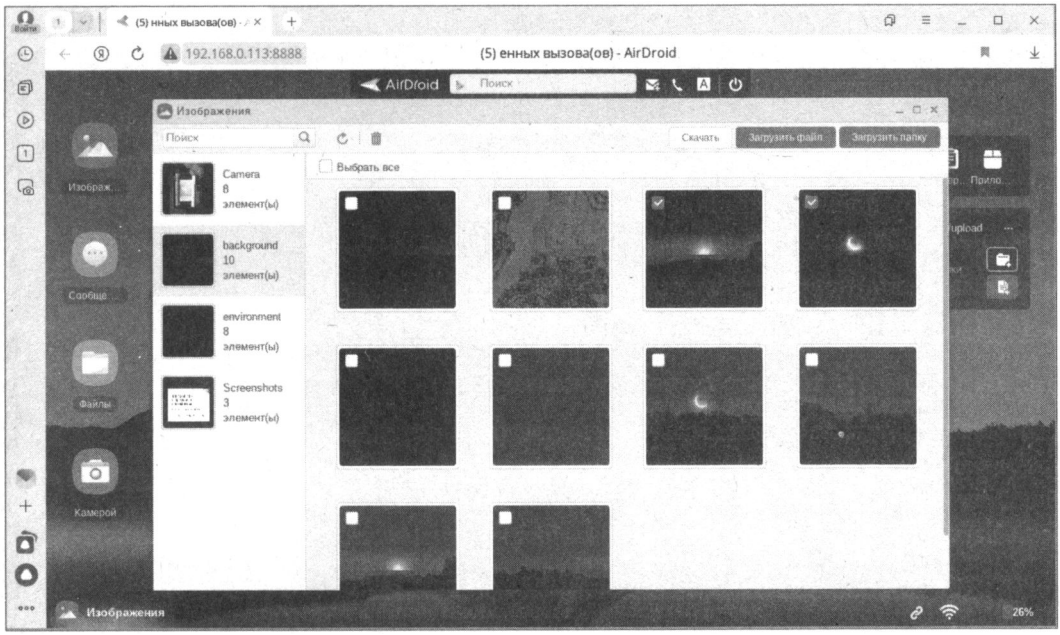

Рис. 23.5. Панель приложения **Изображения** на экране компьютера: представлены миниатюры изображений, хранящихся на смартфоне

4. Изображения будут заархивированы, а архив с ними — отдан на загрузку в браузер. По окончании загрузки откроется приложение Ark (рис. 23.6), в котором будет открыт загруженный архив. Вы можете оставить его как есть, а можете распаковать файлы. Содержимое распакованного архива будет загружено в папку **Загрузки** компьютера (рис. 23.7).

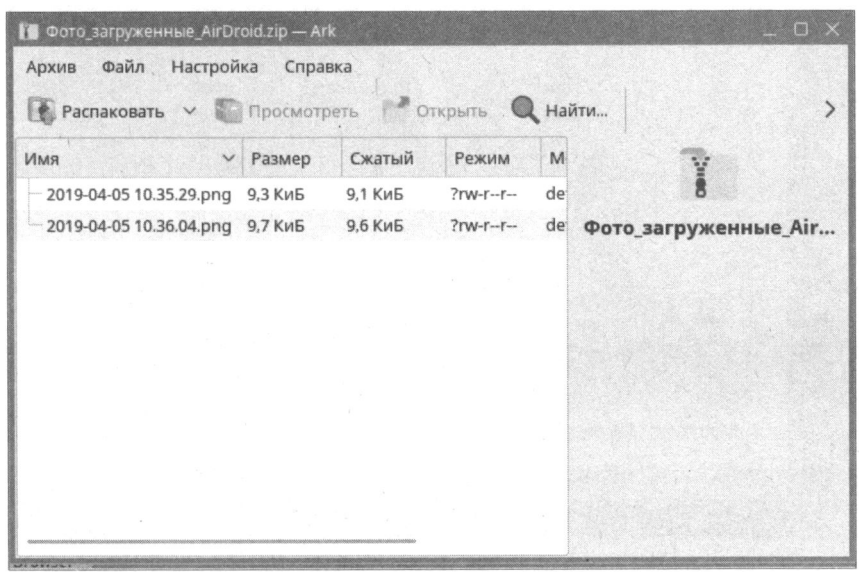

Рис. 23.6. Архиватор Ark

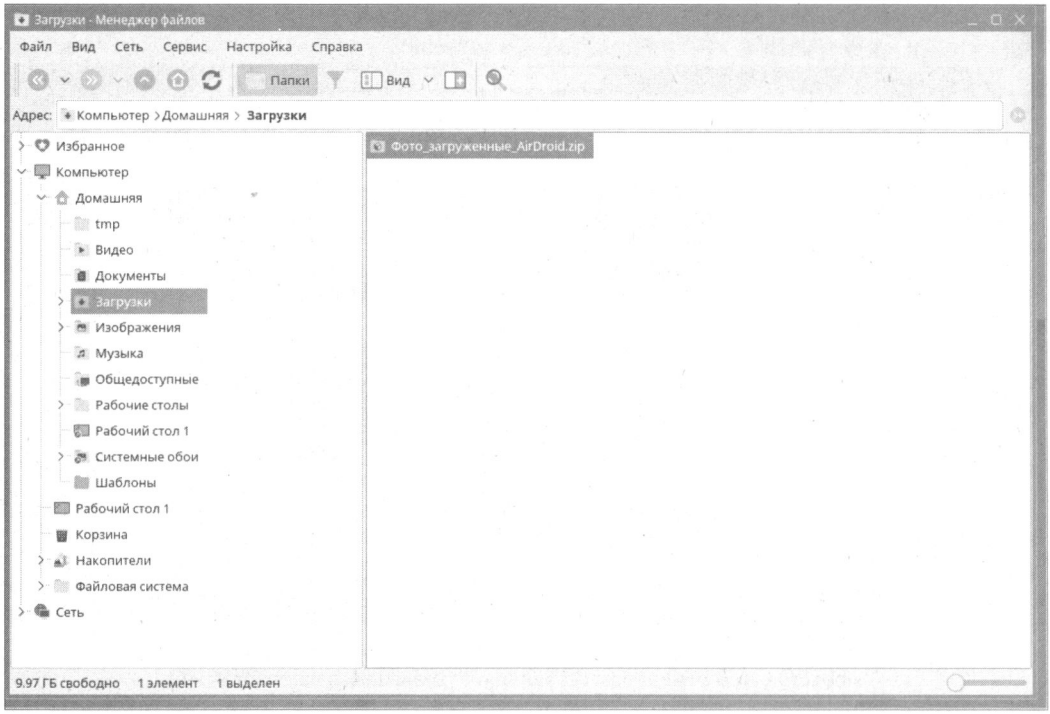

Рис. 23.7. Папка **Загрузки** компьютера

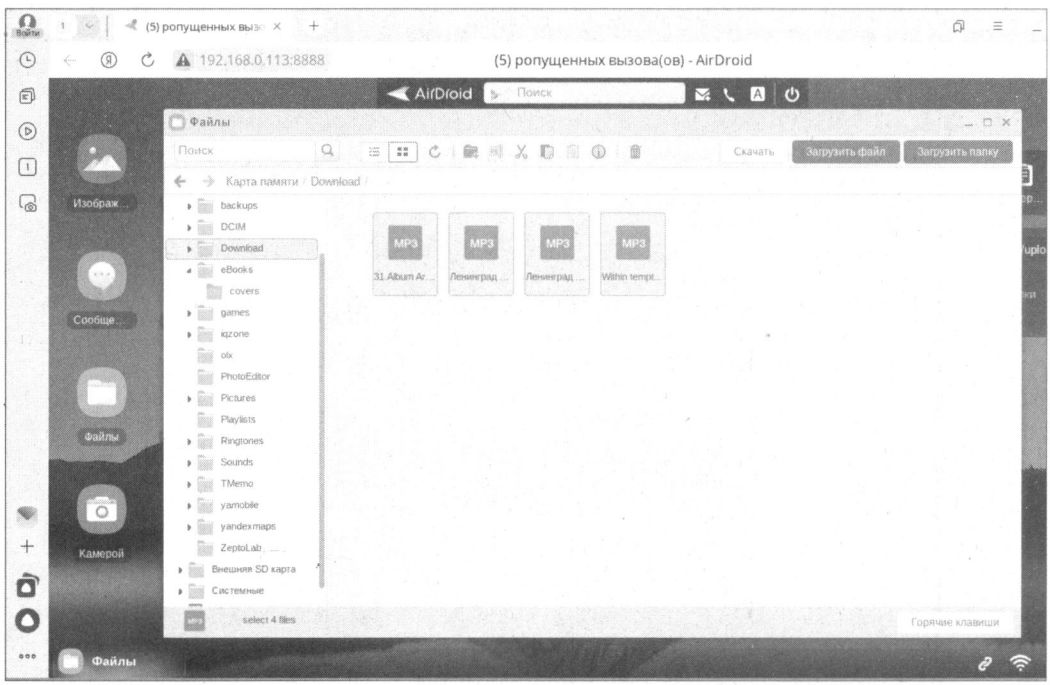

Рис. 23.8. Панель приложения **Файлы** на экране компьютера: представлены файлы изображений, хранящихся на смартфоне

Прямая загрузка файлов и папок

Кнопки **Загрузить файл** и **Загрузить папку**, показанные в правом верхнем углу панели приложения **Изображения** (см. рис. 23.5), также позволяют загрузить файл или папку в галерею изображений смартфона.

Для загрузки файлов со смартфона и на него также можно воспользоваться приложением **Файлы** интерфейса AirDroid:

1. Откройте приложения **Файлы**, щелкнув на значке на экране компьютера, и перейдите в нужную папку (рис. 23.8).
2. Если нужно скачать файлы со смартфона на компьютер, выделите их и нажмите кнопку **Скачать**, — архив со скачанными файлами будет загружен в папку **Загрузки**. Если нужно закачать файлы с компьютера на смартфон, нажмите кнопку **Загрузить файл**, выберите файлы и дождитесь их загрузки. Аналогичные операции позволят загружать и папки.

23.2.2. Используем кабель для передачи файлов

Здесь все проще, и вам не придется устанавливать на свой смартфон какие-либо приложения, — достаточно только подключить смартфон к компьютеру с помощью USB-кабеля. В зависимости от версии Android, возможно, придется разрешить подключение на самом смартфоне. После этого система «увидит» ваш смартфон и предложит открыть его файлы в файловом менеджере (рис. 23.9).

Если вы не успели нажать кнопку **Открыть в менеджере файлов**, откройте менеджер файлов и перейдите в раздел **Накопители** (он находится слева в дереве файловой системы) — вы увидите, к какой папке компьютера подмонтирован ваш смартфон (рис. 23.10). Вам остается только перейти в эту папку для доступа к его файлам (рис. 23.11).

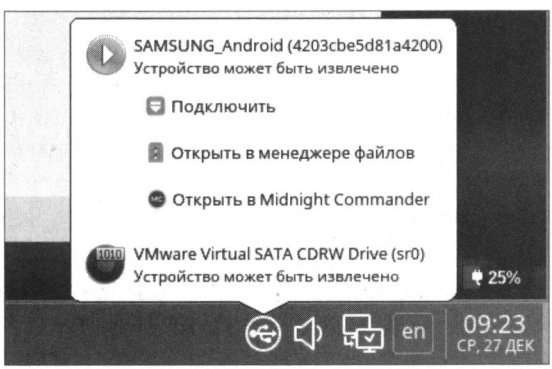

Рис. 23.9. Компьютер обнаружил смартфон, подключенный к нему кабелем

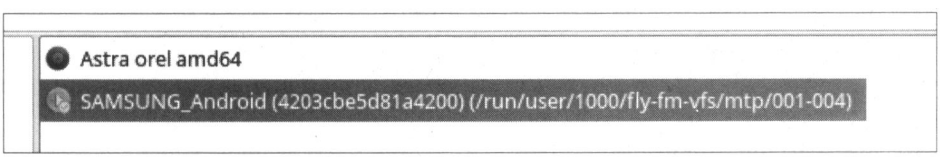

Рис. 23.10. Название папки, к которой подмонтирован смартфон

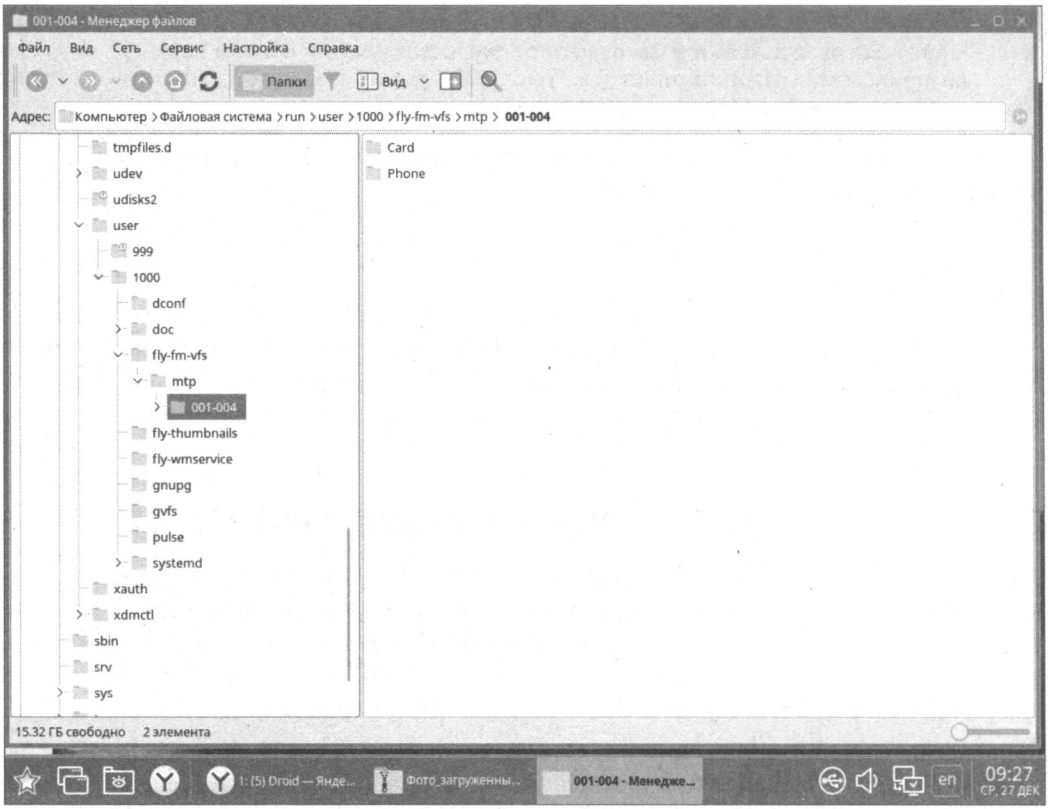

Рис. 23.11. Доступ к файлам смартфона по кабелю

23.3. Интеграция с iPhone

Постепенно вымирает все, что связано с кабелем. Сначала появились беспроводные сети, а кабельные сети остались или в крупных предприятиях, где они были давным-давно проложены, или у провайдеров, где это производственная необходимость. С появлением беспроводной зарядки постепенно происходит отказ и от обычных кабелей для зарядки телефонов.

Понятно, что хочется и обмен данными между компьютером и смартфоном осуществлять «по воздуху», но никак не по кабелю. Тем более что те, кто пробовал передать с iPhone на компьютер по кабелю те же фотографии или видео знают, насколько мучителен сам процесс — файлы огромные, а скорость передачи очень низкая.

Счастливые обладатели MacBook могут передавать файлы «по воздуху», используя AirDrop, но, учитывая, что ваш MacBook привязан к тому же аккаунту, что и ваш iPhone, в этом нет необходимости — фотографии и видео с iPhone будут автоматически загружены и на Mac.

Однако MacBook есть не у всех, и пока вы им не обзавелись, нужно как-то взаимодействовать с iPhone, используя обычный компьютер. К счастью, для вас есть две хорошие новости:

- существует набор приложений iCloud for Linux;
- iCloud for Linux работает в Astra Linux, что немаловажно, учитывая, что используемые дистрибутивом библиотеки не первой свежести.

Итак, давайте узнаем, как установить iCloud for Linux и запустить его.

Установка будет производиться не посредством пакетов, а с помощью снапов, — именно поэтому приложение нормально работает в дистрибутиве с не очень свежими библиотеками. Поэтому, если вы пропустили *главу 12*, вернитесь и убедитесь, что демон snapd у вас установлен и работает.

Для установки приложения:

1. Введите команду: `sudo snap install icloud-for-linux`.
2. Дождитесь установки снапа (рис. 23.12).

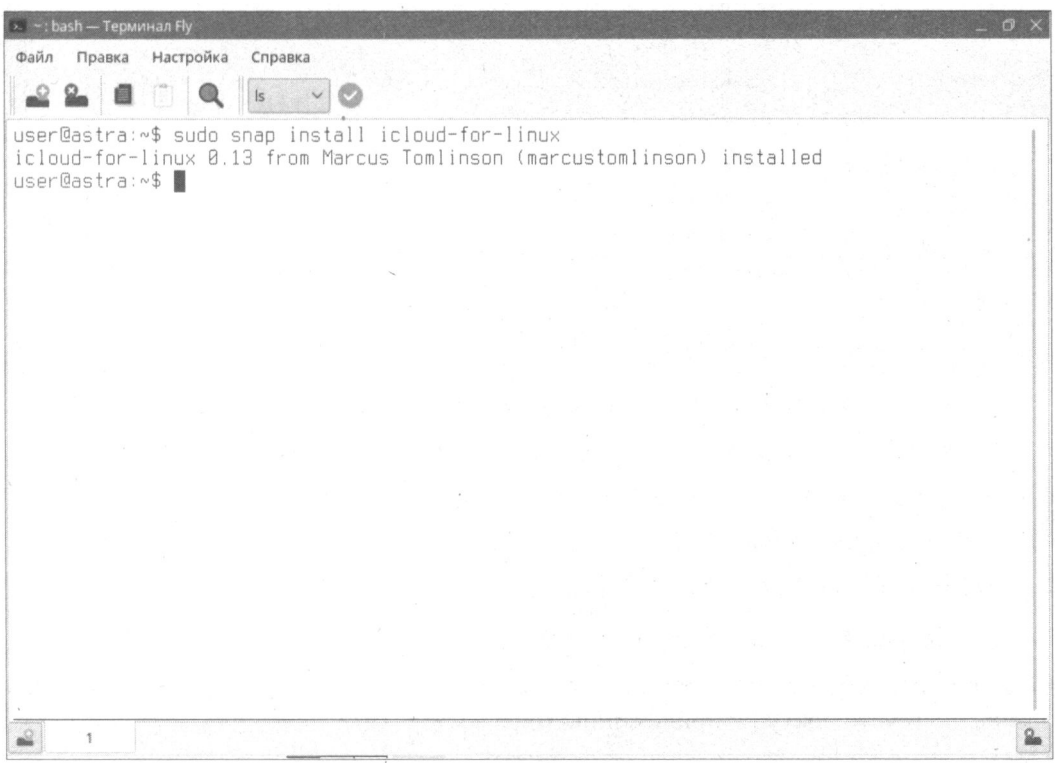

Рис. 23.12. Снап установлен

После установки в группе **Офис** у вас появятся новые приложения для интеграции с iPhone (рис. 23.13). В частности, это будут приложения: iCloud Calendar — для просмотра записей календаря, задач и встреч, iCloud Contacts — для просмотра контактов, iCloud Drive — для доступа к файлам на облачном диске iCloud, iCloud Find My iPhone — для поиска вашего iPhone, если он, не дай бог, будет утерян или украден, iCloud Photos — для доступа к фото и видео и др.

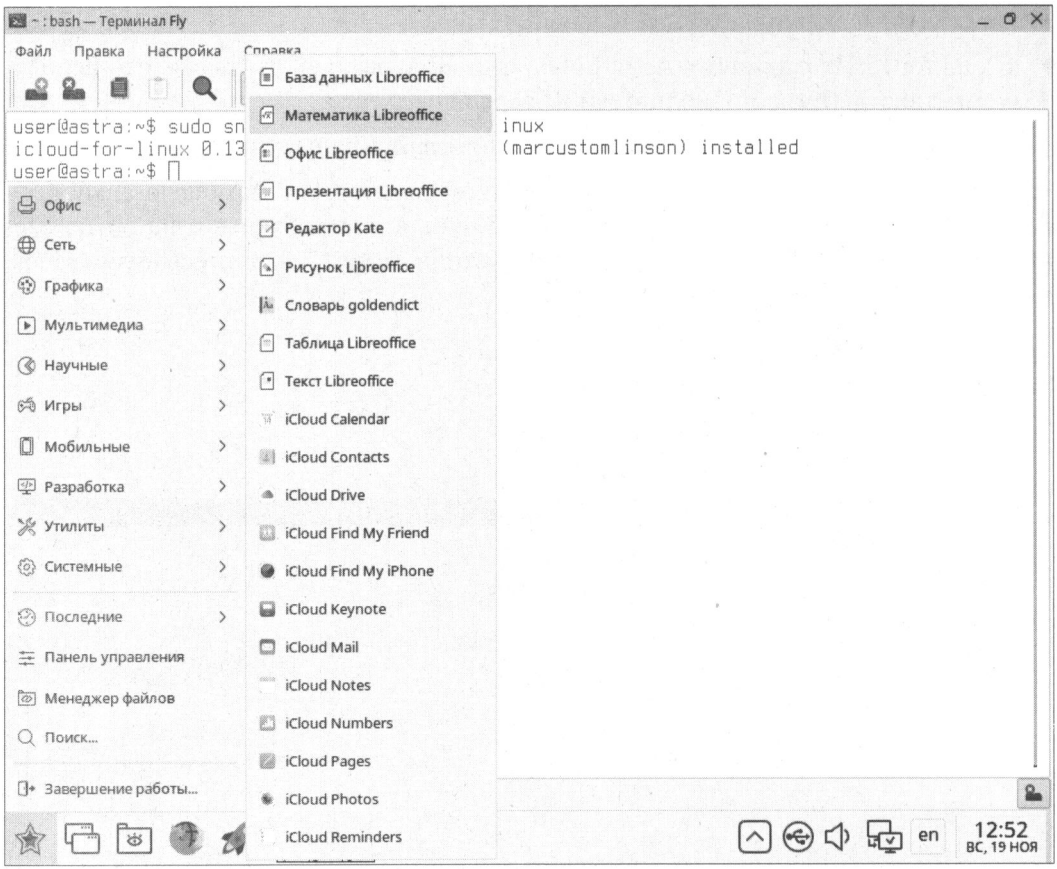

Рис. 23.13. Новые приложения в группе **Офис** вашего компьютера

Использовать каждое приложение достаточно просто:

1. Запустите приложение.
2. Нажмите кнопку **Войти** (рис. 23.14).
3. Введите ваш Apple ID и пароль.
4. На iPhone получите код для доступа к аккаунту, введите его в окне приложения, и вы получите доступ к приложению (рис. 23.15).
5. Из любого приложения вы можете вызвать панель **Управление устройствами** (рис. 23.16) для просмотра привязанных к аккаунту устройств.

Недостаток набора iCloud for Linux состоит в том, что вам придется авторизироваться в каждом приложении из набора. Это создает определенные неудобства, но выбора тут нет: или пользуйтесь тем, что есть, или переходите на всю экосистему от Apple.

Рис. 23.14. Нажмите кнопку **Войти**

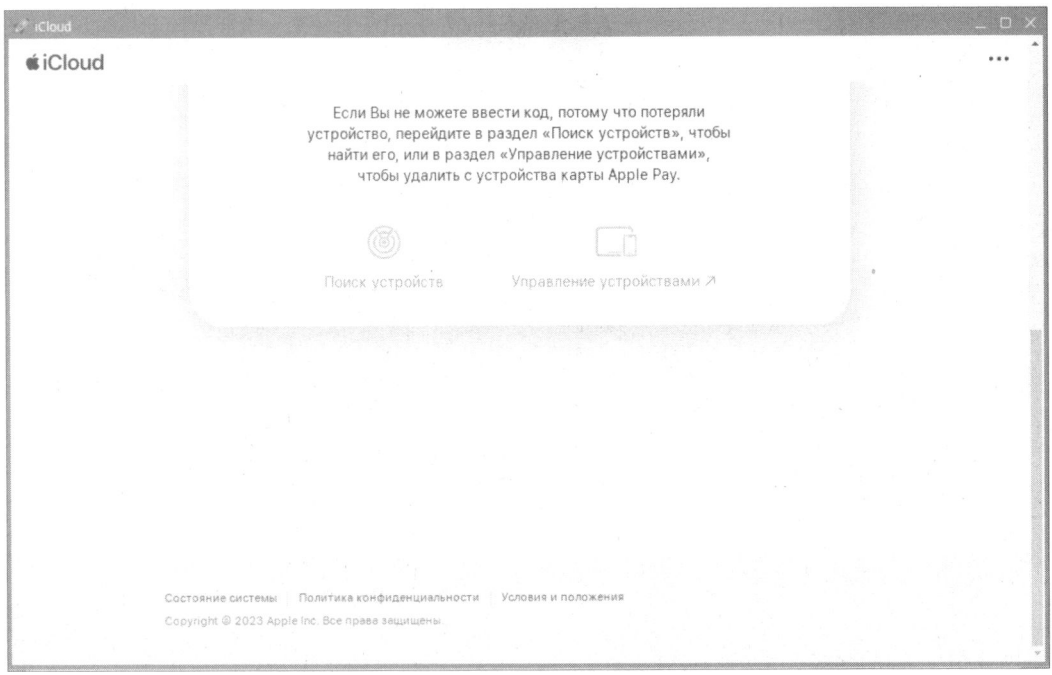

Рис. 23.15. Приложение запущено

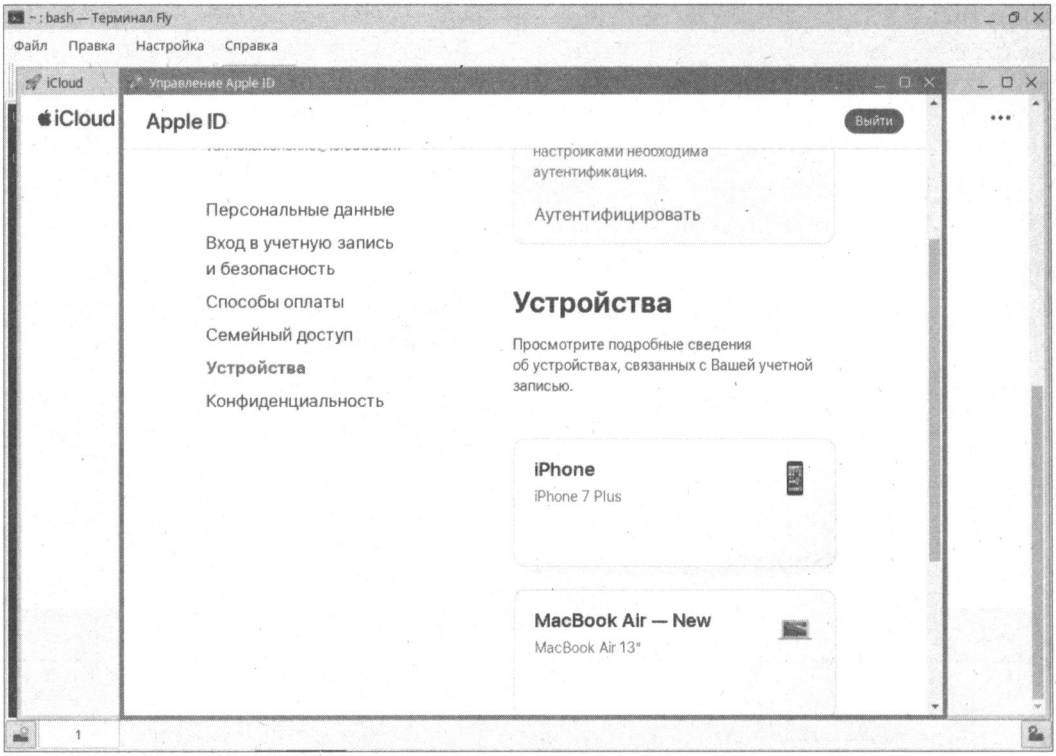

Рис. 23.16. Список устройств

ЧАСТЬ VI

Системное администрирование

ASTRA LINUX®

ГЛАВА 24

Управление учетными записями пользователей

- Полномочия пользователя root
- Команда `sudo`
- Команда `su`
- Создание и удаление пользователей
- Группы пользователей
- Конфигуратор **Политика безопасности**

24.1. Полномочия пользователя root

Пользователь root обладает в системе максимальными полномочиями, и она полностью подвластна этому пользователю, — любая его команда будет выполнена безоговорочно. Поэтому работать под именем пользователя root нужно с осторожностью. Всегда думайте над тем, что собираетесь сделать, — если вы дадите команду на удаление корневой файловой системы, система выполнит и ее. В отличие от пользователя root, пользователю, зарегистрировавшемуся под обычным именем, при попытке выполнить ту или иную команду система сообщит, что у него нет для этого полномочий.

Представим, что кто-то решил пошутить и выложил в Интернете (записал на диск или прислал по электронной почте — способ доставки не важен) вредоносную программу. Если вы ее запустите от имени пользователя root, система может быть повреждена. Запуск этой же программы от имени обычного пользователя ничего страшного не произведет — система откажется ее выполнять. Впрочем, все может быть намного проще: вы сами ошибочно введете команду, способную разрушить систему, или отойдете ненадолго от своего компьютера, а появившийся тут как тут некий «доброжелатель», имея полномочия пользователя root, сможет уничтожить систему одной командой.

Именно поэтому практически во всех современных дистрибутивах вход под именем пользователя root запрещен. В одних дистрибутивах вы не сможете войти как root в графическом режиме (но можете войти в консоли, переключившись на пер-

вую консоль с помощью комбинации клавиш <Ctrl>+<Alt>+<F1>), а в других — вовсе не можете войти в систему как root: ни в графическом режиме, ни в консоли (например, в Ubuntu).

В Astra Linux вход как root не отключен, но пароль root по умолчанию неизвестен. Посмотрим, как можно войти как root в Astra Linux:

- войдите в систему как администратор (как пользователь, которого вы создали при установке) и введите команду: `sudo passwd root`. Затем введите новый пароль root и его подтверждение (рис. 24.1);
- нажмите комбинацию клавиш <Ctrl>+<Alt>+<F1> (так вы перейдете в консоль) и войдите как root (рис. 24.2);
- нажмите комбинацию клавиш <Alt>+<F7>, выйдите из графического сеанса, войдите снова, но уже как пользователь root (рис. 24.3). Откройте терминал Fly, чтобы увидеть текущее имя пользователя. Команда `w` показывает, что пользователь root вошел в графическом режиме (:0), а также на консоли tty1.

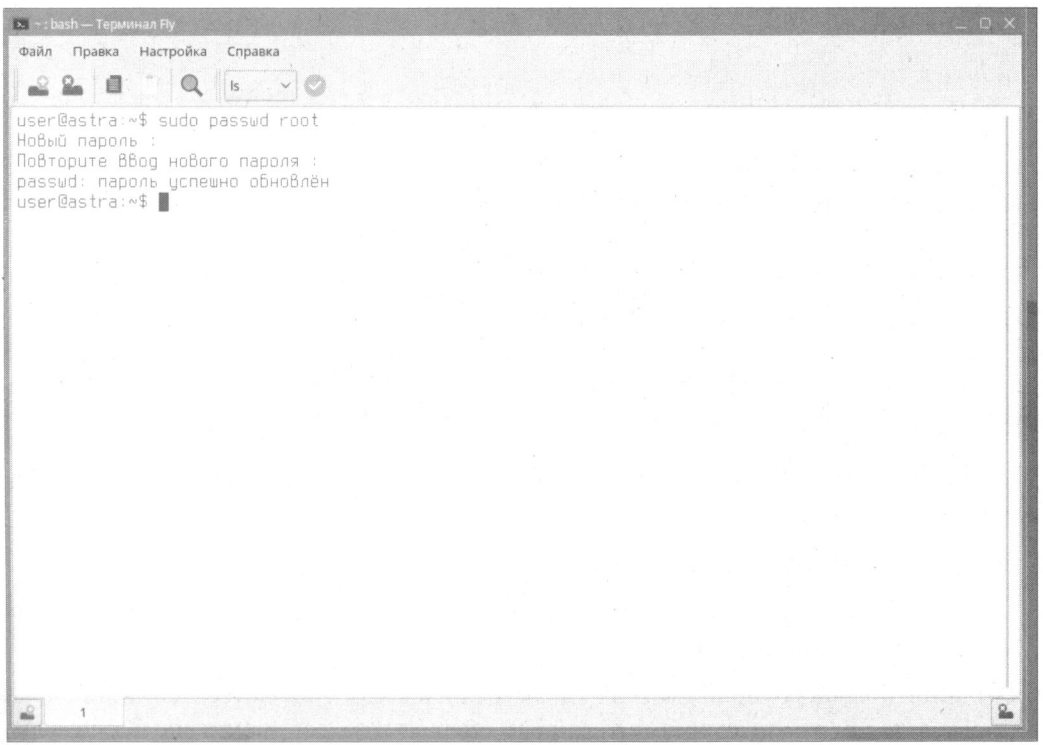

Рис. 24.1. Изменение пароля root

Рис. 24.2. Вход как root в консоли

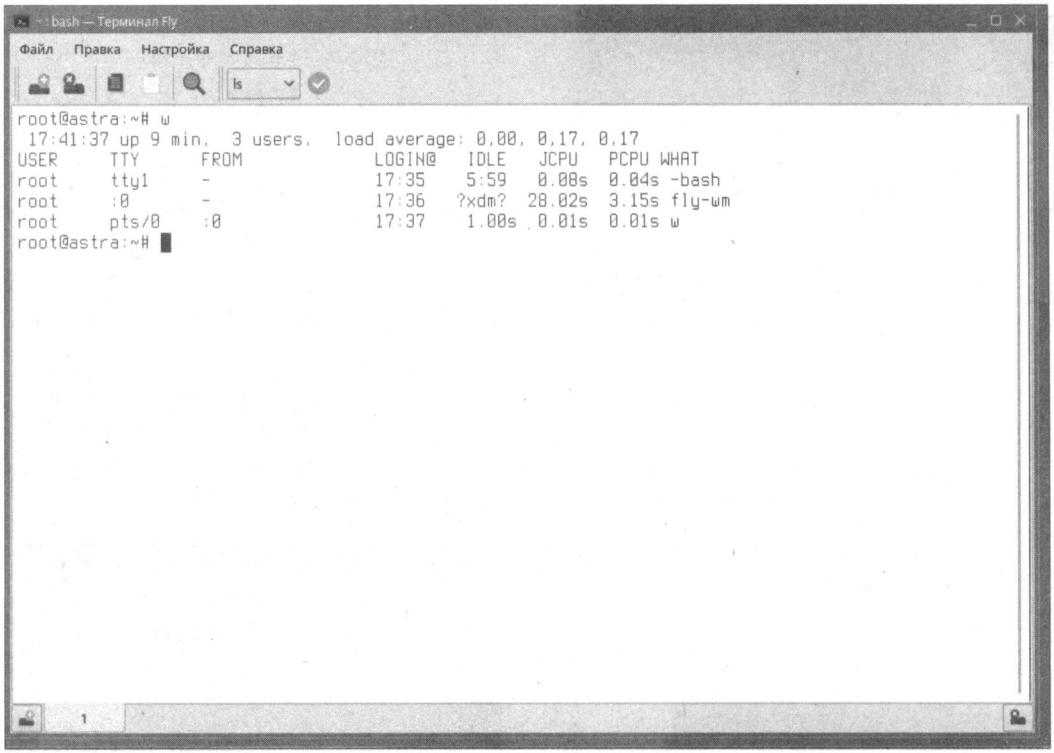

Рис. 24.3. Вход как root в графическом режиме

Если вы запускаете какую-либо графическую программу, требующую привилегий root, или же какая-то программа пытается выполнить действие, требующее таких полномочий (например, произвести запись в файл конфигурации), то увидите окно с требованием ввести соответствующий пароль (рис. 24.4).

Отсюда можно сделать следующие выводы:

- старайтесь реже работать под пользователем root;
- всегда думайте, какие программы вы запускаете под именем root;
- если программа, полученная из постороннего источника, требует root-полномочий, это должно насторожить;
- создайте обычного пользователя (даже если вы сами являетесь единственным пользователем компьютера) и рутинные операции (с документами, использование Интернета и т. д.) производите от имени этого пользователя;
- если полномочия root нужны для выполнения серии команд, используйте команды `su` и `sudo` (см. далее) для кратковременного получения прав root.

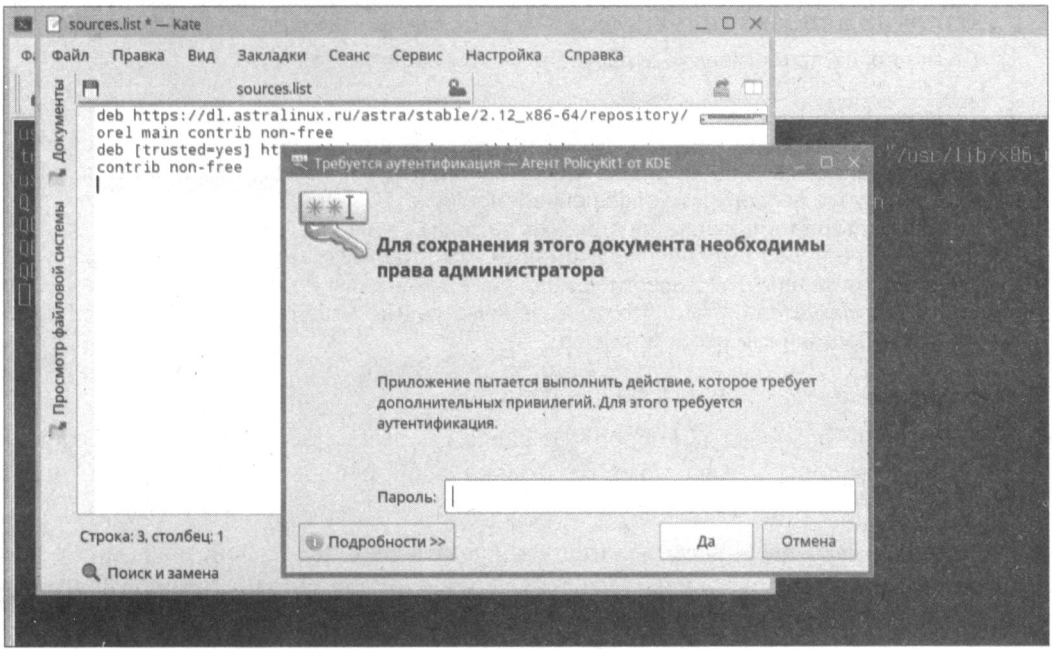

Рис. 24.4. Попытка сохранить файл конфигурации: запрос прав root

24.2. Команда *sudo*

Команда `sudo` позволяет запустить любую команду с привилегиями root. Использовать ее нужно так:

```
sudo <команда_которую_нужно_выполнить_с_правами_root>
```

Например, вам необходимо изменить файл /etc/apt/sources.list. Для этого служит команда:

```
sudo kate /etc/apt/sources.list
```

> ***ТЕКСТОВЫЙ РЕДАКТОР KATE***
>
> Программа kate — это текстовый редактор, ему мы передаем один параметр — имя файла, который нужно открыть.

Если ввести эту же команду, но без `sudo` (просто: `kate /etc/apt/sources.list`), текстовый редактор тоже запустится и откроет файл, но сохранить изменения в нем вы не сможете, поскольку у вас не хватит полномочий. При этом система отобразит окно, в котором нужно будет ввести пароль этого пользователя, но все сработает, только если пользователь является администратором — внесен в группу astra-admin (в других дистрибутивах — sudoers), иначе вы получите сообщение о том, что не хватает полномочий для выполнения операции.

Если вам нужно ввести серию команд `sudo`, но не хочется вводить `sudo` каждый раз, запустите оболочку bash с правами root:

```
sudo bash
```

Отличия использования sudo в Astra Linux

Во многих дистрибутивах sudo работает так: вы вводите команду:

```
sudo <команда, которую нужно выполнить с правами root>
```

система запрашивает ваш пароль, вы его вводите, и если пароль правильный и пользователь внесен в sudoers (является администратором), команда выполняется, в противном случае вы получите сообщение об ошибке. В Astra Linux достаточно того факта, что пользователь является администратором, — пароль вводить не требуется, но обязательно перед выполнением команды, которую надо выполнить с повышенными полномочиями, следует ввести команду sudo, иначе вы не получите права root. Так система страхуется от того, что вы не понимаете, что вводите, но при этом не надоедает вам постоянным вводом пароля.

Преимущества sudo заключаются в следующем:

- вам не нужно помнить два пароля (т. е. свой пользовательский пароль и пароль пользователя root) — вы помните только свой пароль и вводите его, когда нужно;
- с помощью sudo вы можете выполнять практически те же действия, что и под именем root, но перед каждым действием у вас будет запрошен пароль, что позволит еще раз подумать о правильности своих действий;
- каждая команда, введенная с помощью sudo, записывается в журнал /var/log/auth.log, и у вас сохранится история введенных команд с полномочиями root, тогда как при работе под именем root никакого такого журнала не ведется. Кроме того, если что-то пойдет не так, вы хотя бы будете знать, что случилось, изучив этот журнал;
- предположим, некто захотел взломать вашу систему. Он не знает, какие учетные записи имеются на вашем компьютере, зато уверен, что учетная запись root есть всегда. Знает он также, что, завладев паролем к этой учетной записи, можно получить неограниченный доступ к системе. Но не к вашей системе — у вас-то учетная запись root отключена!
- вы можете разрешать и запрещать другим пользователям использовать полномочия root (позже мы разберемся, как это сделать), не предоставляя пароль root, — это практически сводит на нет риск скомпрометировать учетную запись root (впрочем, риск есть всегда — ведь при неправильно настроенной системе с помощью команды sudo можно легко изменить пароль root).

Но у sudo есть и недостатки:

- неудобно задавать перенаправление ввода/вывода — например, команда:

  ```
  sudo ls /etc > /root/somefile
  ```

  работать не будет, и вместо нее следует использовать команду:

  ```
  sudo bash -c "ls /etc > /root/somefile"
  ```

  Длинновато, правда?

- имеются также неудобства, связанные с технологией управления пользователями NSS (Name Service Switch). К счастью, она используется не очень часто,

поэтому основной недостаток sudo будет связан только с перенаправлением ввода/вывода.

С другой стороны, всегда можно отдать команду sudo bash и далее вводить команды без ограничений.

24.3. Команда *su*

Команда su позволяет получить доступ к консоли с правами root любому пользователю (даже если пользователь не внесен в файл /etc/sudoers) при условии, что он знает пароль root. Понятно, что в большинстве случаев этим пользователем будет сам пользователь root, — не станете же вы всем пользователям доверять свой пароль?

Поэтому команда su предназначена в первую очередь для администратора системы, а sudo — для остальных пользователей, которым иногда нужны права root (чтобы они меньше отвлекали администратора от своей работы).

Использовать команду su просто:

```
su
```

После этого нужно будет ввести пароль пользователя root, и вы сможете работать в консоли как обычно. Использовать su удобнее, чем sudo, потому что вам не потребуется вводить su перед каждой командой, которая должна быть выполнена с правами root.

Чтобы закрыть сессию su, нужно или ввести команду exit, или просто закрыть окно терминала.

24.4. Создание и удаление пользователей

Для добавления нового пользователя выполните команду:

```
sudo adduser <имя пользователя>
```

Команда adduser в Astra Linux (рис. 24.5), как и в дистрибутивах Ubuntu и Debian, сразу запрашивает пароль пользователя (нет необходимости после добавления пользователя вводить команду passwd), а также позволяет указать дополнительную информацию о пользователях.

Команда passwd изменяет пароль пользователя. Если вы вводите ее без параметров, то она позволит сменить ваш пароль. Если же задается имя пользователя в качестве параметра, то программа изменит его пароль. Разумеется, для изменения пароля другого пользователя (например, с именем test) нужны полномочия root, поэтому надо использовать команду sudo:

```
sudo passwd test
```

Давайте разберемся, что же происходит при создании новой учетной записи пользователя.

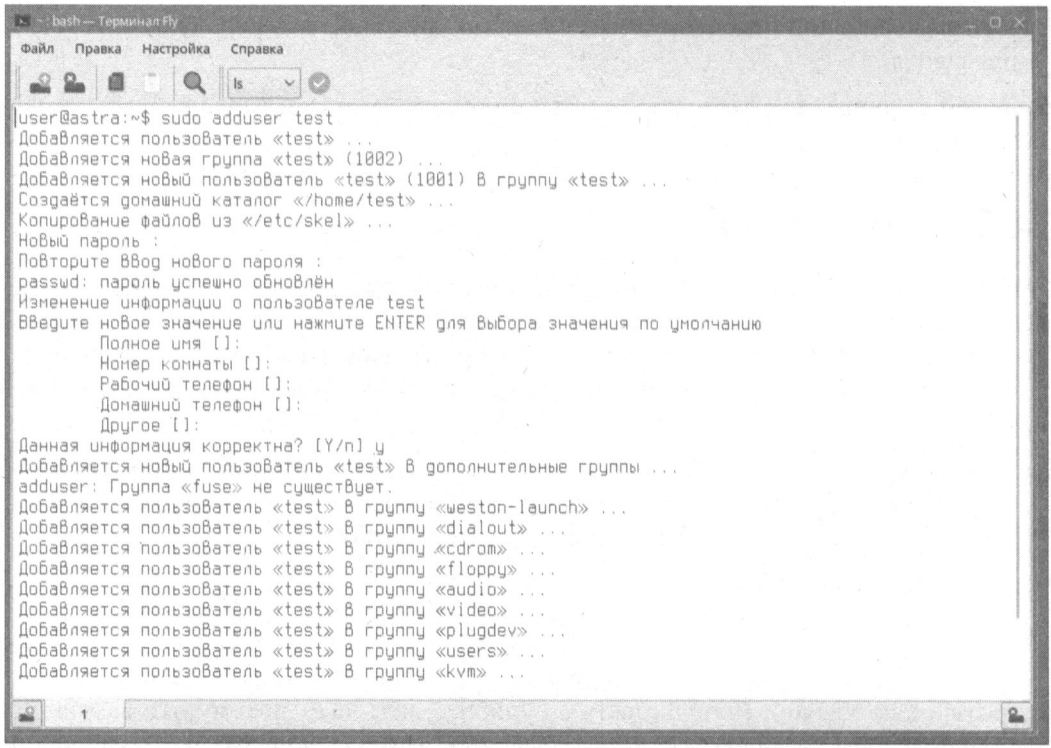

Рис. 24.5. Команда `adduser` в Astra Linux

Во-первых, создается запись в файле /etc/passwd. Формат записи следующий:

имя_пользователя:пароль:UID:GID:полное_имя:домашний_каталог:оболочка

Рассмотрим фрагмент этого файла (две строки):

```
root:x:0:0:root:/root:/bin/bash
test:x:500:500:test:/home/test:/bin/bash
```

- первое поле — это логин пользователя, который он вводит для регистрации в системе. Пароль в современных системах в этом файле не указывается, и второе поле осталось просто для совместимости со старыми системами. Пароли хранятся в файле /etc/shadow, о котором мы поговорим чуть позже;
- третье и четвертое поля: `UID` (User ID) и `GID` (Group ID) — идентификаторы пользователя и группы соответственно. Идентификатор пользователя root всегда равен 0, как и идентификатор группы root. Список групп вы найдете в файле /etc/groups;
- пятое поле — это настоящее имя пользователя. Поле может быть не заполнено, а может содержать фамилию, имя и отчество пользователя, — все зависит от педантичности администратора системы, т. е. от вас. Если вы работаете за компьютером в гордом одиночестве, то, думаю, свою фамилию вы не забудете. А вот если ваш компьютер — сервер сети, тогда просто необходимо указать Ф.И.О.

каждого пользователя, а то, когда придет время обратиться к пользователю по имени, вы его знать не будете (попробуйте запомнить 500 фамилий и имен!);

- шестое поле содержит имя домашнего каталога. Обычно это каталог /home/*<имя_пользователя>*;
- последнее поле — это имя командного интерпретатора, который будет обрабатывать введенные вами команды, когда вы зарегистрируетесь в консоли.

В целях повышения безопасности пароли в свое время были перенесены из файла /etc/passwd в файл /etc/shadow (доступен для чтения/записи только пользователю root), где они и хранятся в закодированном виде.

Во-вторых, при создании пользователя создается каталог /home/*<имя пользователя>*, в который копируется содержимое каталога /etc/skel. Каталог /etc/skel содержит «джентльменский набор» — файлы конфигурации по умолчанию, которые должны быть в любом пользовательском каталоге. Название каталога skel (от skeleton) полностью оправдывает себя — он действительно содержит «скелет» домашнего каталога пользователя.

24.5. Группы пользователей

Иногда пользователей объединяют в *группы*. Группы позволяют более эффективно управлять правами пользователей. Пусть над каким-либо проектом у вас должны совместно работать три разных пользователя — их достаточно объединить в одну группу, и тогда они получат доступ к домашним каталогам друг друга (по умолчанию пользователи не имеют доступа к домашним каталогам других пользователей, поскольку считается, что они находятся в разных группах).

Создать группу, а также поместить пользователя в группу позволяют графические конфигураторы. Вы можете использовать их — они очень удобные, но если вы хотите стать настоящим линуксоидом, то должны знать, что доступные в системе группы указываются в файле /etc/group. Добавить новую группу в систему можно с помощью команды `groupadd`, но, как правило, проще в текстовом редакторе добавить еще одну запись в файл /etc/group, а изменить группу пользователя еще легче — для этого достаточно отредактировать файл /etc/passwd.

24.6. Конфигуратор *Политика безопасности*

Для управления настройками безопасности, а также учетными записями пользователей и групп используется конфигуратор **Политика безопасности** (рис. 24.6), который можно вызвать через меню **Системные**.

Использовать этот конфигуратор достаточно просто, как и любое графическое приложение. Рассмотрим последовательность действий для создания учетной записи пользователя:

1. Перейдите в раздел **Пользователи**, нажмите кнопку **+**.
2. Заполните форму создания пользователя (рис. 24.7) — введите имя пользователя, а также заполните остальные поля при необходимости.

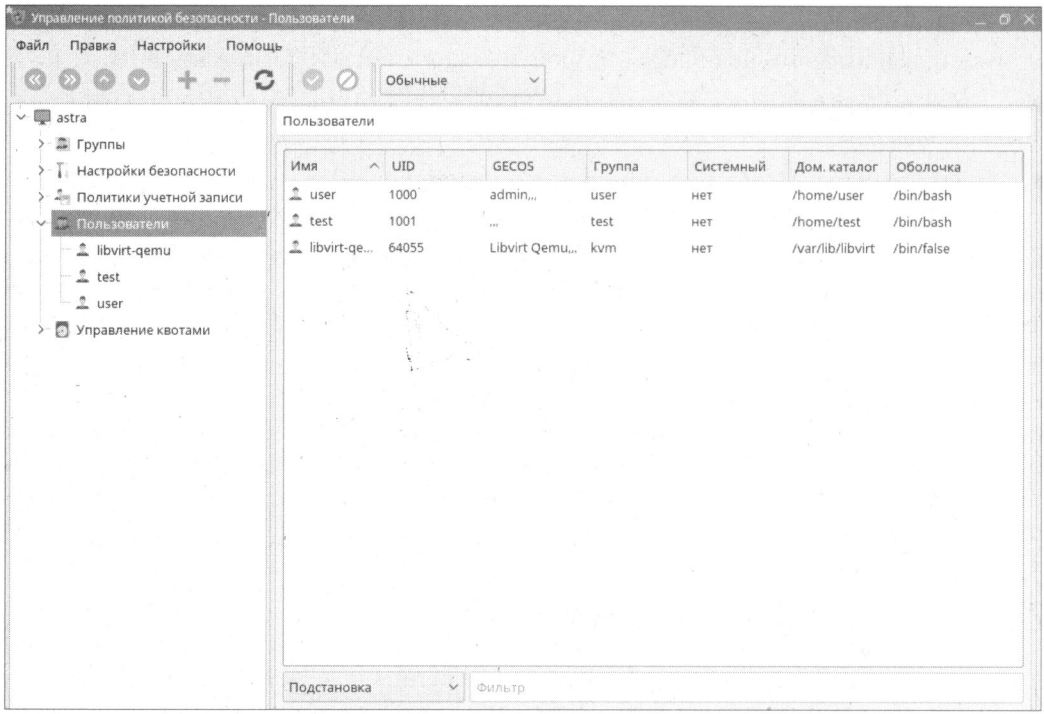

Рис. 24.6. Политика безопасности

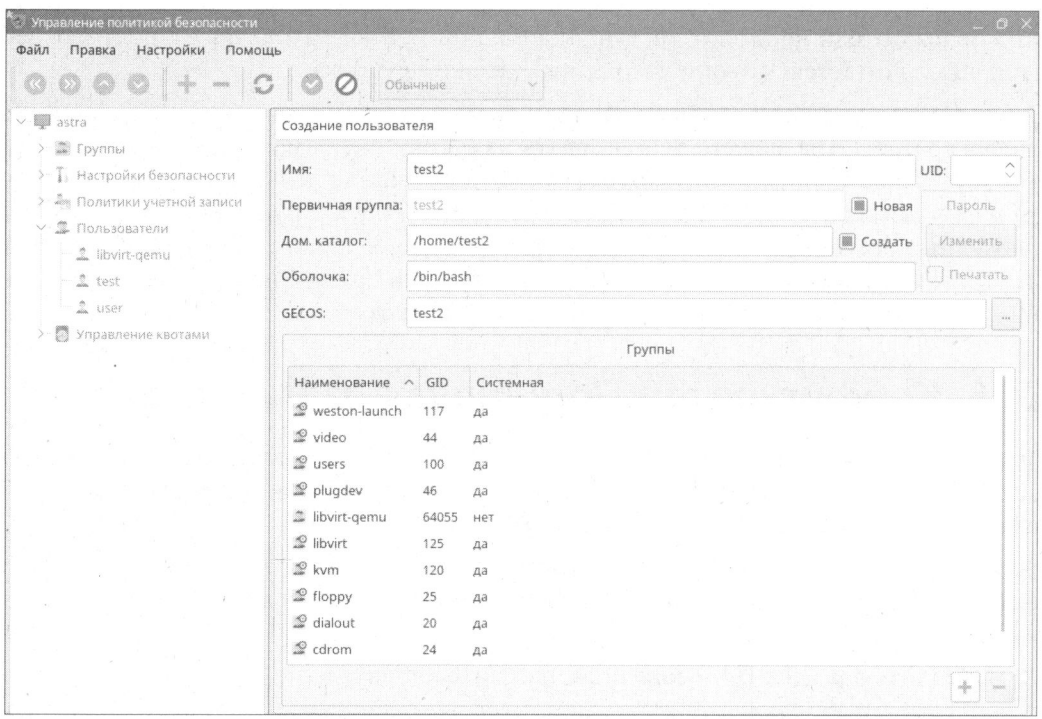

Рис. 24.7. Создание пользователя с помощью графического конфигуратора

3. Нажмите зеленую галку в верхней части окна — конфигуратор запросит пароль нового пользователя (рис. 24.8). Введите его и нажмите кнопку **Да**.
4. Повторите ввод пароля для подтверждения правильности ввода и нажмите кнопку **Да**. Еще раз нажмите кнопку **Да** в окне **Смена пароля**.
5. Убедитесь, что новый пользователь появился в разделе **Пользователи** окна **Политика безопасности**.

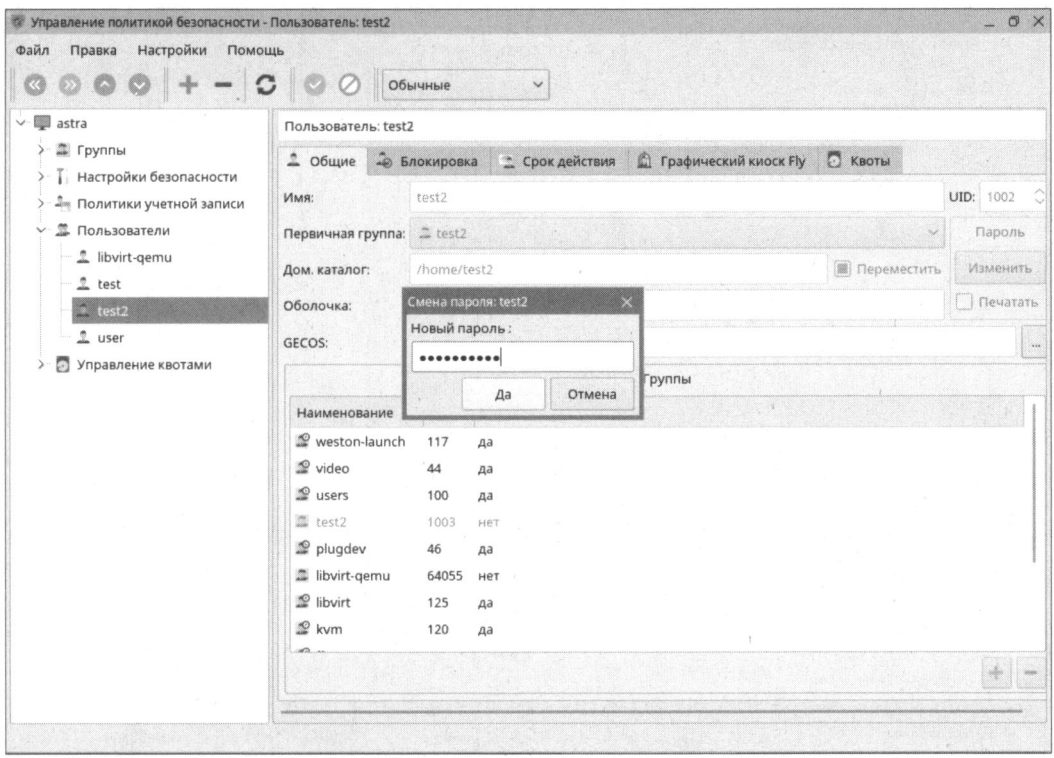

Рис. 24.8. Установка пароля пользователя

Для изменения учетной записи пользователя выполните следующие действия:

1. Выберите учетную запись в разделе **Пользователи** в левой части окна конфигуратора.
2. На вкладке **Общие** (рис. 24.9) можно изменить общие параметры пользователя, а также его пароль, нажав кнопку **Изменить**. Здесь же можно добавить пользователя в ранее созданную группу. Для этого нажмите кнопку **+** и выберите группу, в которую собираетесь добавить пользователя. Если нужно сделать пользователя администратором (чтобы у него была возможность использования команды `sudo`), добавьте его в группу astra-admin.
3. Перейдите на вкладку **Блокировка** (рис. 24.10). Здесь вы можете просмотреть/сбросить счетчик неудачных входов пользователя, задать максимальное количество неудачных попыток входа (после превышения которого учетная запись

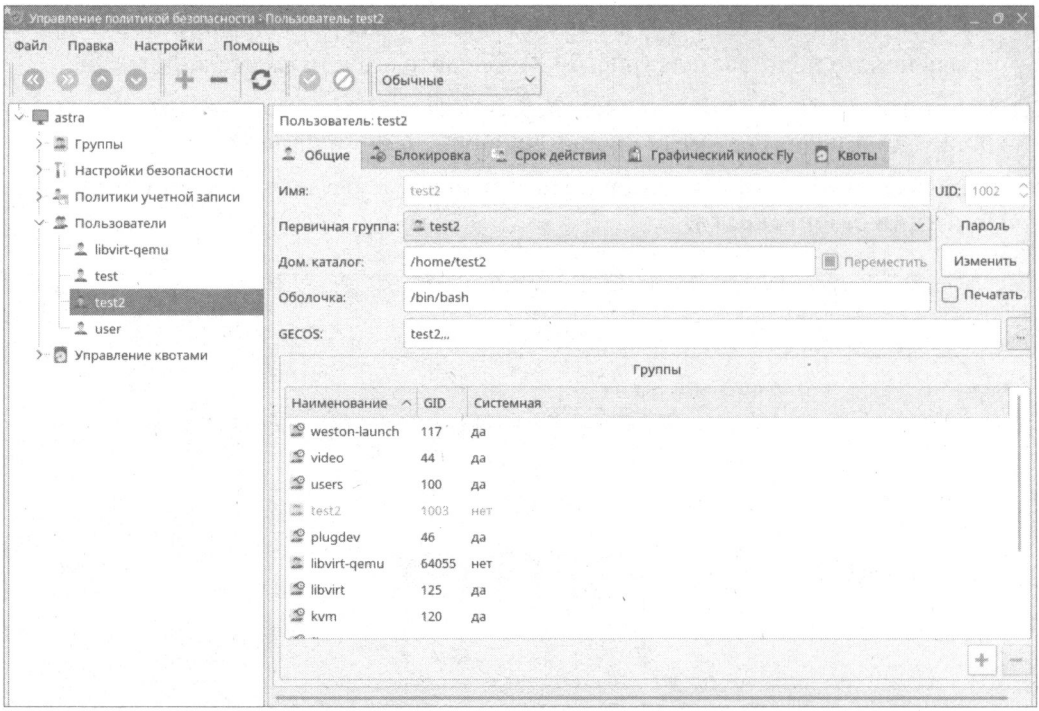

Рис. 24.9. Изменение общих параметров пользователя

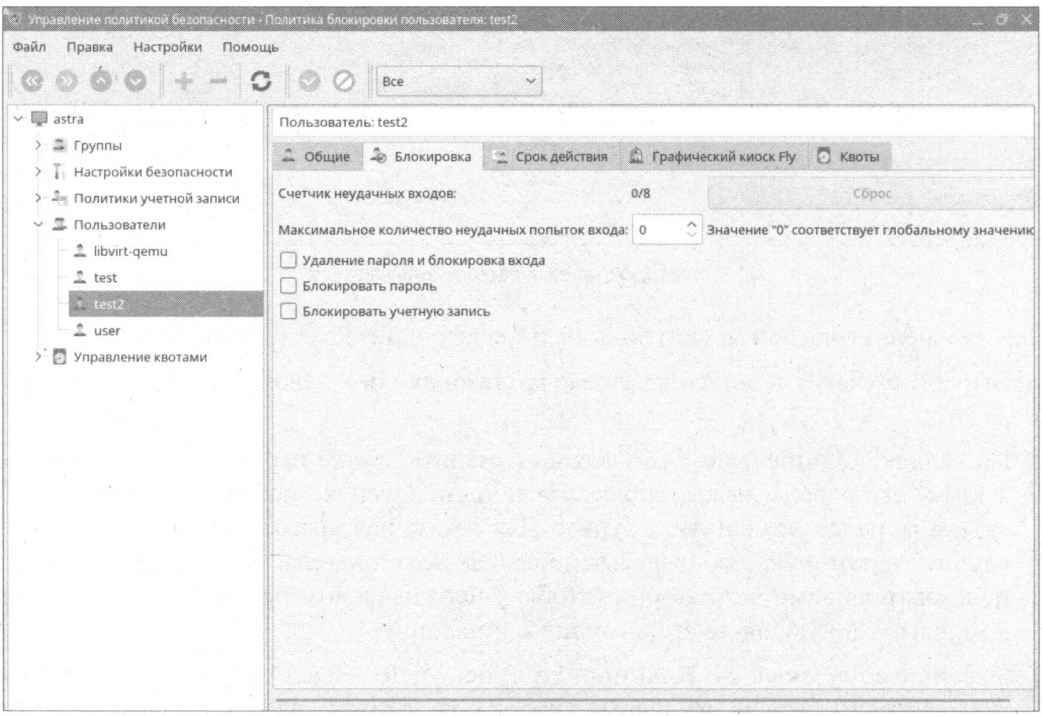

Рис. 24.10. Блокировка учетной записи

будет заблокирована), а также заблокировать учетную запись. Для блокировки доступны три варианта:

- **Удаление пароля и блокировка входа** — пароль пользователя будет удален, учетная запись заблокирована. После разблокирования нужно будет установить новый пароль пользователя;
- **Блокировать пароль** — включает блокировку пароля;
- **Блокировать учетную запись** — учетная запись будет заблокирована. После разблокировки пользователь сможет войти как обычно.

4. Перейдите на вкладку **Срок действия** (рис. 24.11) и задайте срок действия учетной записи. Следует отметить, что для домашнего использования параметры на этой вкладке не важны — разве что вы таким образом можете заставить себя сменить пароль, но будьте осторожны, чтобы самому не заблокировать себя.

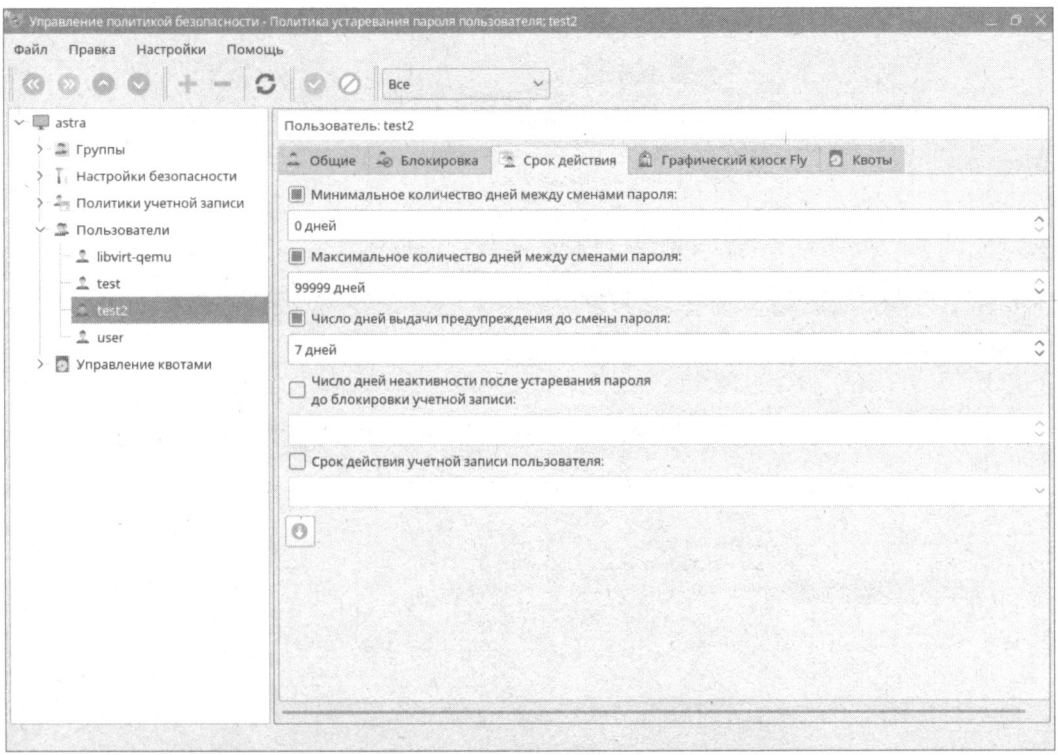

Рис. 24.11. Срок действия учетной записи

5. Перейдите на вкладку **Графический киоск Fly** (рис. 24.12) — эта вкладка как раз будет довольно полезной для домашнего применения, особенно если за вашим компьютером работает ребенок. Здесь вы сможете определить, какие приложения ему можно запускать.
6. Вкладка **Квотирование** содержит параметры, позволяющие задать ограничения использования дискового пространства (рис. 24.13).

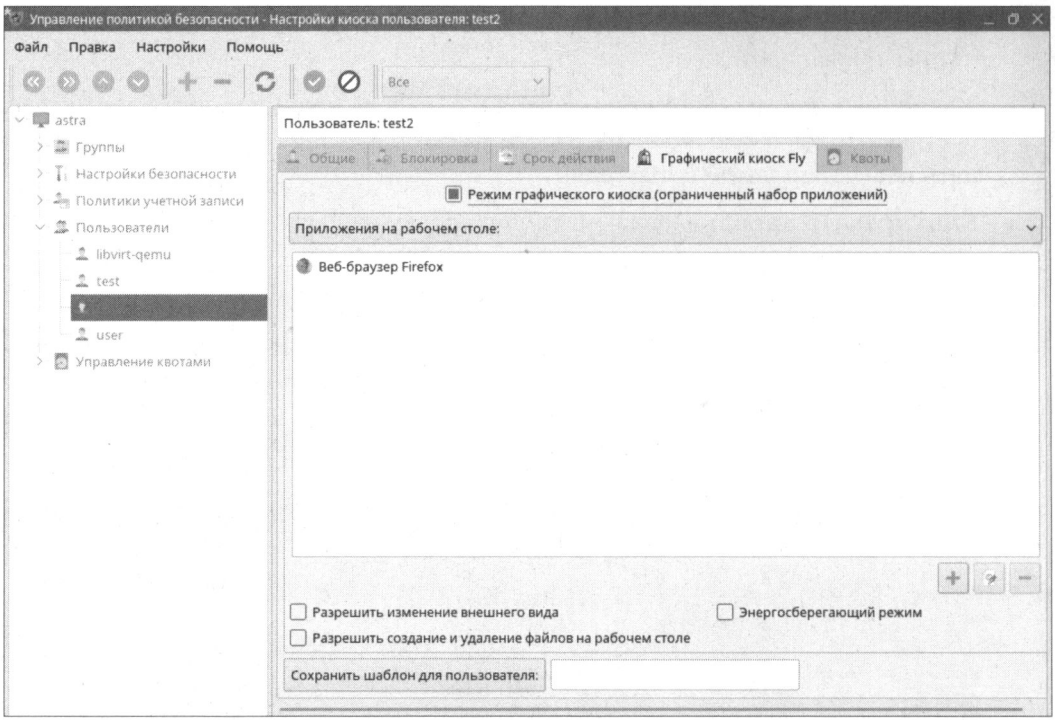

Рис. 24.12. Графический киоск Fly

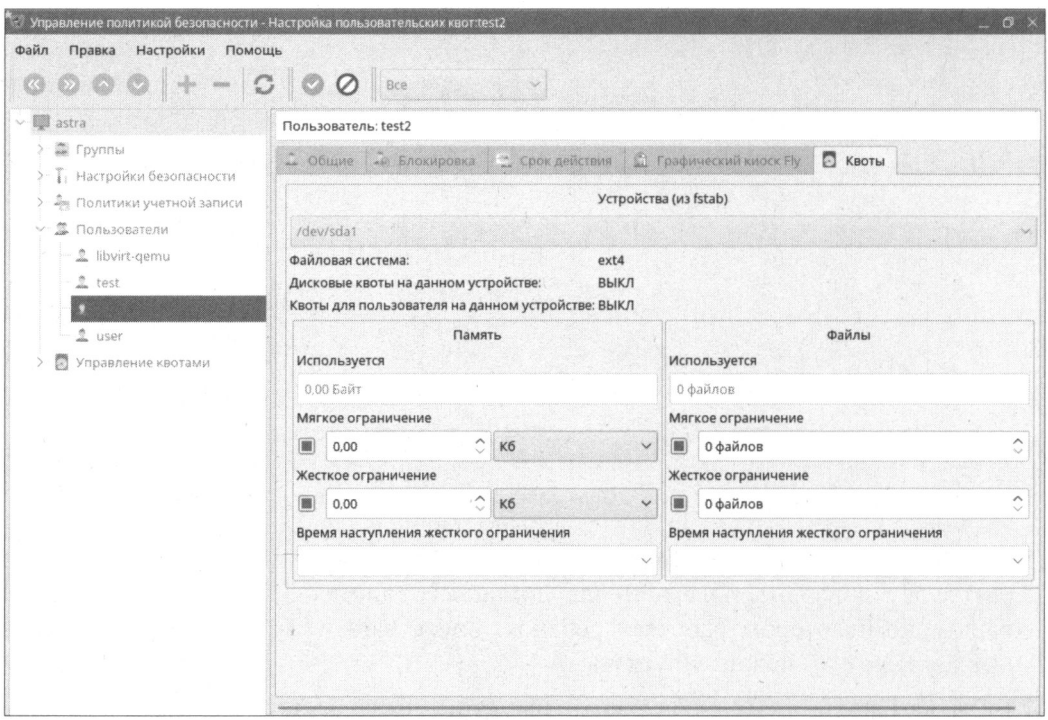

Рис. 24.13. Квотирование

Квотирование

Квотирование — это механизм ограничения дискового пространства пользователей. Linux — система многопользовательская, поэтому без ограничения дискового пространства здесь не обойтись. Когда вы используете компьютер в гордом одиночестве, то все дисковое пространство доступно вам и только вам. А вот когда пользователей несколько, нужно ограничить доступное пространство, чтобы один из пользователей не «узурпировал» все место на диске. Как именно вы будете ограничивать дисковое пространство, решать только вам — можно поделить дисковое пространство поровну между пользователями, можно одним пользователям отдать больше места, а другим — меньше. Как правило, этот механизм целесообразно использовать на сервере крупных компаний, а также на серверах хостинг-провайдеров. На домашнем компьютере квотирование бессмысленно — нет смысла ограничивать самого себя.

Далее мы рассмотрим самые полезные настройки, которые вы можете сделать в приложении **Политика безопасности**. С его помощью можно задать множество разных параметров, но мы рассмотрим только самые полезные из них в контексте офисного/домашнего применения.

24.6.1. Включение ввода пароля для *sudo*

Как уже отмечалось ранее, во всех дистрибутивах Linux команда `sudo` требует ввода пароля пользователя. Это гарантирует, что права root не будут получены пользователем, который случайно оказался за вашим компьютером, например когда вы отошли на обед. В Astra Linux команда `sudo`, как также отмечено ранее, не требует ввода пароля. Но если вы все же хотите включить это требование, выполните следующие действия:

1. Запустите приложение **Политика безопасности**.
2. Перейдите в раздел **Настройки безопасности | Политика консоли и интерпретаторов**.
3. Включите параметр **Включить ввод пароля для sudo**.
4. Нажмите зеленую кнопку **Применить** в верхней части окна.

24.6.2. Блокировка перезагрузки и выключения для пользователей

Если вы не хотите, чтобы обычные пользователи (например, ваш ребенок или коллеги по офису, которым вы предоставили доступ к своему компьютеру) могли выключать компьютер или перезагружать его, выполните следующие действия:

1. Запустите приложение **Политика безопасности**.
2. Перейдите в раздел **Настройки безопасности | Системные параметры**.
3. Включите параметр **Блокировка выключения/перезагрузки ПК для пользователей**.
4. Нажмите зеленую кнопку **Применить** в верхней части окна.

24.6.3. Задание глобального счетчика неудачных попыток входа

Для изменения глобального счетчика неудачных попыток входа выполните следующие действия:

1. Запустите приложение **Политика безопасности**.
2. Перейдите в раздел **Политики учетной записи | Блокировка**.
3. Включите параметр **Неуспешных попыток** и задайте для него нужное значение.
4. Нажмите зеленую кнопку **Применить** в верхней части окна.

24.6.4. Задание сложности пароля

Для задания минимальной длины пароля и других параметров сложности выполните следующие действия:

1. Запустите приложение **Политика безопасности**.
2. Перейдите в раздел **Политики учетной записи | Политики паролей | Сложность**.
3. Установите новое значение для параметра **Минимальная длина пароля**. При желании вы можете включить другие параметры сложности: **Минимальное количество строчных/заглавных букв в новом пароле**, **Минимальное количество цифр в новом пароле** и **Минимальное количество других символов в новом пароле**.
4. Нажмите зеленую кнопку **Применить** в верхней части окна.

ASTRA LINUX

ГЛАВА 25

Ограничение возможностей пользователей. Системный киоск

- Концепция киоска
- Настройка правил для пользователя
- Создание собственных профилей

25.1. Концепция киоска

Любой администратор знаком с концепцией песочницы. Песочница в ИТ является ее полным аналогом в реальном мире: ребенка помещают в песочницу, чтобы он мог играть, и чтобы песок не рассыпался за пределы песочницы — на детскую площадку. О том, что дети могут выбрасывать песок из песочницы, а котики — использовать песочницу не по назначению, промолчу — это естественные издержки. Теоретически процесс в ИТ также может выйти за пределы песочницы, но в идеале этого не должно произойти.

В Astra Linux есть свой аналог песочницы, который называется *системный киоск*. В качестве ребенка выступает обычный пользователь, а администратор при формировании правил для пользователя указывает, что разрешено делать пользователю, а что — нет. В этом и есть принцип системного киоска.

Системный киоск доступен, как в Special, так и в Common Edition, так что пользователи Common Edition не будут чувствовать себя обделенными. Далее мы посмотрим, как использовать киоск.

Зачем это нужно? Представим, что у вас есть небольшой офис, и вы хотите, чтобы каждый из сотрудников занимался своими обязанностями. У вас есть секретарша, и вы не хотите, чтобы она использовала Firefox (нет у нее обязанностей по поиску информации в Интернете), но ей нужно разрешить использовать LibreOffice, так как это связано непосредственно с ее деятельностью. Такие ограничения можно настроить с помощью правил киоска.

25.2. Настройка правил для пользователя

Откройте **Панель управления**, перейдите в раздел **Безопасность**, запустите конфигуратор **Системный киоск**. Нажмите здесь кнопку **+** и выберите пользователя из списка (рис. 25.1). Если ранее уже были заданы правила для кого-то из пользователей, вы можете активировать флажок **Копировать правила другого пользователя** и выбрать этого пользователя из списка.

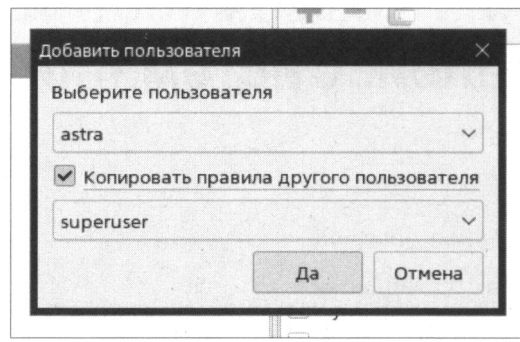

Рис. 25.1. Добавление пользователя в киоск

Добавив пользователя в киоск, в области **Профили** вы можете задать для него набор правил (рис. 25.2).

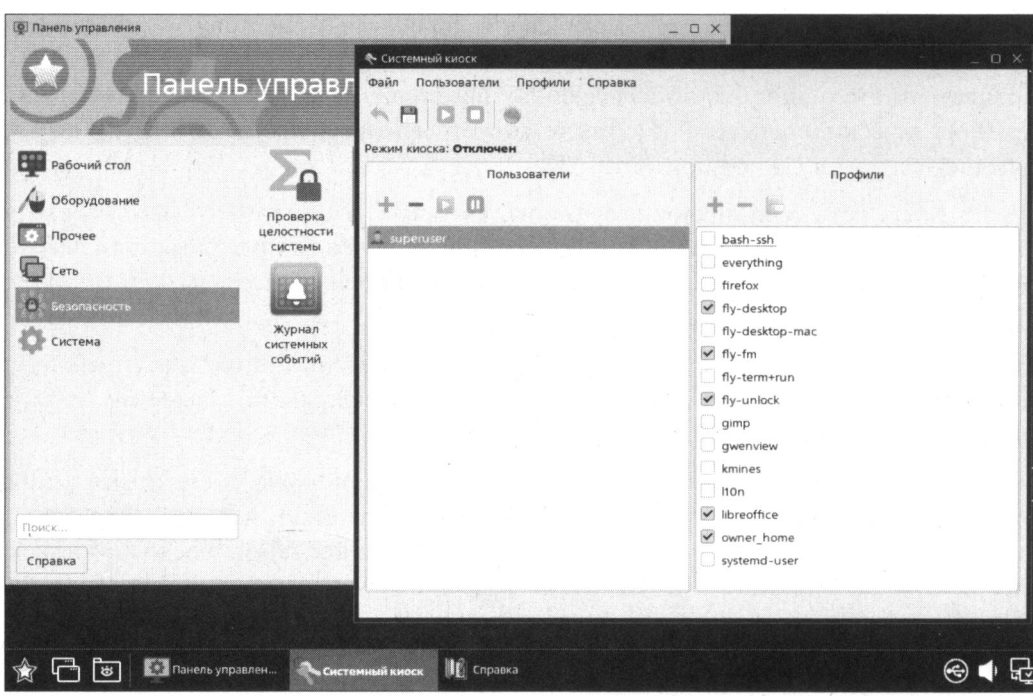

Рис. 25.2. Добавление правил для пользователя

По умолчанию доступны следующие профили:

- bash-ssh — управляет доступом к сессии при подключении через SSH. Если этот профиль выключен, пользователь не может пойти в систему по SSH;
- everything — позволяет работать со всеми программами и файлами системы в соответствии с установленными правами доступа. Если вы хотите ограничить пользователя, не включайте этот профиль;
- fly-desktop — позволяет использовать графический вход в систему. В большинстве случаев нужно включить этот профиль;
- fly-desktop-mac — разрешает выполнять графический вход в систему на ненулевом уровне конфиденциальности;
- fly-fm — разрешает использовать менеджер файлов fly-fm;
- fly-term+run — разрешает работать в графическом терминале Fly;
- fly-unlock — позволяет выполнять разблокировку экрана;
- gimp — позволяет использовать графический редактор GIMP;
- gwenview — разрешает работать в программе просмотра изображений Gwenview;
- kmines — позволяет запускать программу kmines;
- l10n — необходим для установки региональных настроек;
- libreoffice — позволяет работать в программах LibreOffice;
- owner_home — предоставляет доступ к домашним каталогам пользователей в соответствии с установленными правами доступа;
- systemd-user — позволяет запускать пользовательские службы и управлять ими.

Как показано на рис. 25.2, для пользователя включены профили fly-desktop (разрешает использовать графический интерфейс), fly-fm (разрешает использовать файловый менеджер), owner_home (разрешает использовать домашний каталог — это важно!), fly-unlock (разрешает разблокировать экран, если он заблокировался по какой-то причине), libreoffice (разрешает использование LibreOffice). По сути, это как раз и есть набор правил для секретарши, о котором мы говорили ранее.

25.3. Создание собственных профилей

Набор системных профилей скудный. А что будет, если администратор установил другой браузер — например, Chromium, но профиля для него нет. Получается, Firefox пользователю запускать нельзя, а Chromium — можно? К счастью, есть возможность определить собственные профили. Для этого:

1. Выполните команду меню **Профили | Создать профиль**.
2. В появившейся панели (рис. 25.3) выберите команду, для которой вы хотите создать профиль. Можно нажать кнопку **Из меню** и выбрать программу из меню, если она там есть. Для некоторых программ, выполнение которых не

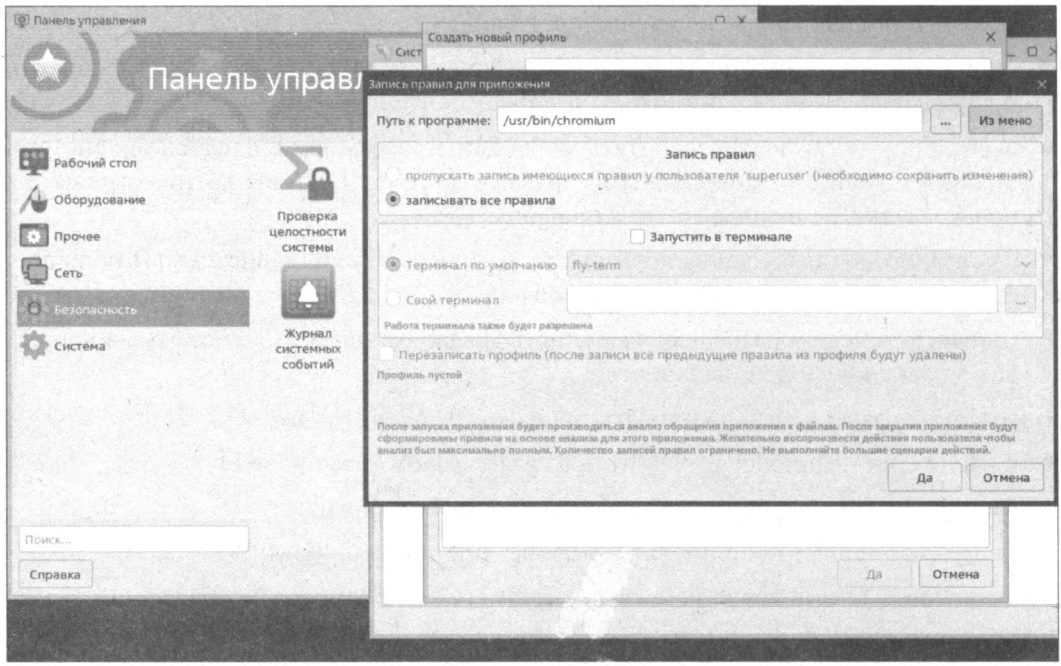

Рис. 25.3. Выбор программы для создания профиля

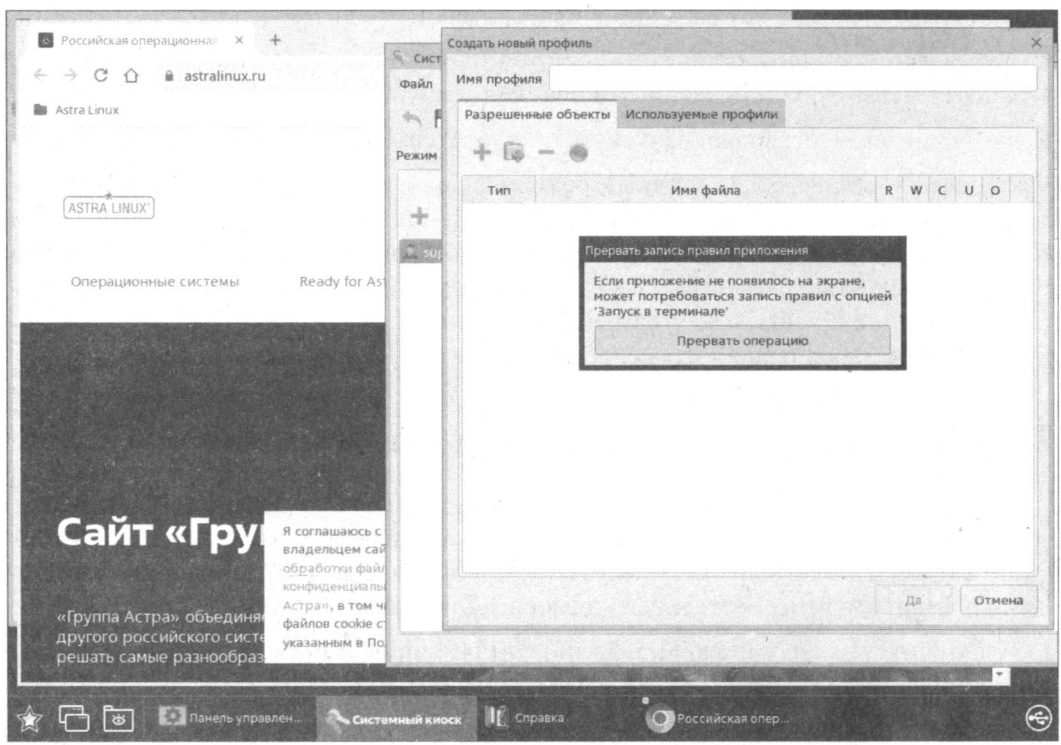

Рис. 25.4. Трассировка программы

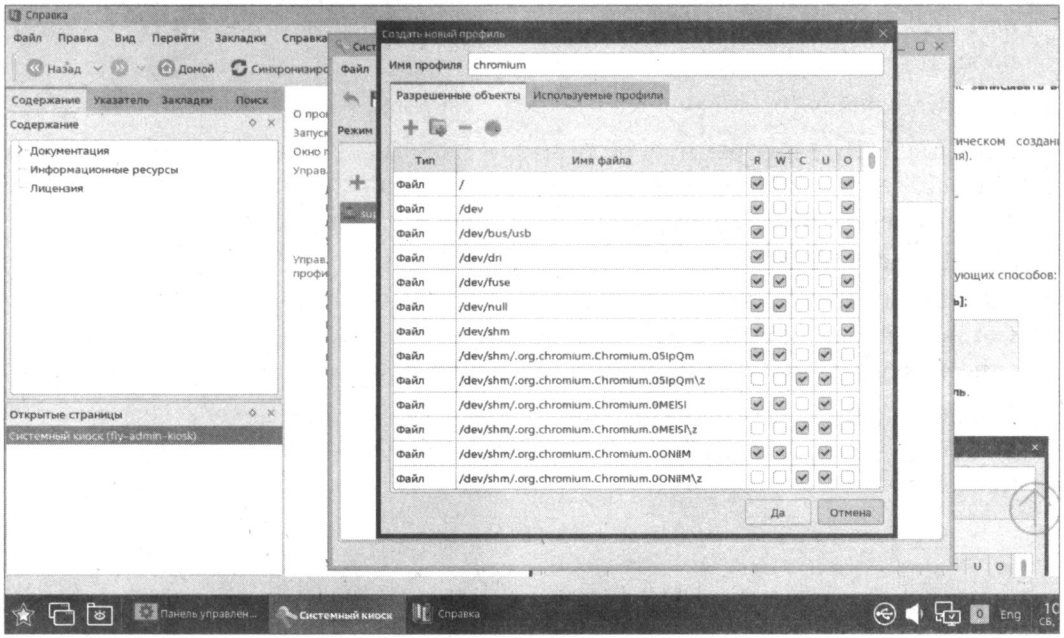

Рис. 25.5. Результаты трассировки

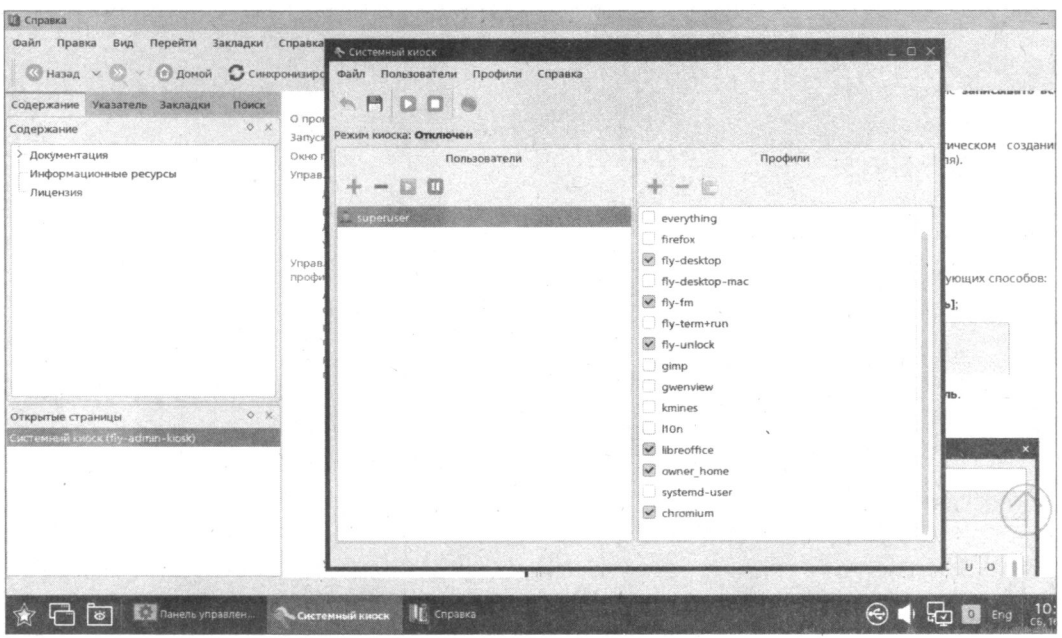

Рис. 25.6. Профиль Chromium создан

начнется после нажатия кнопки **Да**, нужно включить режим **Запустить в терминале**.

3. Нажмите кнопку **Да** — начнется режим трассировки программы.
4. Используйте эту программу в режиме трассировки. Например, если вы выбрали браузер, как в нашем примере, посещайте различные сайты — в общем, выполняйте привычные действия. Чем больше действий вы выполните, тем более подробной будет трассировка (рис. 25.4).
5. Завершите работу программы. Теперь стоит немного подождать, пока в окне **Создать новый профиль** не появятся результаты трассировки (рис. 25.5) — это список ресурсов, к которым обращалась программа.
6. Введите название профиля и нажмите кнопку **Да** — в списке профилей появится новый профиль, который можно включить или выключить для пользователя (рис. 25.6).

ASTRA LINUX®

ГЛАВА 26

Мандатный контроль целостности (МКЦ)

- Принцип работы МКЦ
- Включение МКЦ и назначение уровня целостности файлам и каталогам
- Назначение уровня целостности пользователю

26.1. Принцип работы МКЦ

В состав Astra Linux Special Edition входит компонент ограничения доступа к файлам и каталогам — мандатный контроль целостности (МКЦ). Этот компонент отвечает за то, чтобы информацию не могли изменять пользователи, которым это не положено. Другими словами, в дополнение к традиционным правам доступа вводится еще одно ограничение — на случай, если кто-то получит максимальные полномочия (пользователь или процесс) и попытается получить доступ к важным файлам.

Суть МКЦ в том, что пользователь или процесс, работающий на своем уровне целостности, может изменять только файлы и каталоги, у которых такой же уровень целостности или ниже. Основных уровней целостности два: 0 (низкий) и 63 (высокий). В документации по Astra Linux упоминается еще один уровень — 127 («Брест»), но это специальный уровень для файлов, которые никогда не должны быть изменены.

Если у пользователя низкий уровень целостности (0), то он не сможет изменить файлы и каталоги, для которых назначен уровень целостности (63). При включенном МКЦ каждому файлу и каталогу назначается уровень целостности — например, у каталога /etc это 63, у /tmp — 0. Пользователь с уровнем 0 не сможет изменить конфигурацию системы, которая хранится в каталоге /etc, но зато процессы, работающие от имени этого пользователя, смогут записывать временные файлы в каталог /tmp.

26.2. Включение МКЦ и назначение уровня целостности файлам и каталогам

Запустите конфигуратор **Управление политикой безопасности** и перейдите в раздел **Мандатный контроль целостности** (рис. 26.1). Сняв на вкладке **Управление** флажок **Подсистема Мандатного Контроля Целостности**, вы выключите подсистему МКЦ, если она вам не нужна. По умолчанию она включена.

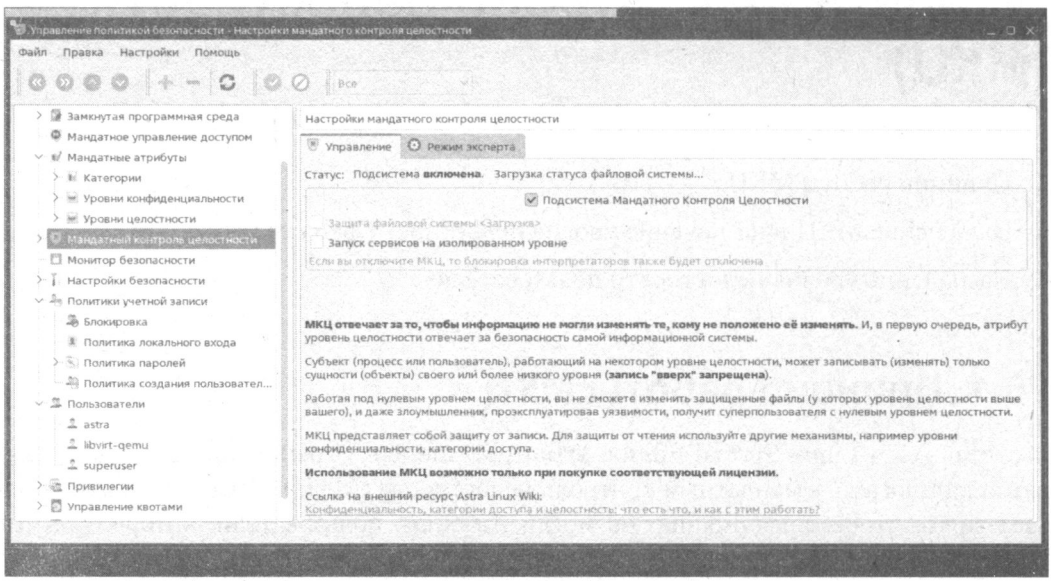

Рис. 26.1. Управление статусом МКЦ

Вкладка **Режим эксперта** (рис. 26.2) разрешает изменять файлы и каталоги. Всё здесь достаточно просто — щелкните на файле или каталоге в дереве файловой системы, выберите другой уровень целостности, а затем нажмите кнопку **Применить**.

С уровнем 127 («Брест») нужно быть предельно осторожным. Вы можете назначить его для файлов и каталогов, но не можете указать в качестве максимального уровня целостности для кого-либо из пользователей. Этим и обеспечивается, что уровень «Брест» используется для тех файлов и каталогов, которые ни при каких условиях не должны быть изменены.

26.3. Назначение уровня целостности пользователю

Запустите конфигуратор **Управление политикой безопасности**, перейдите в раздел **Пользователи** и выберите учетную запись пользователя. Здесь на вкладке **МРД** (рис. 26.3) вы сможете назначить для этого пользователя минимальный и

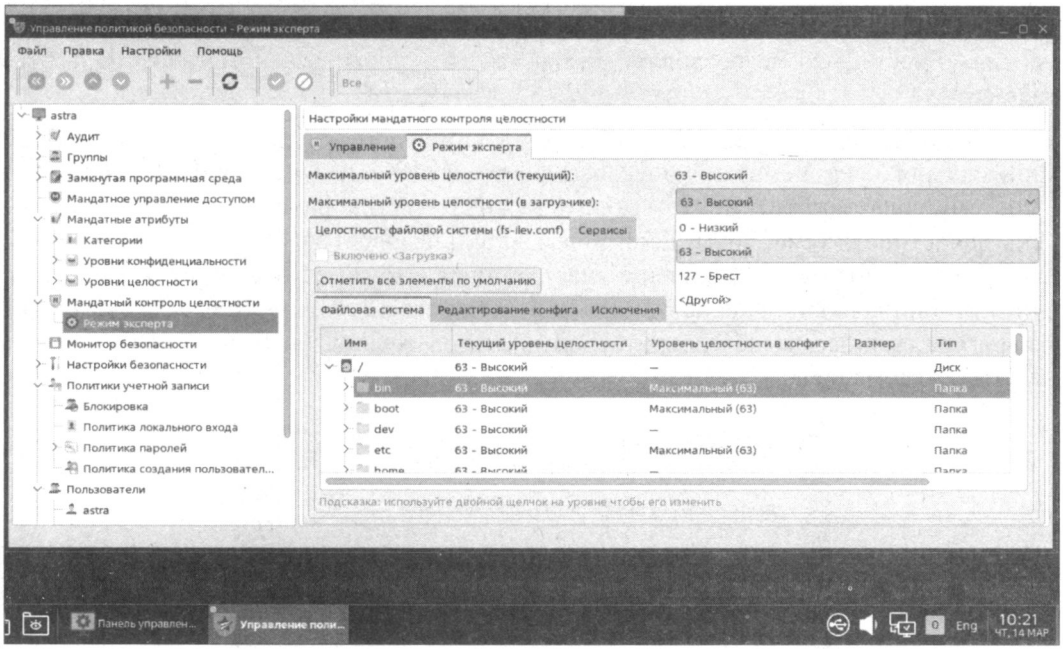

Рис. 26.2. Режим эксперта МКЦ

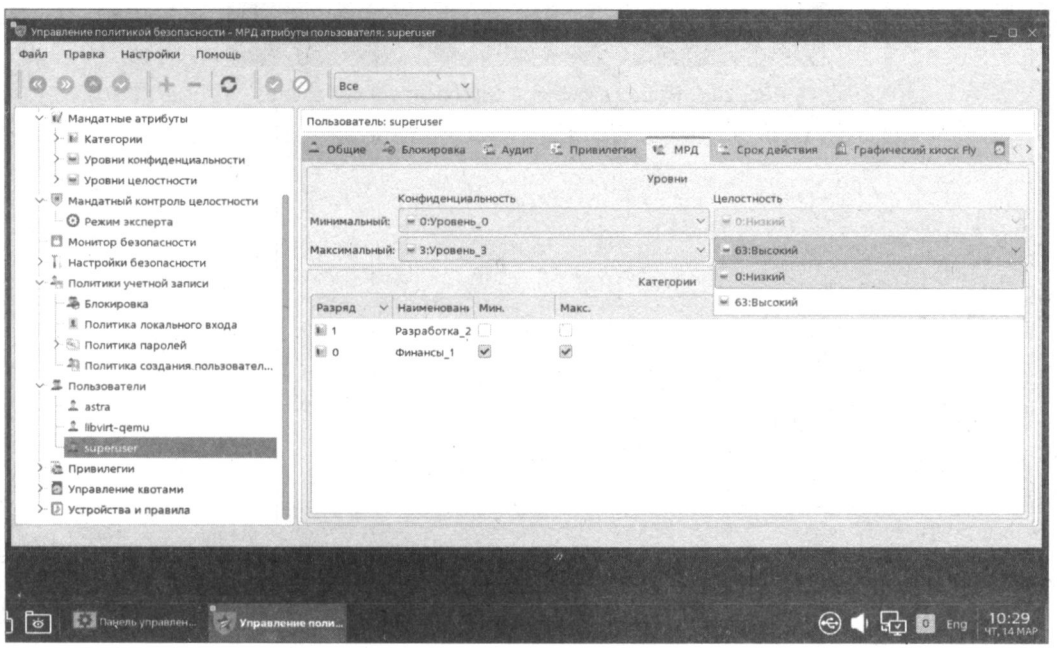

Рис. 26.3. Назначение уровня целостности для пользователя

максимальный уровни целостности. Как видите, в списке доступных уровней нет уровня «Брест», что не позволяет назначить этот уровень какому-либо пользователю.

Если для пользователя назначены разные уровни целостности, например в качестве минимального указан низкий, а в качестве максимального — высокий, то при входе в систему пользователь сможет выбрать нужный ему для работы в этом сеансе уровень целостности (рис. 26.4). Если пользователь выберет высокий уровень целостности, интерфейс будет окрашен в красный цвет, что будет постоянно напоминать ему о потенциальной опасности, — ведь у него есть возможность изменять важные конфигурационные (и не только) файлы, следовательно, он потенциально может нанести системе вред.

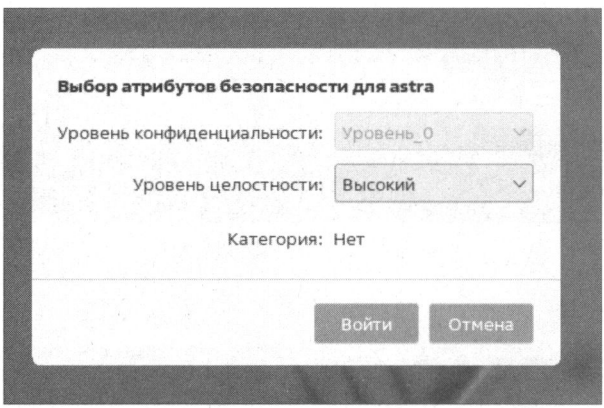

Рис. 26.4. Выбор атрибутов безопасности при входе

ASTRA LINUX®

ГЛАВА 27

Мандатное управление доступом (МРД)

- Введение в МРД
- Определение уровней конфиденциальности
- Определение категорий конфиденциальности
- Назначение уровней и категорий конфиденциальности учетным записям пользователей
- Назначение привилегий пользователям
- Установка классификационной метки файла/каталога

27.1. Введение в МРД

Уровень «Смоленск» дополняет возможности режима «Воронеж» функциями мандатного управления доступом (МРД) для защиты от угрозы конфиденциальности информации. Смысл МРД в следующем:

- читать данные из файла/каталога могут только те процессы и пользователи, которые обладают не меньшим уровнем конфиденциальности (не целостности, а именно конфиденциальности!), чем у запрашиваемых данных;
- записывать данные в файл или каталог могут только процессы с уровнем конфиденциальности, равным или меньшим, чем у этого файла/каталога.

При входе в систему пользователь может выбрать уровень конфиденциальности и уровень целостности (рис. 27.1). Разумеется, это только в том случае, если администратор задал для него максимальный и минимальный уровни, иначе диалоговая панель **Выбор атрибутов безопасности** появляться не будет.

Основная задача МРД — ограничить доступ к информации, чтобы она не попала к тем пользователям, у которых нет соответствующих полномочий. Для реализации ограничения доступа используются мандатные атрибуты уровень конфиденциальности и категория конфиденциальности. *Уровень конфиденциальности* позволяет задать степень секретности — например, «Не секретно», «Секретно» и «Совершенно секретно». Понятно, что в любой организации есть персонал, которому разре-

шено работать только с несекретными документами, и у него не должно быть доступа к документам уровней конфиденциальности «Секретно» и «Совершенно секретно». Основная суть назначения уровней конфиденциальности — чтобы пользователь не смог читать документы, которые помечены более высокими уровнями конфиденциальности.

Рис. 27.1. Выбор атрибутов безопасности

Уровень конфиденциальности

Уровень конфиденциальности определяет степень секретности объекта доступа (файл, каталог) и соответствующий ему уровень доступа, назначенный субъекту (пользователю и процессу, который работает от имени пользователя). Субъект с определенным уровнем конфиденциальности может читать документы с таким же уровнем или ниже. Ему запрещено читать документы с более высоким уровнем конфиденциальности, что исключает намеренную или случайную утечку информации, к которой у субъекта не должно быть доступа.

Однако документов, помеченных как «Секретно» и «Совершенно секретно», может быть очень много. Встает вопрос: как более точно разграничить доступ внутри этих уровней? Например, в вашей организации есть несколько групп, работающих над разными направлениями. При этом каждая из групп не должна иметь доступ к документам другой группы. Однако каждая из групп — «совершенно секретна».

Для более тонкого разграничения доступа служат *категории конфиденциальности*. С их использованием вы создаете внутри каждого уровня конфиденциальности еще и набор категорий — например, по каждому из важных разрабатываемых в вашей организации продуктов, и назначаете пользователям категории конфиденциальности. Иногда одному пользователю приходится назначать несколько категорий конфиденциальности. Например, вы создали три категории: «ПродуктА», «ПродуктБ» и «ПродуктВ» — по одной для каждого продукта. К сожалению, хороших специалистов мало, поэтому у продуктов А и В один и тот же системный архитектор. Поэтому ему нужно назначить обе категории: «ПродуктА» и «ПродуктВ».

> ***Категория конфиденциальности***
>
> Пользователям назначаются категории конфиденциальности. Не имея категорию конфиденциальности, назначенную документу, пользователь не имеет права прочитать этот документ.

Кроме атрибутов уровень и категория конфиденциальности, используется еще один атрибут — уровень целостности, отвечающий за безопасность информационной системы и защищающий ее от несанкционированного доступа и вирусов.

> ***Уровень целостности***
>
> Пользователь или процесс, работающий на некотором уровне целостности, может записывать только объекты своего или более низкого уровня целостности. Это дополнительное средство разграничения доступа к файлам и каталогам. Уровень целостности относится к МКЦ и был подробно рассмотрен в *главе 26*. Если вы не прочитали ее до сих пор, самое время это сделать.

Для успешной настройки МРД нужно понимать, кто или что является субъектом и объектом мандатного доступа. Субъект — это тот, кто выполняет действие, т. е. пользователь или процесс. Объект — то, над чем выполняется действие, т. е. файл или каталог. Вводится и понятие контейнера — это объект, содержащий другие объекты (файлы или каталоги).

Операция записи разрешается в случае, если уровень конфиденциальности и категории конфиденциальности объекта и субъекта совпадают, а уровень целостности субъекта равен или выше уровня целостности объекта.

Операция чтения/выполнения разрешена, если уровень конфиденциальности субъекта не ниже уровня конфиденциальности объекта.

Также нужно учитывать правила наследования. Так, если субъект создает другой субъект, т. е. процесс порождает другой процесс, то созданный процесс полностью наследует метку безопасности родителя, т. е. наследуются все атрибуты: уровень и категория конфиденциальности и уровень целостности.

Если субъект создает объект (т. е. вы создали файл), то созданный объект наследует только уровень конфиденциальности и категорию конфиденциальности, но всегда получает нулевой уровень целостности.

Изменить классификационную метку безопасности (т. е. изменить уровень/категорию конфиденциальности) могут только субъекты с привилегией `PARSEC_CAP_CHMAC` (см. далее). А вот чтобы изменить метку объекта (т. е. изменить уровень

целостности), нужна не только привилегия PARSEC_CAP_CHMAC у субъекта, но и максимальный («Высокий») уровень целостности.

Ранее упоминалась метка безопасности — она же классификационная метка. В состав метки безопасности входят следующие мандатные атрибуты:

- уровень конфиденциальности;
- категория конфиденциальности;
- уровень целостности;
- дополнительные мандатные атрибуты управления доступом:
  - ccnr — этот атрибут может быть присвоен только контейнерам. Определяет, что контейнер может содержать файлы и каталоги с разными классификационными метками, но при этом все метки не должны превышать его собственную классификационную метку. Чтение содержимого контейнера разрешается субъектам, независимо от их классификационной метки, но при этом им доступна информация только про находящиеся в контейнере объекты с классификационной меткой, не превышающей классификационную метку субъекта. Так, если в контейнер добавлены документы уровней «Совершенно секретно» и «Секретно», читающий его пользователь с уровнем «Секретно» увидит только документы своего (или ниже, если они есть) уровня;
  - ehole — присваивается объектам, которые не являются контейнерами и имеют минимальную классификационную метку. Позволяет игнорировать мандатные правила управления доступом. Предназначен для сущностей с более высокой классификационной меткой, чем его собственная метка;
  - whole — присваивается объектам, которые не являются контейнерами и имеют максимальную классификационную метку. Разрешает запись в них субъектам, которые имеют более низкую классификационную метку;
  - привилегии PARSEC — назначаются субъектам и предоставляют права выполнения определенных действий (табл. 27.1).

Таблица 27.1. *Привилегии Parsec, предоставляющие права выполнения административных действий*

Привилегия	Описание
PARSEC_CAP_CAP	Разрешает процессу устанавливать произвольный непротиворечивый набор привилегий для себя
PARSEC_CAP_CHMAC	Разрешает изменять метки безопасности файловых объектов
PARSEC_CAP_AUDIT	Предоставляет возможность управлять политикой аудита
PARSEC_CAP_BYPASS_XATTR	Разрешает процессу игнорировать подписи файлов
PARSEC_CAP_CCNR_RELAX	Разрешает осуществлять в каталоге с установленным атрибутом ccnr действия над вложенными объектами с метками не выше метки этого каталога

Таблица 27.1 (окончание)

Привилегия	Описание
PARSEC_CAP_UNSAFE_SETXATTR	Разрешает субъекту устанавливать мандатные атрибуты объектов файловой системы без учета мандатных атрибутов родительского объекта-контейнера
PARSEC_CAP_SUMAC	Разрешает запускать процессы с другой классификационной меткой
PARSEC_CAP_SIG	Разрешает посылать сигналы процессам, игнорируя мандатные права
PARSEC_CAP_SETMAC	Разрешает процессу изменять (повышать или понижать) значения атрибутов собственной классификационной метки
PARSEC_CAP_READSEARCH	Разрешает игнорировать мандатную политику при чтении и поиске файловых объектов
PARSEC_CAP_PROCFS	Разрешает игнорировать ограничения МРД и МКЦ на файловой системе /proc
PARSEC_CAP_PRIV_SOCK	Разрешает создавать новые сетевые сокеты процесса в привилегированном режиме
PARSEC_CAP_MAC_SOCK	Разрешает изменять метки безопасности точек соединения
PARSEC_CAP_IPC_OWNER	Деактивирует мандатные ограничения при работе с сущностями межпроцессного взаимодействия, такими как разделяемая память, очередь сообщений и т. д.
PARSEC_CAP_IGNMACCAT	Разрешает игнорировать мандатную политику по неиерархическим категориям конфиденциальности
PARSEC_CAP_IGNMACINT	Разрешает игнорировать мандатную политику по уровням целостности
PARSEC_CAP_IGNMACLVL	Разрешает игнорировать мандатную политику по уровням конфиденциальности

27.2. Определение уровней конфиденциальности

Настроить уровни и категории конфиденциальности можно с помощью конфигуратора панели управления **Управление политикой безопасности**. Вы можете вызвать его через Панель управления или же выполнить команду `fly-admin-smc`. После запуска конфигуратора надо перейти в раздел **Мандатные атрибуты | Уровни конфиденциальности** (рис. 27.2).

Здесь вы можете добавить новые уровни, а также переименовать уже созданные четыре уровня. Для переименования уровня конфиденциальности выполните на нем двойной щелчок, введите новое название и нажмите зеленую кнопку **Применить**  .

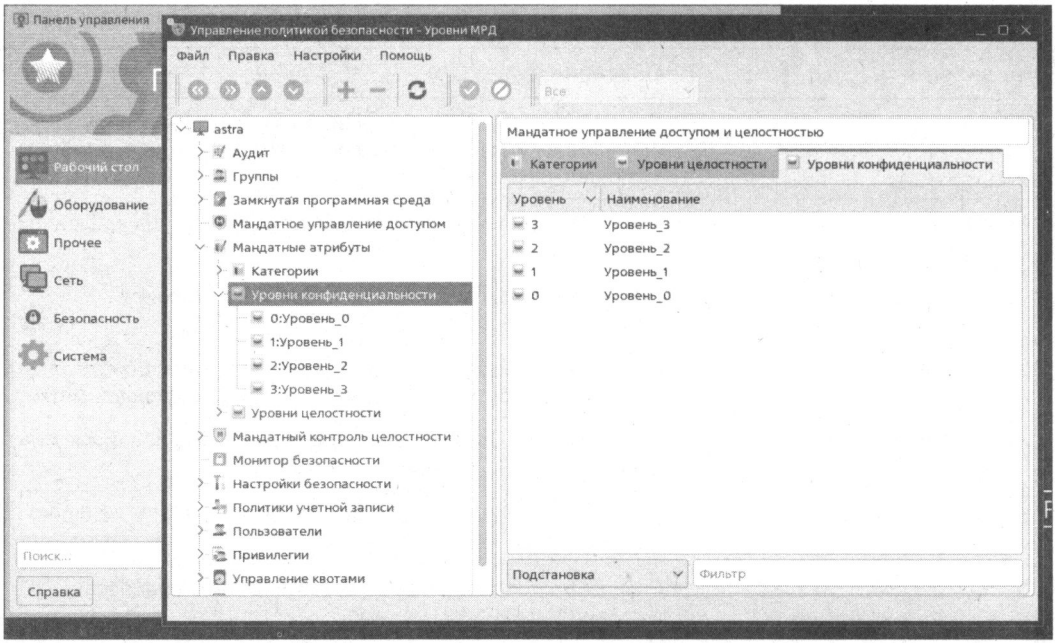

Рис. 27.2. Определение уровней конфиденциальности

Чтобы создать новый уровень, нажмите кнопку **+** на панели инструментов, выберите числовое значение уровня, введите название уровня и нажмите кнопку **Применить** .

27.3. Определение категорий конфиденциальности

Откройте конфигуратор **Управление политикой безопасности** и перейдите в раздел **Мандатные атрибуты | Категории**. Здесь вы можете создать различные категории конфиденциальности и переименовать уже существующие (рис. 27.3).

27.4. Назначение уровней и категорий конфиденциальности учетным записям пользователей

> ***Внимание!***
>
> Прежде, чем вы что-то успеете сломать, рекомендую создать отдельную учетную запись пользователя, над которым вы будете проводить эксперименты. Не изменяйте параметры МРД для пользователя-администратора, от имени которого вы производите настройку.

Находясь в конфигураторе **Управление политикой безопасности**, перейдите в раздел **Пользователи,** выберите учетную запись пользователя, параметры кото-

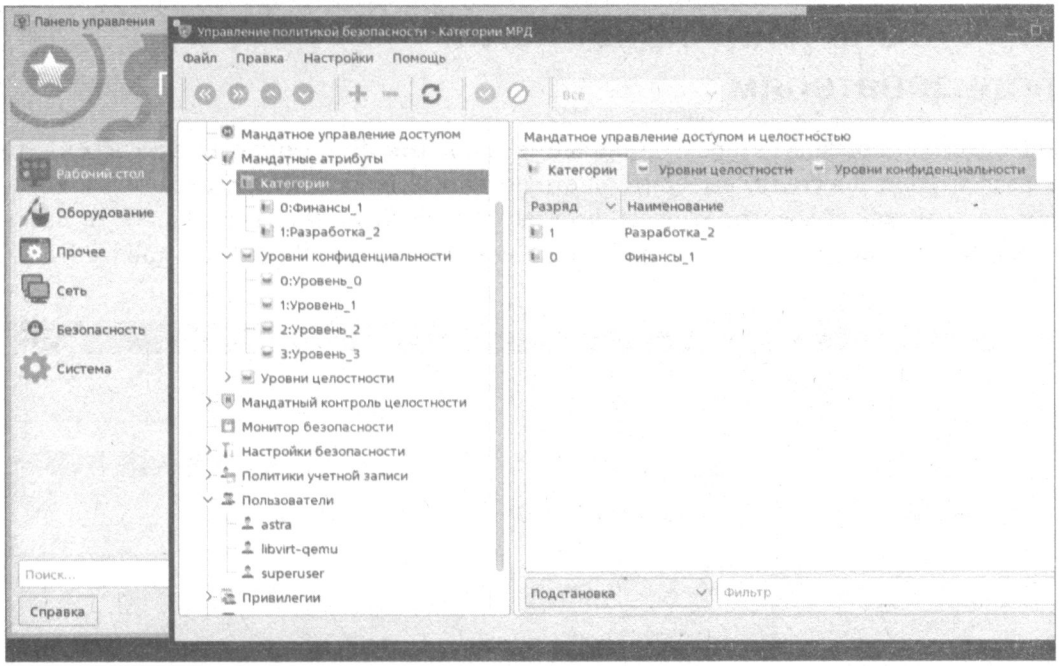

Рис. 27.3. Настройка категорий конфиденциальности

рого вам нужно изменить, и перейдите на вкладку **МРД**. Выберите минимальный и максимальный уровни безопасности для пользователя, установите нужный уровень целостности, а также выберите категории конфиденциальности, назначенные пользователю (рис. 27.4). Когда все будет готово, нажмите кнопку **Применить** .

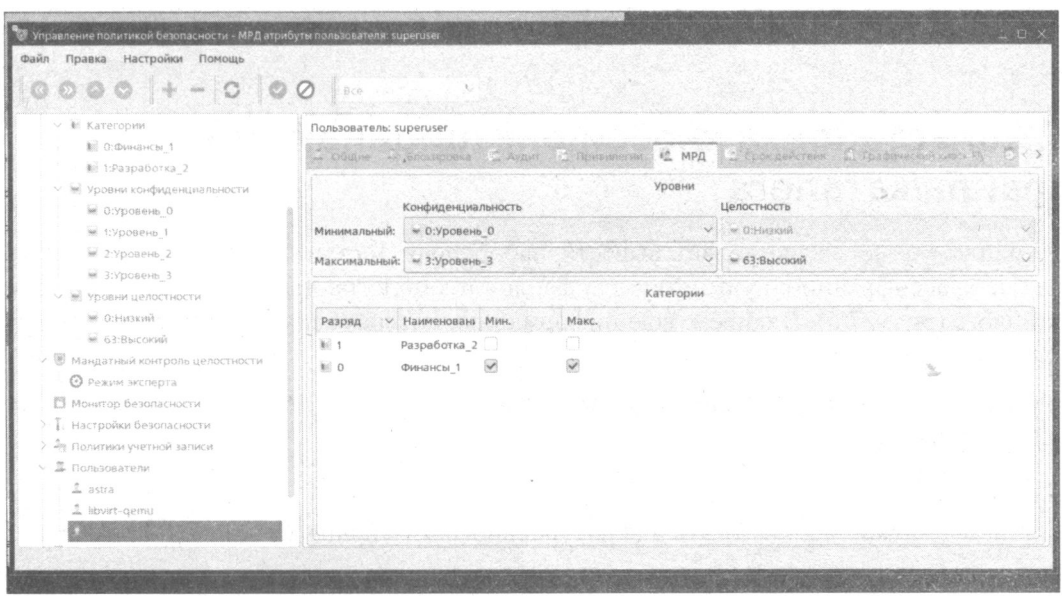

Рис. 27.4. Назначение уровней и категорий конфиденциальности пользователю

27.5. Назначение привилегий пользователям

Находясь в конфигураторе **Управление политикой безопасности**, перейдите в раздел **Пользователи,** выберите учетную запись пользователя, параметры которого вам нужно изменить, и перейдите на вкладку **Привилегии** (рис. 27.5). Здесь можно установить как привилегии Linux, так и привилегии PARSEC (см. табл. 27.1).

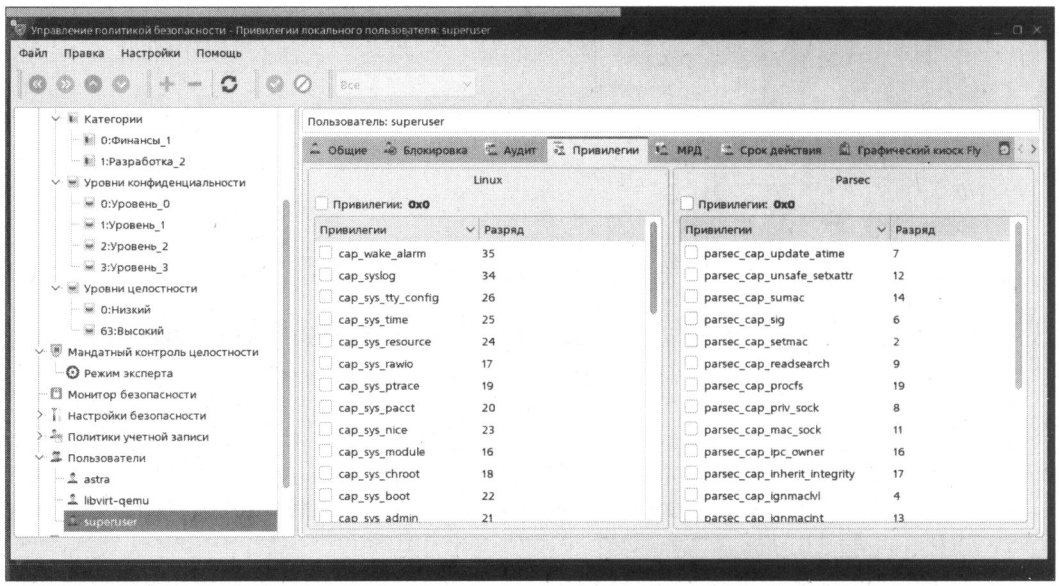

Рис. 27.5. Назначение привилегий пользователя

27.6. Установка классификационной метки файла/каталога

Команда `pdp-ls -M` позволяет вывести набор прав доступа, владельца, группу, а также классификационную метку для файлов и каталогов, находящихся в текущем каталоге (рис. 27.6). В общем, всю информацию, которая может повлиять на доступ к файлу.

Установить классификационную метку можно командой `pdpl-file`. Синтаксис ее следующий:

```
pdpl-file <метка> <файл/каталог>
```

Опцию `-R` можно использовать для рекурсивного изменения метки — например:

```
pdpl-file -R 1:0:0:0 reports
```

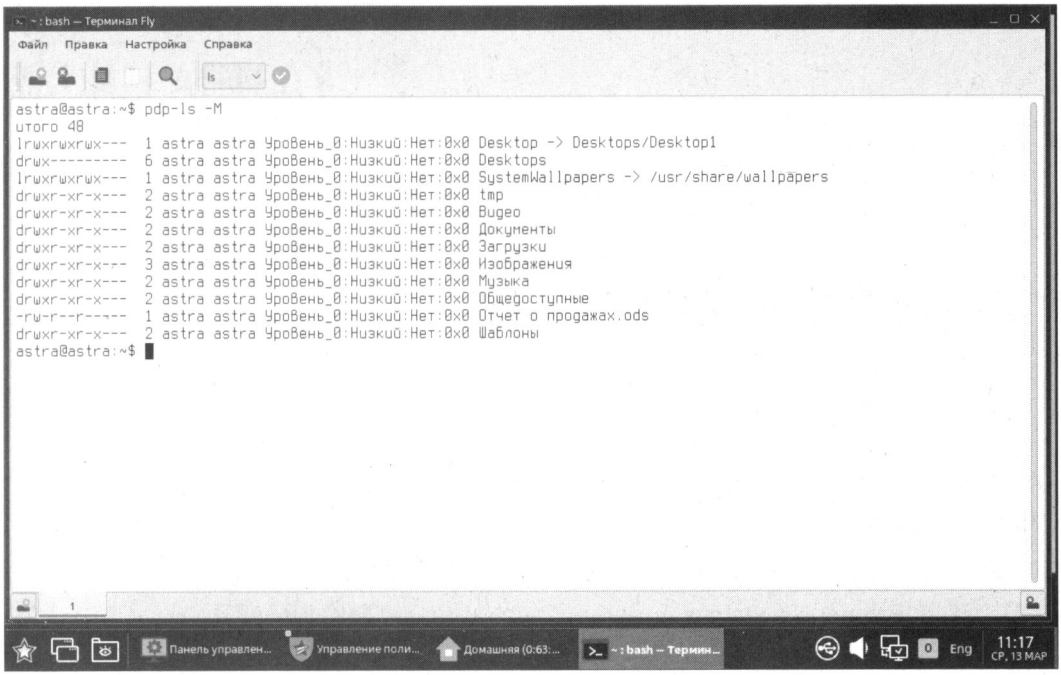

Рис. 27.6. Вывод команды `pdp-ls -M`

ГЛАВА 28

Нештатные ситуации

- Разбор нештатной ситуации
- Восстановление пароля root
- Резервное копирование при включенных МКЦ и МРД
- Восстановление резервных копий при включенных МКЦ и МРД
- Режим восстановления

28.1. Разбор нештатной ситуации

Начнем сразу с практического примера. На рис. 28.1 показано, что при загрузке системы возникла нештатная ситуация, а именно — отказался монтироваться один из накопителей.

Система сообщает, что перешла в аварийный режим (emergency mode), и предлагает ввести пароль root для входа и ручного исправления ошибок или же нажать комбинацию клавиш <Ctrl>+<D> для продолжения загрузки в обычном режиме (если отказался монтироваться несистемный раздел, то загрузка системы возможна, но данные на том разделе будут недоступны).

Аварийный режим и режим восстановления

Обнаружив неисправность в процессе загрузки, система перейдет в аварийный режим (emergency mode). Если же вы сами знаете, что неисправность имеется, то можете сразу загрузить режим восстановления (recovery mode), выбрав его при загрузке ОС в меню загрузчика. Для пользователя эти режимы ничем не отличаются. При входе в оба режима нужно указывать пароль root, после входа откроется командная строка с правами root, и ничего больше. Никаких графических средств восстановления запущено не будет, поэтому названия разные, но суть одна.

О пароле root

У нас с вами здесь есть следующая нестыковочка. Если вы не задавали пароль root, то вы его не знаете, следовательно, ввести его не можете. Будет хорошо, если вы, прочитав это примечание, установите пароль root командой `sudo passwd root`, и тогда вам не придется ломать голову, когда чрезвычайная ситуация произойдет. А вот если нет, тогда вам придется еще и восстанавливать пароль root (см. *разд. 28.2*).

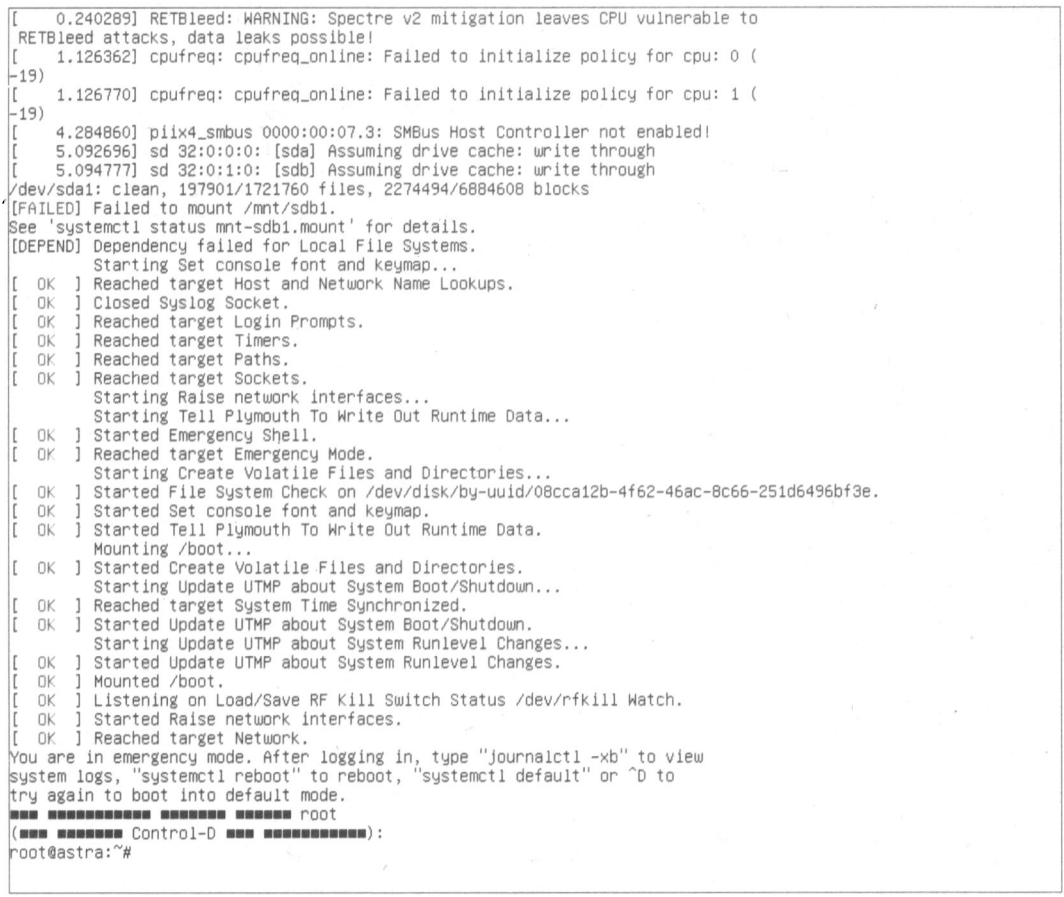

Рис. 28.1. Нештатная ситуация

Итак, войдя в аварийный режим, введите команду:

```
journalctl -xb
```

Эта команда покажет последние загрузочные сообщения, и среди них будет информация о возникшей ошибке. В зависимости от полученной информации надо будет предпринять какие-либо действия.

В нашем случае, поскольку ошибка возникла во время монтирования накопителя, поможет команда:

```
systemctl status mnt-sdb1.mount
```

Фрагмент `sdb1` в этой команде нужно заменить на имя устройства вашего накопителя.

Как показано на рис. 28.2, система пытается смонтировать несуществующую файловую систему ex4. Такой файловой системы не существует — очевидно, в файле **/etc/fstab** была допущена ошибка. Открываем его (рис. 28.3) и редактируем — вместо `ex4` нужно указать `ext4` и сохранить файл. Осталось только перезагрузить компьютер командой `reboot`.

```
See 'systemctl status mnt-sdb1.mount' for details.
[DEPEND] Dependency failed for Local File Systems.
         Starting Set console font and keymap...
[  OK  ] Reached target Host and Network Name Lookups.
[  OK  ] Closed Syslog Socket.
[  OK  ] Reached target Login Prompts.
[  OK  ] Reached target Timers.
[  OK  ] Reached target Paths.
[  OK  ] Reached target Sockets.
         Starting Raise network interfaces...
         Starting Tell Plymouth To Write Out Runtime Data...
[  OK  ] Started Emergency Shell.
[  OK  ] Reached target Emergency Mode.
         Starting Create Volatile Files and Directories...
[  OK  ] Started File System Check on /dev/disk/by-uuid/08cca12b-4f62-46ac-8c66-251d6496bf3e.
[  OK  ] Started Set console font and keymap.
[  OK  ] Started Tell Plymouth To Write Out Runtime Data.
         Mounting /boot...
[  OK  ] Started Create Volatile Files and Directories.
         Starting Update UTMP about System Boot/Shutdown...
[  OK  ] Reached target System Time Synchronized.
[  OK  ] Started Update UTMP about System Boot/Shutdown.
         Starting Update UTMP about System Runlevel Changes...
[  OK  ] Started Update UTMP about System Runlevel Changes.
[  OK  ] Mounted /boot.
[  OK  ] Listening on Load/Save RF Kill Switch Status /dev/rfkill Watch.
[  OK  ] Started Raise network interfaces.
[  OK  ] Reached target Network.
You are in emergency mode. After logging in, type "journalctl -xb" to view
system logs, "systemctl reboot" to reboot, "systemctl default" or ^D to
try again to boot into default mode.
■■■ ■■■■■■■■■■ ■■■■■■■ ■■■■■■ root
(■■■ ■■■■■■■ Control-D ■■■ ■■■■■■■■■■■):
root@astra:~# systemctl status mnt-sdb1.mount
● mnt-sdb1.mount - /mnt/sdb1
   Loaded: loaded (/etc/fstab; generated; vendor preset: enabled)
   Active: failed (Result: exit-code) since Mon 2023-10-23 11:06:16 MSK; 1min 40s ago
    Where: /mnt/sdb1
     What: /dev/sdb1
     Docs: man:fstab(5)
           man:systemd-fstab-generator(8)
  Process: 485 ExecMount=/bin/mount /dev/sdb1 /mnt/sdb1 -t ex4 (code=exited, status=32)

окт 23 11:06:16 astra systemd[1]: Mounting /mnt/sdb1...
окт 23 11:06:16 astra systemd[1]: mnt-sdb1.mount: Mount process exited, code=exited status=32
окт 23 11:06:16 astra systemd[1]: Failed to mount /mnt/sdb1.
окт 23 11:06:16 astra systemd[1]: mnt-sdb1.mount: Unit entered failed state.
root@astra:~# _
```

Рис. 28.2. Состояние монтирования устройства `sdb1`

```
/etc/fstab          [-M--] 33 L:[  1+14  15/ 16] *(834 / 859b) 0120 0x078                [*][X]
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sda1 during installation
UUID=8c39e0af-c808-48d1-84cf-02aa946d0984 /               ext4    errors=remount-ro 0       1
# /boot was on /dev/sda2 during installation
UUID=08cca12b-4f62-46ac-8c66-251d6496bf3e /boot           ext4    defaults        0       2
# swap was on /dev/sda3 during installation
UUID=ada8b09d-ea05-4258-aa27-6a22e4982209 none            swap    sw              0       0
/dev/sr0        /media/cdrom0   udf,iso9660 user,noauto     0       0
/dev/sdb1       /mnt/sdb1       ex4     defaults    0   0

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перем~тить 7Поиск 8Удалить 9МенюMC 10Выход
```

Рис. 28.3. Редактирование файла /etc/fstab

Примечание

Эта чрезвычайная ситуация была специально сымитирована. На практике вам предстоит бороться с реальными отказами.

28.2. Восстановление пароля root

Установить пароль root можно так:

1. Войти как пользователь-администратор.
2. Открыть терминал (или войти в консоль).
3. Ввести команду `sudo passwd root`.

Но бывают сложные ситуации, когда система полноценно не загружается, доступен только аварийный режим, а пароль от root вы не помните. Для выхода из такой ситуации:

1. Перезагрузите компьютер.
2. В меню загрузчика нажмите клавишу <e> — откроется экран редактирования конфигурации загрузочной метки.
3. Найдите в нем ключевое слово `linux` и в конце строки добавьте пробел и строку `init=/bin/bash`, как показано на рис. 28.4.

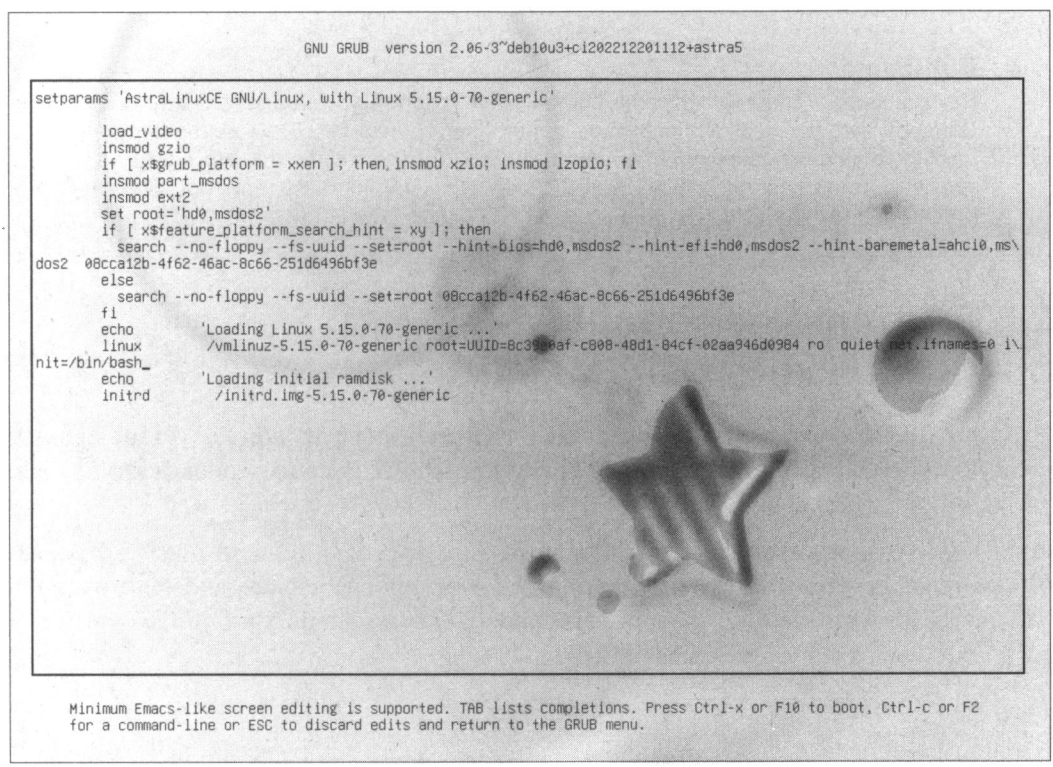

Рис. 28.4. Редактирование конфигурации загрузчика

4. Нажмите клавишу <F10> для загрузки системы с заданной конфигурацией — вы увидите консоль с правами root (рис. 28.5).
5. Введите команду `passwd root` для смены пароля. Аналогичным образом можно сменить пароль другого пользователя или добавить нового пользователя командой `adduser`.
6. Введите команду `reboot` для перезагрузки — теперь у вас есть ключи от системы, и вы можете делать с ней все, что вам захочется.

```
[    0.107985] RETBleed: WARNING: Spectre v2 mitigation leaves CPU vulnerable to
 RETBleed attacks, data leaks possible!
[    0.909073] cpufreq: cpufreq_online: Failed to initialize policy for cpu: 0 (
-19)
[    0.909383] cpufreq: cpufreq_online: Failed to initialize policy for cpu: 1 (
-19)
[    3.246779] piix4_smbus 0000:00:07.3: SMBus Host Controller not enabled!
[    3.932031] sd 32:0:0:0: [sda] Assuming drive cache: write through
[    3.932554] sd 32:0:1:0: [sdb] Assuming drive cache: write through
/dev/sda1: clean, 197918/1721760 files, 2275365/6884608 blocks
bash: cannot set terminal process group (-1): Inappropriate ioctl for device
bash: no job control in this shell
root@(none):/# _
```

Рис. 28.5. Консоль с правами root

Внимание!

Да, так можно взломать любую систему, на которой установлена Astra Linux. Минимальной защитой может послужить пароль загрузчика GRUB — вы не сможете отредактировать загрузочную метку, если будете знать этот пароль. Но никто не мешает злоумышленнику загрузиться с LiveCD/LiveUSB и сбросить пароль root. Поэтому единственная надежная защита данных — это их шифрование.

28.3. Резервное копирование при включенных МКЦ и МРД

Утилита tar позволяет архивировать файлы и каталоги с учетом установленных прав доступа. Но в Astra Linux Special Edition, кроме обычных прав доступа, есть также и мандатные атрибуты.

Если при создании архива не указать опции `--xattrs` и `--acls`, то он будет создан без учета мандатных атрибутов. Другими словами, после распаковки такого архива вам придется устанавливать атрибуты заново, что не очень удобно. Поэтому при создании архива всегда указываем опции `--xattrs` и `--acls`:

```
sudo tar --xattrs --acls -czf etc-bk.tar.gz /etc
```

Этой командой мы создаем резервную копию каталога /etc со всеми необходимыми атрибутами.

28.4. Восстановление резервных копий при включенных МКЦ и МРД

Создать резервную копию — это еще не все. Важно правильно распаковать архив. В предыдущем разделе мы создали резервную копию каталога /etc. Теперь попытаемся восстановить из нее файлы. Как вы уже догадались, это будет не просто. Для правильной распаковки архива, содержащего файлы с мандатными атрибутами, нужно ввести целых три команды:

```
# echo 1 | tee /parsecfs/unsecure_setxattr
# /usr/sbin/execaps -c 0x1000 -- tar --xattrs --xattrs-
include=security.{PDPL,AUDIT,DEF_AUDIT} --acls -xzf etc-bk.tar.gz -C /etc
# echo 0 | tee /parsecfs/unsecure_setxattr
```

Их нужно вводить с полномочиями root, поэтому лучше сначала ввести команду `sudo bash` (чтобы не вводить много раз команду `sudo`), а уже после этого вводить все эти команды.

Первая команда разрешает мандатную привилегию `PARSEC_CAP_UNSAFE_SETXATTR`, которая позволяет небезопасную установку атрибутов. Далее мы выполняем команду:

```
/usr/bin/execaps -c 0x1000
```

которая устанавливает для команды разархивирования:

```
tar --xattrs --xattrs-include=security.{PDPL,AUDIT,DEF_AUDIT} --acls -xzf
etc-bk.tar.gz -C /etc
```

мандатную привилегию `PARSEC_CAP_UNSAFE_SETXATTR`, которая разрешает устанавливать мандатные атрибуты объектов файловой системы без учета мандатных атрибутов родительского объекта-контейнера.

Наконец, третья команда возвращает все как было — она восстанавливает запрет на применение `PARSEC_CAP_UNSAFE_SETXATTR`.

Как видите, процедура сложная, но если делать все правильно, то проблем не будет.

28.5. Режим восстановления

В меню загрузчика GRUB есть метка восстановления `recovery mode`. Однако ее выбор не приводит к отображению средства восстановления в стиле Windows, как можно было бы подумать. Вам предложат ввести пароль администратора (пользователя, созданного при установке), а затем выполнить следующие команды (рис. 28.6):

- `journal -xb` — для просмотра загрузочных сообщений;
- `systemctl reboot` — для перезагрузки;
- `systemctl default` или `exit` — для загрузки в обычном режиме.

Рис. 28.6. Режим восстановления: выполните команды `journalctl -xb` — для просмотра логов, `systemctl reboot` — для перезагрузки и `systemctl default` или `exit` — для загрузки в обычном режиме

ASTRA LINUX®

ГЛАВА 29

Управление процессами

- Команды для мониторинга процессов
- Аварийное завершение процесса
- Изменение приоритета процесса
- Планирование запуска

29.1. Команды для мониторинга процессов

Каждому процессу в Linux присваивается уникальный номер — *идентификатор процесса* (PID, Process ID). Зная ID процесса, вы можете управлять процессом, а именно — завершить процесс или изменить его приоритет. Принудительное завершение процесса необходимо, если процесс завис и его нельзя завершить обычным образом. А изменение приоритета может понадобиться, если вы хотите, чтобы процесс доделал свою работу быстрее.

Чтобы узнать PID, используются команды `ps`, `top`, `htop`.

29.1.1. Команда *ps*

Откройте терминал, запустите `mc` и нажмите комбинацию клавиш <Ctrl>+<O> — чтобы скрыть панели. Затем введите команду `ps` для просмотра процессов на текущем псевдотерминале и посмотрите на ее вывод (рис. 29.1) — у нас есть процессы `bash` (оболочка) и сама команда `ps`. Но куда подевался процесс `mc`?

Давайте введем команду `ps -a`, позволяющую просмотреть список всех процессов пользователя на всех псевдотерминалах. Вывод команды `ps` можно отфильтровывать по нужному процессу, чтобы получить в выводе только его PID, — например, так:

```
ps -a | grep mc
```

Как показано на том же рис. 29.1, мы получили PID процесса `mc` (он равен 1788). Процесс перешел на другой псевдотерминал, поэтому мы его не увидели в первоначальном выводе.

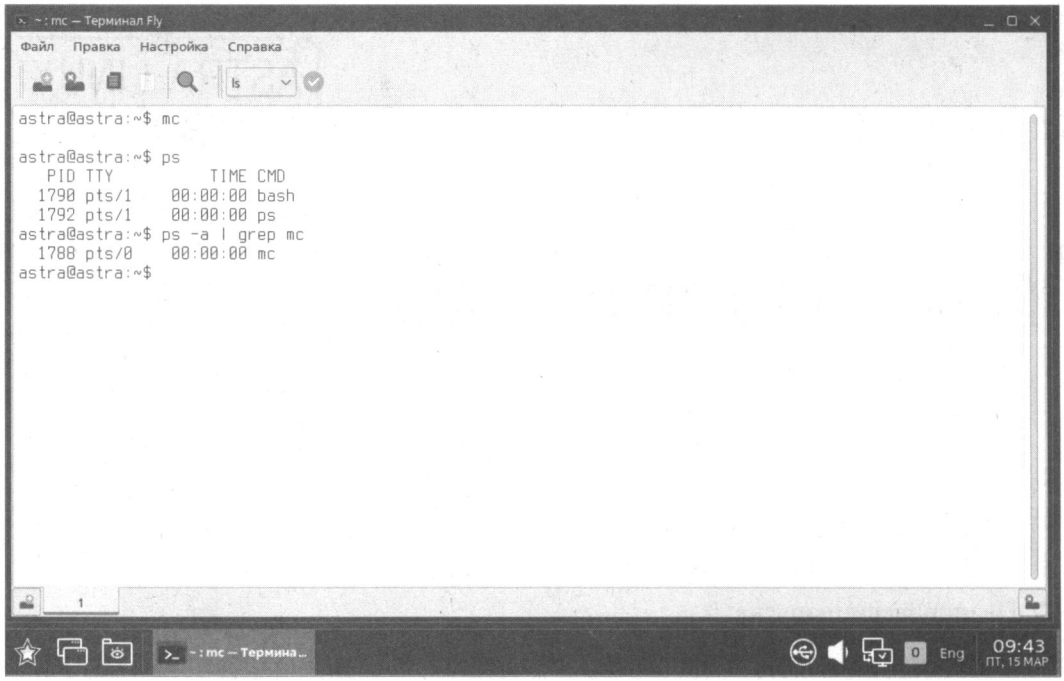

Рис. 29.1. Вывод команды `ps`

Псевдотерминал

В *главе 5* вкратце было рассказано, что такое *псевдотерминал*. Во времена мейнфреймов (70-е годы прошлого века) к суперкомпьютеру подключалось много терминалов пользователя, состоящих из монитора и клавиатуры. С появлением персональных компьютеров операционные системы UNIX, а потом и Linux перекочевали и на них, но проблема осталась в том, что у персонального компьютера (на то он и персональный) есть только одно рабочее место — одна клавиатура и один монитор. Однако UNIX от этого не стала однопользовательской — в ней появилась концепция *виртуальных терминалов*, переключаться между которыми можно было комбинациями клавиш <Alt>+<F*n*> (где *n* — номер функциональной клавиши) — например, <Alt>+<F2>). Когда появилась графическая система X Window, одним из первых графических приложений для этой системы стал графический терминал — в него вы можете вводить команды и видеть результат выполнения без необходимости возврата в консоль. Такие терминалы и стали называться *псевдотерминалами*.

29.1.2. Команда *top*

Иногда бывает, что система ужасно тормозит, — весь день работала нормально, а вдруг начала притормаживать.

Если вы даже не догадываетесь, из-за чего это случилось, вам нужно использовать команду `top` (рис. 29.2) — она выводит список процессов с сортировкой по процессорному времени. При этом на вершине списка будет находиться процесс, который занимает больше процессорного времени, чем сама система. Вероятно, из-за него и происходит эффект «торможения».

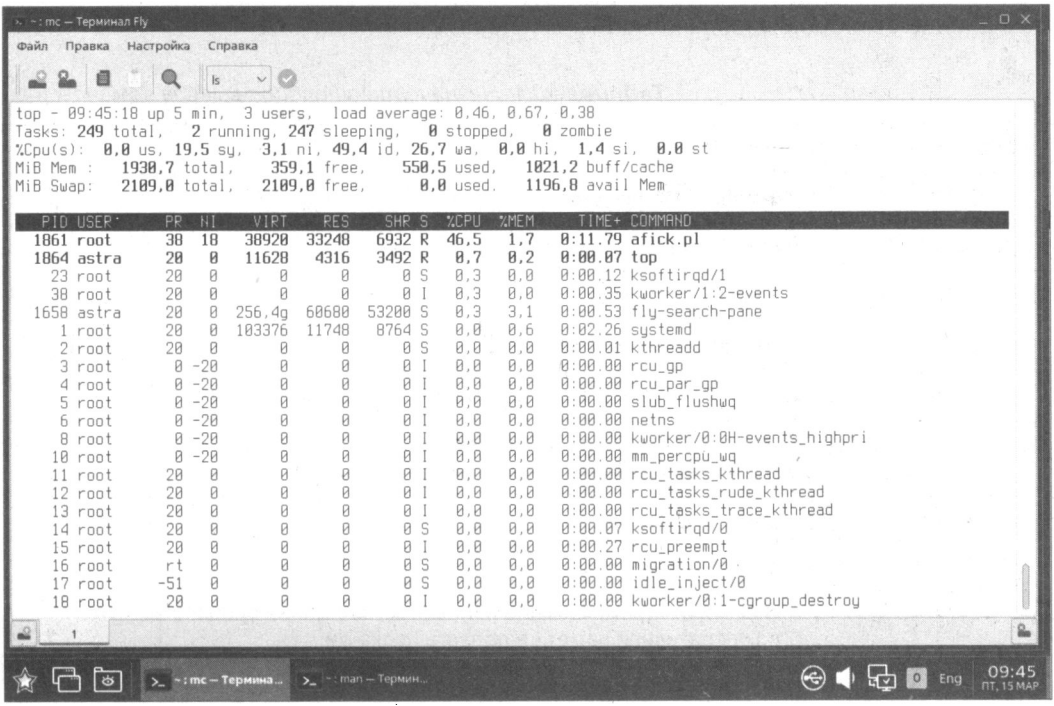

Рис. 29.2. Вывод программы `top`

На рис. 29.2 показано, что больше всего процессорного времени (46,5%) занимает программа afick.pl. Судя по названию — это Perl-сценарий. Вам нужно найти его, открыть и посмотреть, что он делает. Если то, что он делает, вам не нужно, надо понять, какая программа его вызывает, и разбираться уже с этой программой. Можно «убить» процесс (об этом позже), но, вероятнее всего, он будет перезапущен опять вызывающей его программой.

> ***Программа afick.pl***
>
> Если вам интересно, что делает именно программа afick.pl, могу пояснить, что она вычисляет контрольные суммы MD5-файлов, чтобы понять, какие из них изменились. Она является частью системы безопасности Astra Linux, в других дистрибутивах я ее не встречал.

Выйти из вывода команды `top` можно, нажав клавишу <Q>. Кроме нее действуют следующие клавиши:

- <U> — показывает только пользовательские процессы (т. е. те процессы, которые запустил пользователь, под именем которого вы работаете в системе);
- <D> — изменяет интервал обновления;
- <F> — изменяет столбец, по которому сортируются задачи. По умолчанию задачи сортируются по столбцу **%CPU**, т. е. по процессорному времени, занимаемому процессом;
- <H> — получить справку по остальным командам программы top.

Назначение столбцов вывода команды `top` показано в табл. 23.1.

***Таблица 23.1.** Назначение столбцов вывода команды top*

Столбец	Описание
PID	Идентификатор процесса
USER	Имя пользователя, запустившего процесс
PR	Приоритет процесса
NI	Показатель nice (см. *разд. 29.3*)
VIRT	Виртуальная память, использованная процессом (в Кбайт)
RES	Размер процесса, не перемещенный в область подкачки (в Кбайт). Этот размер равен размерам сегментов кода и данных, т. е. `RES = CODE + DATA`
S	Состояние процесса: • R — выполняется; • S — «спит» (режим ожидания), в этом состоянии процесс выгружен из оперативной памяти в область подкачки; • D — «непрерываемый сон» (uninterruptible sleep), из такого состояния процесс может вывести только прямой сигнал от оборудования; • T — процесс в состоянии трассировки или остановлен; • Z (зомби) — специальное состояние процесса, когда сам процесс уже завершен, но его структура еще осталась в памяти
%CPU	Занимаемое процессом процессорное время
%MEM	Использование памяти процессом
TIME+	Процессорное время, израсходованное с момента запуска процесса
COMMAND	Команда, которая использовалась для запуска процесса (обычно имя исполнимого файла процесса)

29.1.3. Команда *htop*

Команда `htop` по умолчанию в системе не устанавливается. Для ее установки нужно ввести команду:

```
sudo apt install htop
```

Эта команда является аналогом `top`, но визуально более привлекательна, — в частности, она выводит диаграммы загрузки каждого ядра и оперативной памяти (рис. 29.3).

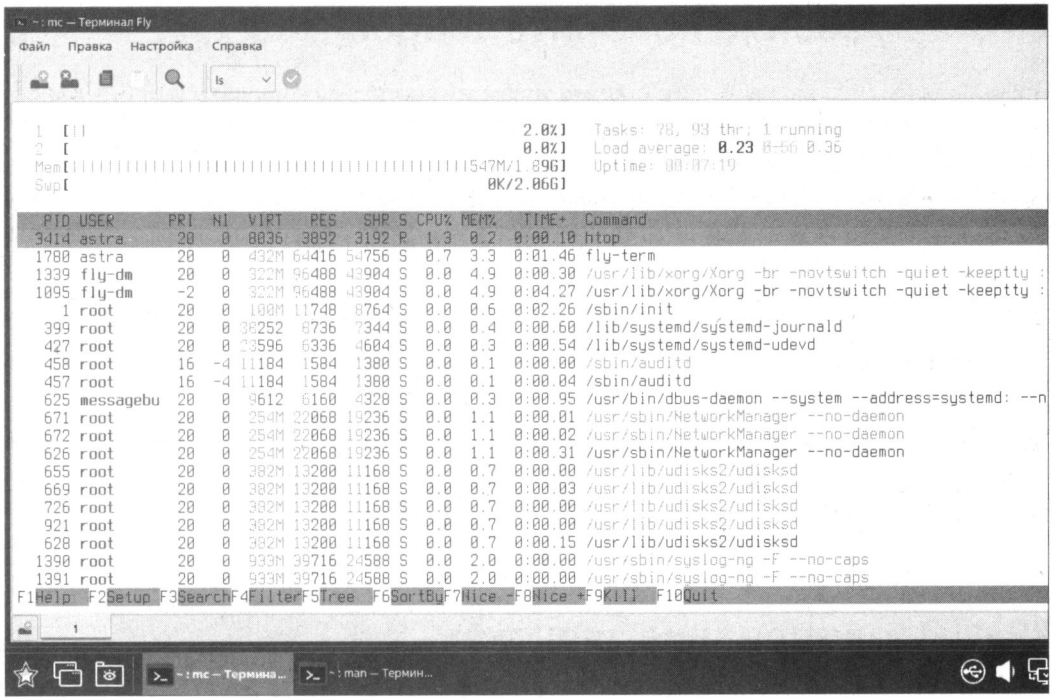

Рис. 29.3. Вывод команды `htop`

29.2. Аварийное завершение процесса

Теперь, когда мы знаем PID нашего процесса, мы можем его «убить»:

```
# kill 1788
```

Если выполнить теперь команду `ps -a`, то в списке процессов mc уже не будет.

Проще всего вычислить PID процесса с помощью следующей команды:

```
# ps -ax | grep <имя>
```

Например:

```
# ps -ax | grep firefox.
```

Следует признать, что все эти действия, связанные с вычислением PID процесса, мы рассмотрели только для того, чтобы познакомиться с командой `ps`.

Так что, если вы знаете только имя процесса, гораздо удобнее использовать команду:

```
# killall <имя процесса>
```

Но имейте в виду, что такая команда завершит все экземпляры этого процесса. А вполне может быть, что у вас на одной консоли находится mc, который нужно «убить», а на другой — нормально работающий mc. Команда `killall` «убьет» оба процесса.

29.3. Изменение приоритета процесса

Предположим, что вы работаете с видео и вам нужно перекодировать файл из одного видеоформата в другой. Конвертирование видео занимает много процессорного времени, а хотелось бы все сделать как можно быстрее и уйти пораньше домой. Тогда вам поможет команда nice — она позволяет запустить любую программу с указанным приоритетом. Ясно — чем выше приоритет, тем быстрее будет выполняться программа. Формат вызова команды следующий:

```
nice -n <приоритет> команда аргументы
```

Максимальный приоритет задается числом `-20`, а минимальный — числом `19`. Приоритет по умолчанию равен 10.

Если процесс уже запущен, тогда для изменения его приоритета можно использовать команду `renice`:

```
renice -n <приоритет> -p PID
```

29.4. Планирование запуска

Существуют различные способы планирования запуска:

- планировщик cron — позволяет запланировать периодический запуск нужной вам программы;
- планировщик at — планирует разовый запуск программы в нужное время;
- таймеры systemd — система инициализации позволяет не только автоматически загружать сервисы, но и планировать автоматический запуск заданий.

Первый способ — классический, им наиболее часто пользуются системные администраторы, и он стал стандартом де-факто. Второй способ используется не очень часто, поэтому пакет at даже не устанавливается по умолчанию. Для его установки нужно ввести команду:

```
sudo apt install at
```

Что же касается таймеров systemd, многие администраторы даже не знают, что такая возможность есть. Используя ее для настройки автоматического запуска программы, вам придется создать юнит systemd, что гораздо сложнее, чем просто добавить строчку в файл расписания, — как в случае с cron. Единственное преимущество таймеров — возможность запуска программы после определенного события, чего явно не умеет cron. Если вам нужна именно такая функциональность, обратитесь к документации по systemd[1], а мы далее рассмотрим классический способ автоматического запуска программ.

[1] См. **https://wiki.archlinux.org/title/Systemd/Timers**.

29.4.1. Необходимость выполнения команд по расписанию

Очень часто приходится периодически выполнять одни и те же действия. Например, каждый день проверять обновление антивируса (или раз в неделю — в зависимости от того, как часто выходят для него обновления) или каждые 30 минут — обновлять цены и остатки вашего интернет-магазина. Можно выполнять эти действия самому, но это вариант не лучший. Представьте, что ваш рабочий день будет начинаться с команды запуска программы обновления антивируса, а каждые 30 минут вам придется запускать обновление цен и остатков. Во-первых, это не очень удобно, а во-вторых, можно легко забыть выполнить ту или иную команду. Например, в пятницу вечером вы можете забыть выполнить команду создания резервной копии, а в понедельник утром что-то случится с сервером, и вы не досчитаетесь всего пользовательского каталога. Не очень приятно, правда?

29.4.2. Редактирование основной таблицы расписания планировщика

В Astra Linux используется планировщик anacron. Во многом он похож на стандартный cron, который присутствует в других дистрибутивах, но все же есть и небольшие различия.

Главное преимущество anacron заключается в том, что он, в отличие от cron, учитывает время, когда компьютер был выключен. Планировщик cron родом из UNIX, а эта операционная система устанавливалась только на серверах, которые всегда включены. Предположим, что вам нужно каждый понедельник в 7 часов утра рассылать некоторую информацию своим сотрудникам. Вы настроили cron так, чтобы он запускал сценарий отправки сообщений каждый понедельник в 7 утра. Но вот беда — в 6 часов утра выключили электричество, а включили его, скажем, в 7:20. Но 7:20 — это не 7:00, следовательно, cron не выполнит задание по отправке сообщений, а ваши сотрудники не получат важную информацию.

Anacron работает не так. Если он обнаружит, что некоторые задания не выполнены по тем или иным причинам (выключение электричества, перезагрузка компьютера), он обязательно их выполнит. Поэтому ваши сотрудники получат информацию, но с небольшой задержкой. Все же лучше, чем получить важную информацию лишь в следующий понедельник.

Но и у anacron есть свои недостатки. В частности, пользователи не могут создавать свои собственные расписания (только пользователь root может создать расписание для пользователя), и файл /etc/crontab может редактировать только root.

Формат таблицы расписания в последних версиях anacron такой же, как и в случае с cron:

```
минуты (0-59) часы (0-23) день (1-31) месяц (1-12) день_недели (0-6, 0 — Вс)
команда
```

Содержимое файла /etc/crontab по умолчанию представлено на рис. 29.4. Обратите внимание, что сейчас редактируется системная (system-wide) таблица планировщика.

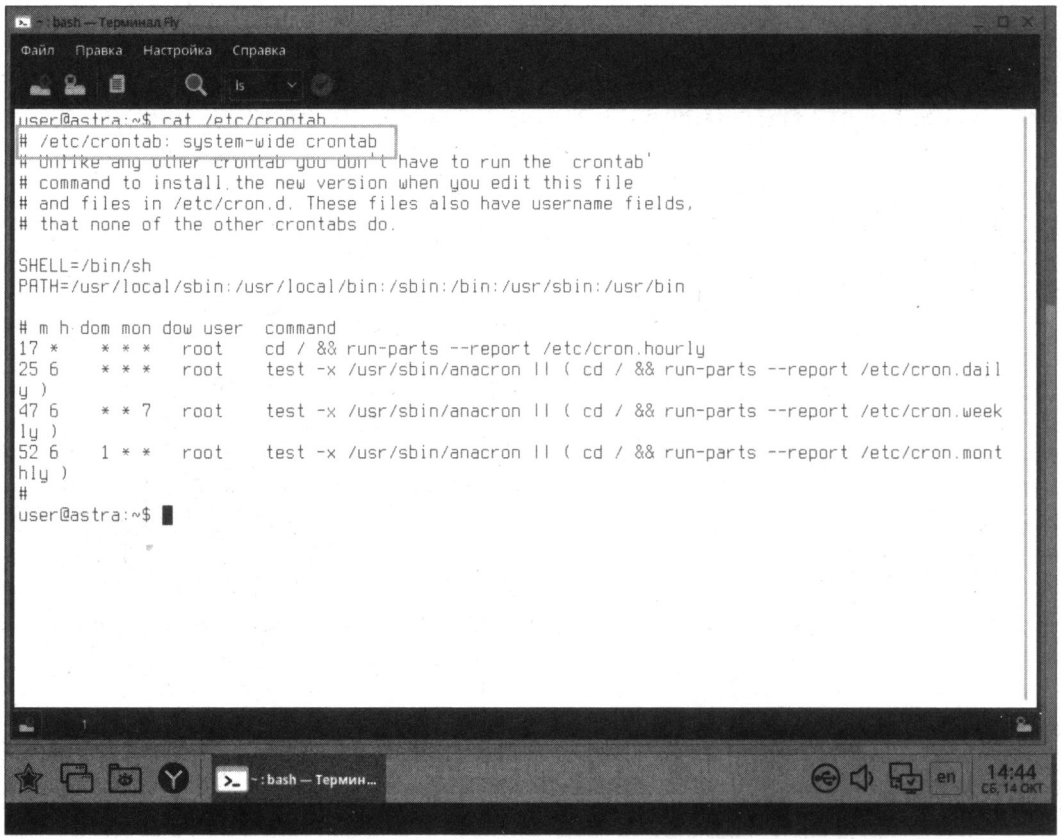

Рис. 29.4. Редактирование файла /etc/crontab

Посмотрим, что внутри этого файла. Каждый час в 17 минут (обратите внимание — не в 00!) запускается скрипт /etc/cron.hourly. Если вы добавите в него свою команду, то она будет выполнена каждый час в 17 минут. Аналогично со следующими скриптами:

- cron.daily — выполняется каждый день в 6:25;
- cron.weekly — каждую неделю в понедельник (7 — это понедельник) в 6:47;
- cron.montly — каждый месяц 1-го числа в 6:52.

> ***Разница в минутах...***
>
> По умолчанию таблица настроена так, что есть небольшая разница в минутах выполнения. Можно было бы настроить ее так, чтобы все эти команды выполнялись в 00 минут, но это бы создало лишнюю и неоправданную нагрузку на сервер.

29.4.3. Редактирование пользовательской таблицы расписания

У каждого пользователя может быть собственная таблица планировщика, но редактировать ее, как уже было отмечено ранее, может только пользователь root. Редактирование собственной таблицы пользователя может пригодиться, если вы хотите, чтобы выполняемые команды запускались с правами конкретного пользователя, а не с правами root, если вызывать их из системной таблицы планировщика (рис. 29.5).

Для редактирования таблицы конкретного пользователя используется команда:

```
sudo crontab -e -u <имя>
```

Пользователь, который вводит эту команду, должен быть администратором, т. е. у него должно быть право использовать команду `sudo`.

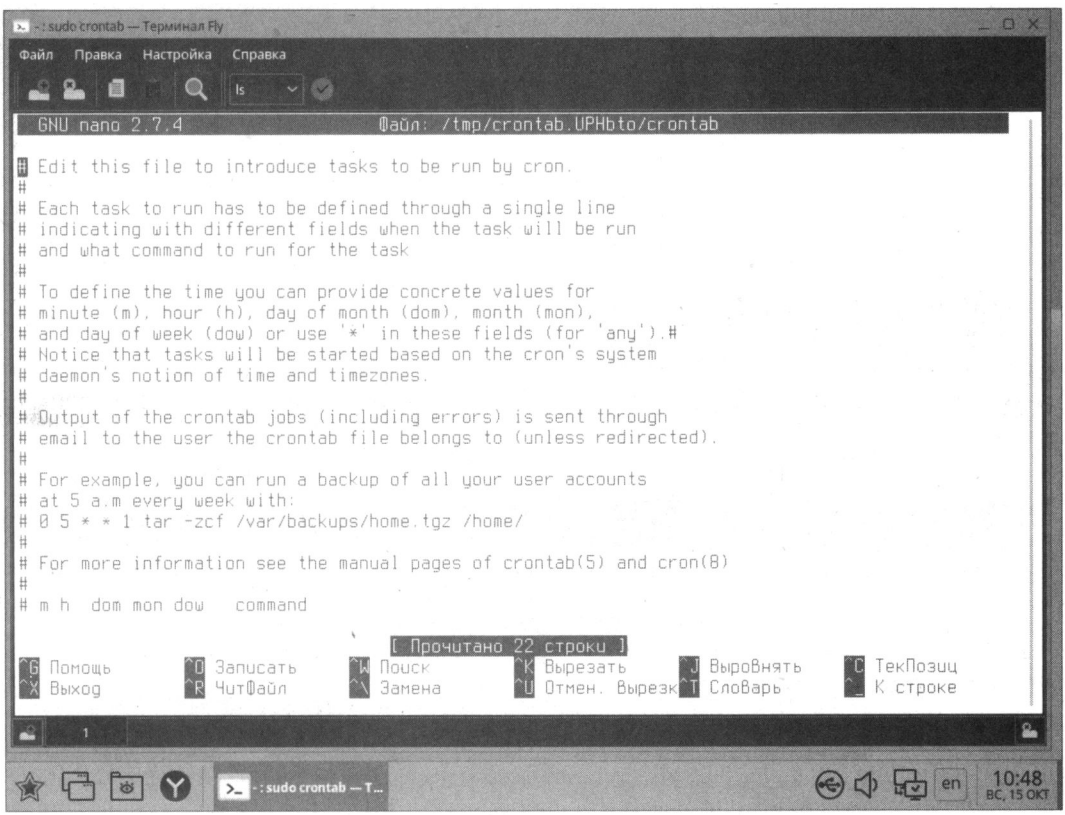

Рис. 29.5. Редактирование пользовательской таблицы расписания

29.4.4. Практическое упражнение: добавление собственной записи в таблицу расписания

Выполним небольшое практическое упражнение — создадим собственный скрипт и добавим его в системное расписание и в расписание пользователя.

Выполните следующие команды (рис. 29.6):

1. Создайте файл /tmp/dt.sh:

```
touch /tmp/dt.sh
```

Рис. 29.6. Процесс создания тестового скрипта

2. Разрешите запуск этого файла:

```
chmod +x /tmp/dt.sh
```

3. Добавьте в него команду, которая выводит текущую дату в файл cur.date, находящийся в домашнем каталоге пользователя:

```
echo "date >> ~/cur.date" > /tmp/dt.sh
```

4. Просмотрите содержимое этого файла, чтобы убедиться, что там все правильно:

```
cat /tmp/dt.sh
```

5. Запустите сценарий:

```
/tmp/dt/sh
```

6. Просмотрите файл cur.date, чтобы убедиться, что он выполняется:

```
cat ~/cur.date
```

Теперь в любом текстовом редакторе (можно в редакторе nano) откройте файл /etc/crontab:

```
sudo nano /etc/crontab
```

и добавьте в него строку (рис. 29.7):

```
*/1 * * * * /tmp/dt.sh
```

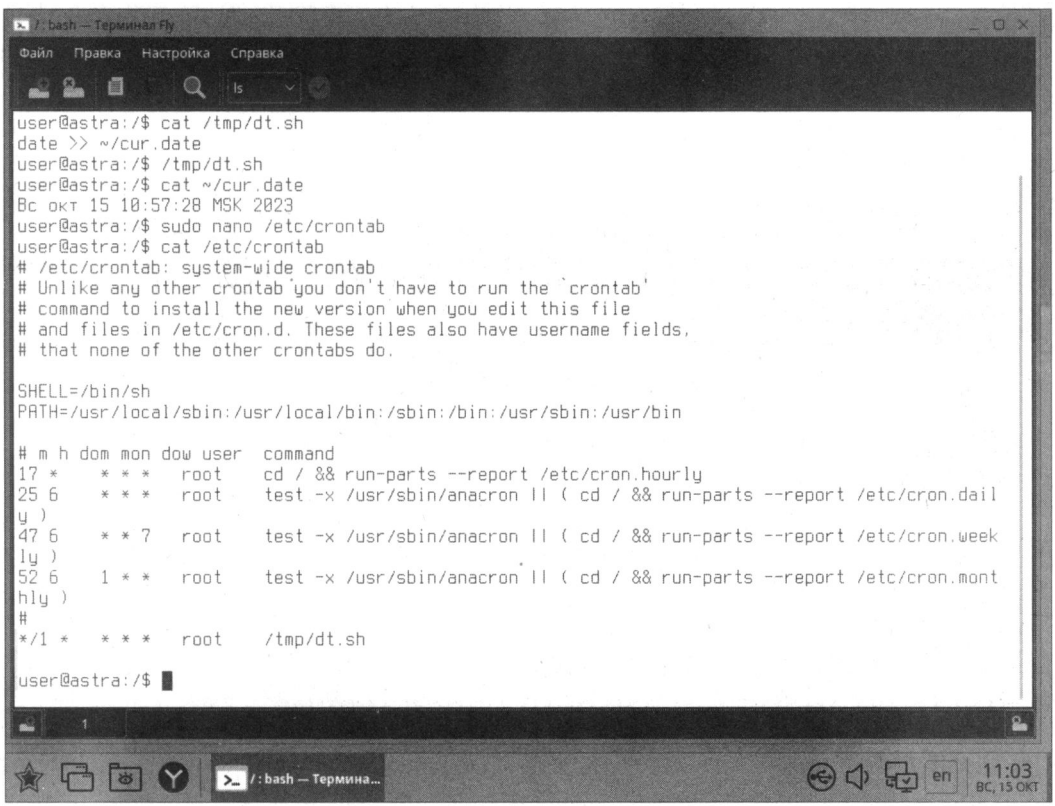

Рис. 29.7. Добавление команды в системную таблицу расписания

Этим мы обеспечили запуск нашего скрипта каждую минуту: `*/1`. Поскольку сценарий будет запускаться от имени root, файл cur.date нужно искать в каталоге /root. Откройте его и просмотрите содержимое. Как можно видеть на рис. 29.8, наш скрипт действительно запускается раз в минуту: 11:04, 11:05, 11:06.

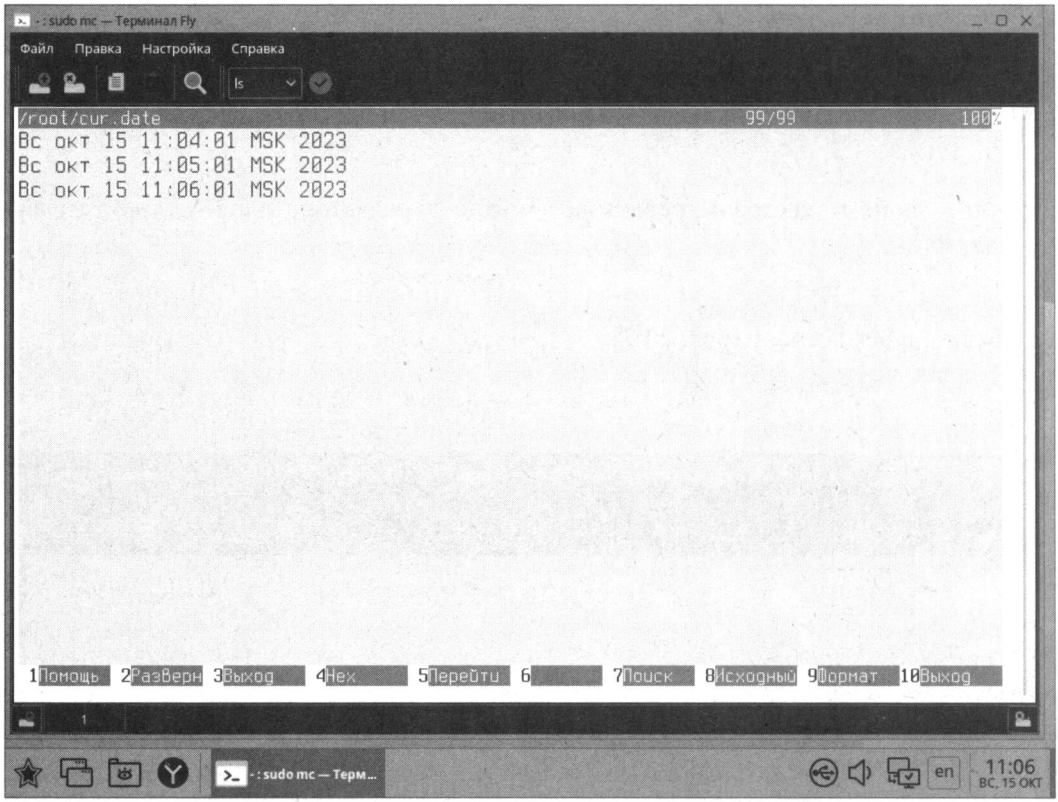

Рис. 29.8. Проверка работы сценария

САМОСТОЯТЕЛЬНОЕ УПРАЖНЕНИЕ

Аналогичным образом добавьте команду вызова файла */tmp/dt.sh* в таблицу расписания вашего пользователя (под которым вы обычно работаете). Вместо имени root не забудьте указать имя этого пользователя. Просмотрите файл *cur.date* в домашнем каталоге пользователя, чтобы убедиться, что скрипт выполняется.

29.4.5. Использование панели управления для редактирования таблицы расписания

При наличии навыков редактирования таблицы расписания из командной строки не составит труда разобраться с графическим интерфейсом для планировщика, который предоставляет Панель управления Astra Linux. Выполните следующие действия:

1. Запустите Панель управления и перейдите в раздел **Система**.
2. Запустите модуль **Планировщик задач** (рис. 29.9).
3. Используйте переключатель **Показать для**, чтобы просмотреть таблицу расписания конкретного пользователя, либо верните его в положение **система** для просмотра системной таблицы расписания.

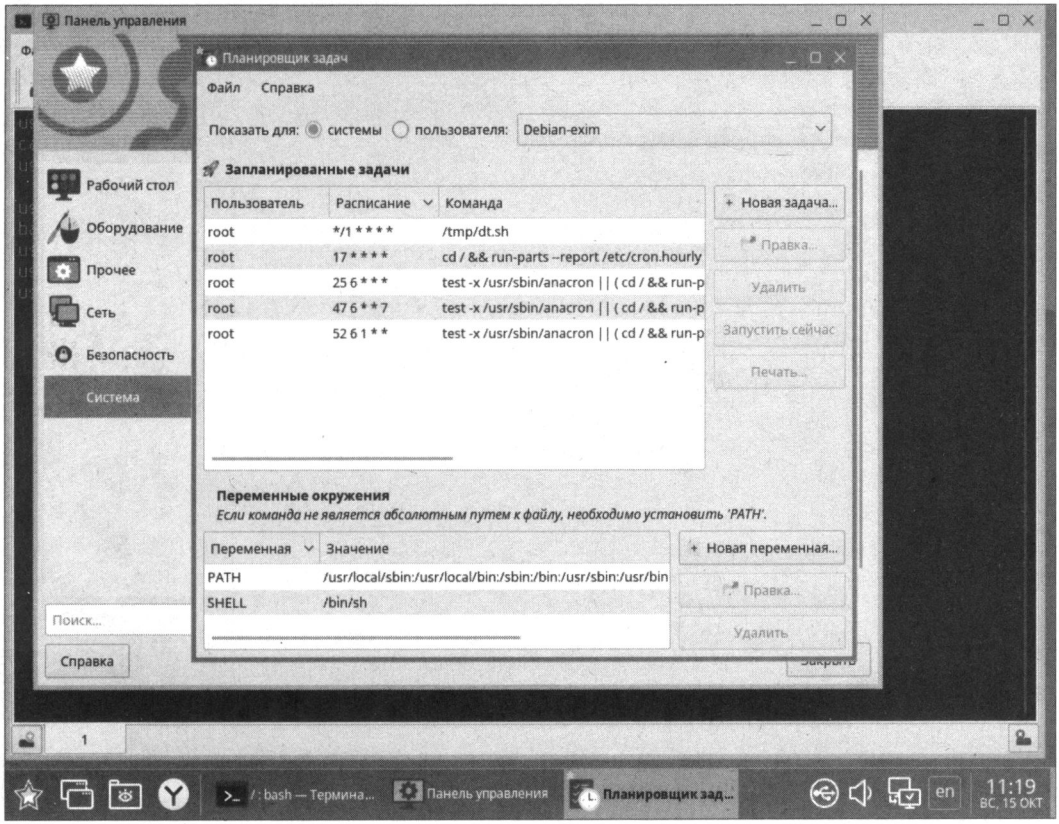

Рис. 29.9. Планировщик задач

4. Кнопка **Удалить** позволяет удалить добавленную задачу. Если вы еще не удалили задачу dt.sh из таблицы, выделите эту задачу и нажмите кнопку **Удалить**, а затем — кнопку **Да** для подтверждения удаления.

5. Кнопка **Новая задача** позволяет добавить в таблицу расписания новую задачу (рис. 29.10):

   а) Введите или выберите команду (поле **Команда**).

   б) Используйте список **Запустить как** для выбора пользователя, от имени которого запускается задача.

   в) Включите флажок **Графическое приложение**, если запускаемая команда является графическим приложением.

   г) Если нужно запускать команду при запуске системы, включите флажок **Запускать при загрузке системы**.

   д) Если не важно, когда именно запускать команду, включите флажок Запускать каждый день и нажмите кнопку **Да**. Если же важно конкретное время, установите его с помощью областей **Месяцы, Дни месяца, Дни недели, Часы, Минуты** и нажмите кнопку **Да**.

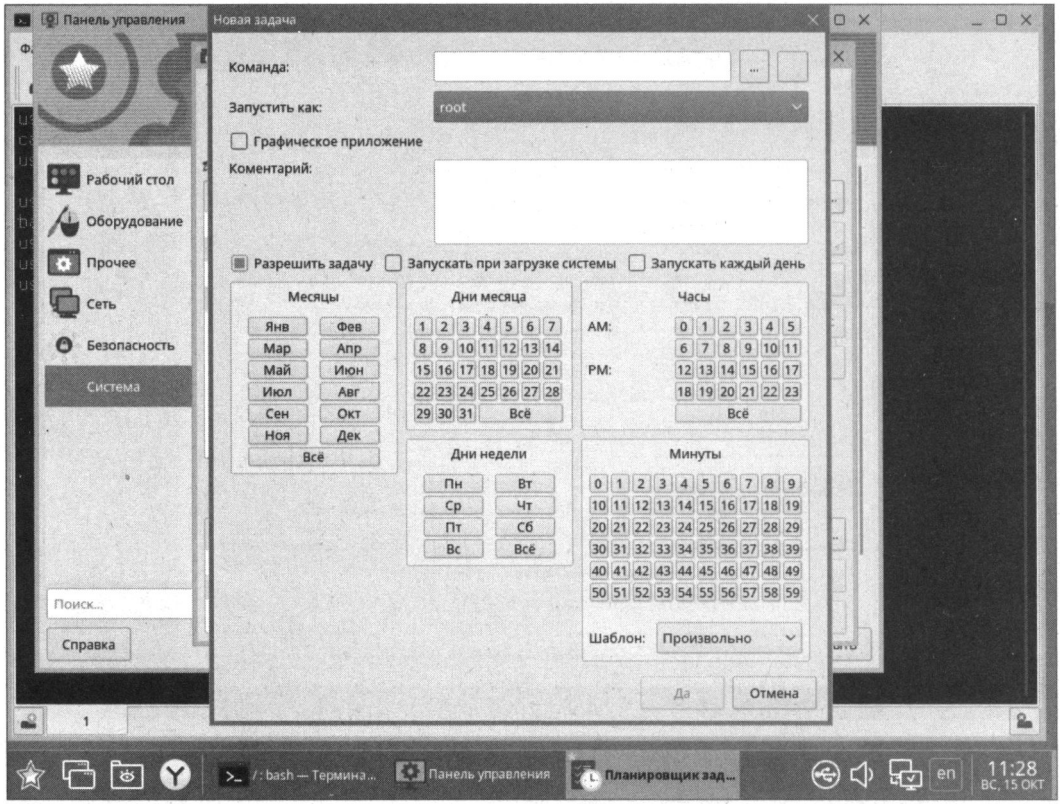

Рис. 29.10. Новая задача

ASTRA LINUX®

ГЛАВА 30

Автоматизация задач

- Настройка оболочки
- Пример простейшей автоматизации
- Привет, мир!
- Использование переменных в собственных сценариях
- Передача параметров сценарию
- Массивы и bash
- Циклы
- Условные операторы
- Функции
- Примеры сценариев

30.1. Настройка оболочки

bash — это самая популярная командная оболочка (командный интерпретатор), использующаяся практически во всех дистрибутивах Linux по умолчанию. Основное предназначение bash — выполнение команд, введенных пользователем. Пользователь вводит команду, bash ищет программу, соответствующую команде, в каталогах, указанных в переменной окружения PATH. Если такая программа найдена, то bash запускает ее и передает ей введенные пользователем параметры. В противном случае выводится сообщение о невозможности выполнения команды.

Файл /etc/profile содержит глобальные настройки bash, он влияет на всю систему — на каждую запущенную оболочку. Обычно /etc/profile не нуждается в изменении, а при необходимости изменить параметры bash редактируют один из следующих файлов:

- ~/.bash_profile — обрабатывается при каждом входе в систему;
- ~/.bashrc — обрабатывается при каждом запуске дочерней оболочки;
- ~/.bash_logout — обрабатывается при выходе из системы.

Файл ~/.bash_profile часто не существует, а если и существует, то в нем есть всего одна строка:

```
source ~/.bashrc
```

Эта строка означает, что нужно прочитать файл .bashrc. Поэтому будем считать основным конфигурационным файлом `bash` именно файл .bashrc. Но помните, что он влияет на оболочку текущего пользователя (такой файл находится в домашнем каталоге каждого пользователя — не забываем, что символ «~» означает домашний каталог). Если же вдруг понадобится задать параметры, которые повлияют на всех пользователей, то нужно редактировать файл /etc/profile.

В файле .bash_history (тоже находится в домашнем каталоге) хранится история команд, введенных пользователем. Так что вы можете просмотреть свои же команды, которые накануне вводили.

Какие настройки могут быть в .bashrc? Как правило, в этом файле задаются псевдонимы команд, определяется внешний вид приглашения командной строки, задаются значения переменных окружения.

Псевдонимы команд задаются с помощью команды `alias`, вот несколько примеров:

```
alias h='fc -l'
alias ll='ls -laFo'
alias l='ls -l'
alias g='egrep -i'
```

Псевдонимы работают просто: при вводе команды `l` на самом деле будет выполнена команда `ls -l`.

Теперь рассмотрим пример изменения приглашения командной строки. Глобальная переменная `PS1` отвечает за внешний вид командной строки. По умолчанию командная строка имеет формат:

пользователь@компьютер:рабочий_каталог

Значение `PS1` при этом будет:

```
PS1='\u@\h:\w$'
```

В табл. 30.1 приведены допустимые модификаторы командной строки.

***Таблица 30.1.** Модификаторы командной строки*

Модификатор	Описание
`\a`	ASCII-символ звонка (код 07). Не рекомендуется его использовать — очень скоро начнет раздражать
`\d`	Дата в формате «день недели, месяц, число»
`\h`	Имя компьютера до первой точки
`\H`	Полное имя компьютера
`\j`	Количество задач, запущенных в оболочке в настоящее время

Таблица 30.1 (окончание)

Модификатор	Описание
\l	Название терминала
\n	Символ новой строки
\r	Возврат каретки
\s	Название оболочки
\t	Время в 24-часовом формате (ЧЧ: ММ: СС)
\T	Время в 12-часовом формате (ЧЧ: ММ: СС)
\@	Время в 12-часовом формате (AM/PM)
\u	Имя пользователя
\v	Версия bash (сокращенный вариант)
\V	Версия bash (полная версия: номер релиза, номер патча)
\w	Текущий каталог (полный путь)
\W	Текущий каталог (только название каталога, без пути)
\!	Номер команды в истории
\#	Системный номер команды
\$	Если UID пользователя равен 0, будет выведен символ #, иначе — символ $
\\	Обратный слеш
$ ( )	Подстановка внешней команды

Вот пример задания альтернативного приглашения командной строки:

```
PS1='[\u@\h] $(date +%m/%d/%y) \$'
```

Вид приглашения будет такой:

```
[user@host] 27/10/25 $
```

В квадратных скобках выводится имя пользователя и имя компьютера, затем используется конструкция `$()` для подстановки даты в нужном нам формате и символ приглашения, который изменяется в зависимости от идентификатора пользователя.

Установить переменную окружения можно с помощью команды `export`, что будет показано позже.

30.2. Пример простейшей автоматизации

Представим, что нам нужно выполнить резервное копирование всех важных файлов, для чего создать архивы каталогов /etc, /home и /usr. Понятно, что понадобятся три команды вида:

```
tar -cvjf имя_архива.tar.bz2 каталог
```

Затем нам нужно записать все эти три файла на DVD или на внешний носитель.

Если выполнять такую операцию раз в месяц (или хотя бы раз в неделю), то ничего страшного. Но представьте, что вам нужно делать это каждый день или даже несколько раз в день? Думаю, такая рутинная работа вам быстро надоест. А ведь можно написать *сценарий*, который сам будет создавать резервные копии и записывать их на DVD или внешний диск! Все, что вам нужно, — это вставить чистый DVD перед запуском сценария. А если у вас внешний диск, который используется исключительно для бэкапов и постоянно подключен к компьютеру, делать вообще ничего не придется.

Можно пойти и иным путем: написать сценарий, который будет делать резервные копии системных каталогов и записывать их на другой раздел жесткого диска. Ведь не секрет, что резервные копии делаются не только на случай сбоя системы, но и для защиты от некорректного изменения данных пользователем. Помню, я как-то удалил важную тему форума и попросил своего хостинг-провайдера сделать откат. И был приятно удивлен, когда мне предоставили на выбор три резервные копии, — осталось лишь выбрать наиболее подходящую. Не думаете же вы, что администраторы провайдера только и занимались тем, что три раза в день копировали домашние каталоги пользователей? Поэтому автоматизация — штука полезная, и любому администратору нужно знать, как автоматизировать свою рутинную работу.

30.3. Привет, мир!

По традиции напишем первый сценарий, выводящий всем известную фразу «Привет, мир!» (листинг 15.1). Для редактирования сценариев вы можете использовать любимый текстовый редактор — например nano или ee.

Листинг 15.1. Первый сценарий

```
#!/bin/bash
echo "Привет, мир!"
```

Первая строка нашего сценария — это указание, что он должен быть обработан программой **/bin/bash**. Обратите внимание: если между `#` и `!` окажется пробел, то эта директива не сработает, поскольку будет воспринята как обычный комментарий. Комментарии начинаются, как вы уже догадались, с решетки:

```
# Комментарий
```

Вторая строка — это оператор `echo`, выводящий нашу строку. Сохраните сценарий под именем `hello` и введите команду:

```
$ chmod +x hello
```

Для запуска сценария введите команду:

```
./hello
```

На экране вы увидите строку:

```
Привет, мир!
```

Чтобы вводить для запуска сценария просто `hello` (без `./`), сценарий нужно скопировать в каталог /usr/bin (точнее, в любой каталог из переменной окружения `PATH`):

```
# cp ./hello /usr/bin
```

30.4. Использование переменных в собственных сценариях

В любом серьезном сценарии вы не обойдетесь без использования *переменных*. Переменные можно объявлять в любом месте сценария, но до места их первого применения. Рекомендуется объявлять переменные в самом начале сценария, чтобы потом не искать, где вы объявили ту или иную переменную.

Для объявления переменной используется следующая конструкция:

переменная=значение

Пример объявления переменной:

```
ADDRESS=www.bhv.ru
echo $ADDRESS
```

Обратите внимание на следующие моменты:

- при объявлении переменной знак доллара не ставится, но он обязателен при использовании переменной;
- при объявлении переменной не должно быть пробелов до и после знака =.

Значение для переменной указывать вручную не обязательно — его можно прочитать с клавиатуры:

```
read ADDRESS
```

или со стандартного вывода программы:

```
ADDRESS='hostname'
```

Чтение значения переменной с клавиатуры осуществляется с помощью инструкции `read`. При этом указывать символ доллара не нужно. Вторая команда устанавливает в качестве значения переменной `ADDRESS` вывод команды `hostname`.

В Linux часто используются *переменные окружения*. Это специальные переменные, содержащие служебные данные. Вот примеры некоторых часто используемых переменных окружения:

- `BASH` — полный путь до исполняемого файла командной оболочки `bssh`;
- `BASH_VERSION` — версия `bash`;
- `HOME` — домашний каталог пользователя, который запустил сценарий;
- `HOSTNAME` — имя компьютера;
- `RANDOM` — случайное число в диапазоне от 0 до 32 767;

- `OSTYPE` — тип операционной системы;
- `PWD` — текущий каталог;
- `PS1` — строка приглашения;
- `UID` — ID пользователя, который запустил сценарий;
- `USER` — имя пользователя.

Для установки собственной переменной окружения служит команда `export`:

```
# присваиваем переменной значение
$ADDRESS=www.bhv.ru
# экспортируем переменную – делаем ее переменной окружения
# после этого переменная ADDRESS будет доступна в других сценариях
export $ADDRESS
```

30.5. Передача параметров сценарию

Очень часто сценариям нужно передавать различные параметры — например, режим работы или имя файла/каталога. Для передачи параметров используются следующие специальные переменные:

- `$0` — содержит имя сценария;
- `$n` — содержит значение параметра (`n` — номер параметра);
- `$#` — позволяет узнать количество параметров, которые были переданы.

Рассмотрим небольшой пример обработки параметров сценария. Я понимаю, что конструкцию `case-esac` мы еще не рассматривали, но общий принцип должен быть понятен (листинг 30.2).

Листинг 30.2. Пример обработки параметров сценария

```
# Сценарий должен вызываться так:
# имя_сценария параметр

# Анализируем первый параметр
case "$1" in
  start)
    # Действия при получении параметра start
    echo "Запускаем сетевой сервис"
    ;;
  stop)
    # Действия при получении параметра stop
    echo "Останавливаем сетевой сервис"
    ;;
*)
    # Действия в остальных случаях
    # Выводим подсказку о том, как нужно использовать сценарий,
    # и завершаем работу сценария
```

```
    echo "Usage: $0 {start|stop }"
    exit 1
    ;;
esac
```

Думаю, приведенных комментариев достаточно, поэтому подробно рассматривать работу сценария из листинга 30.2 мы не станем.

30.6. Массивы и *bash*

Интерпретатор `bash` позволяет использовать *массивы*. Массивы объявляются подобно переменным.

Вот пример объявления массива:

```
ARRAY[0]=1
ARRAY[1]=2

echo $ARRAY[0]
```

30.7. Циклы

Как и в любом языке программирования, в `bash` можно использовать *циклы*. Мы рассмотрим циклы `for` и `while`, хотя в `bash` доступны также циклы `until` и `select`, но они применяются довольно редко.

Синтаксис цикла `for` выглядит так:

```
for переменная in список
do
  команды
done
```

В цикле при каждой итерации переменной будет присвоен очередной элемент списка, над которым будут выполнены указанные команды. Чтобы было понятнее, рассмотрим небольшой пример:

```
for n in 1 2 3;
do
  echo $n;
done
```

Обратите внимание: список значений и список команд должны заканчиваться точкой с запятой.

Как и следовало ожидать, наш сценарий выведет на экран следующее:

```
1
2
3
```

Синтаксис цикла `while` выглядит немного иначе:

```
while условие
do
  команды
done
```

Цикл `while` выполняется до тех пор, пока истинно заданное условие. Подробно об условиях мы поговорим в следующем разделе, а сейчас напишем аналог предыдущего цикла, т. е. нам нужно вывести 1, 2 и 3, но с помощью `while`, а не `for`:

```
n=1
while [ $n -lt  4 ]
do
  echo "$n "
  n=$(( $n+1 ));
done
```

30.8. Условные операторы

В `bash` доступны два *условных оператора*: `if` и `case`. Синтаксис оператора `if` следующий:

```
if условие_1 then
    команды_1
elif условие_2 then
    команды_2
...
elif условие_N then
    команды_N
else
    команды_N+1
fi
```

Оператор `if` в `bash` работает аналогично оператору `if` в других языках программирования. Если истинно первое условие, то выполняется первый список команд, иначе — проверяется второе условие и т. д. Количество блоков `elif`, понятно, не ограничено.

Самая ответственная задача — это правильно составить условие. Условия записываются в квадратных скобках. Вот пример записи условий:

```
# Переменная N = 10
[ N==10 ]

# Переменная N не равна 10
[ N!=10 ]
```

Операции сравнения указываются не с помощью привычных знаков > или <, а с помощью следующих выражений:

- `-lt` — меньше;
- `-gt` — больше;
- `-le` — меньше или равно;
- `-ge` — больше или равно;
- `-eq` — равно (используется вместо ==).

Применять эти выражения нужно следующим образом:

```
[ переменная выражение значение|переменная ]
```

Например:

```
# N меньше 10
[ $N -lt 10 ]
# N меньше A
[ $N -lt $A ]
```

В квадратных скобках вы также можете задать выражения для проверки существования файла и каталога:

- `-e` *файл* — условие истинно, если файл существует;
- `-d` *каталог* — условие истинно, если каталог существует;
- `-x` *файл* — условие истинно, если файл является исполняемым.

С оператором `case` мы уже немного знакомы, но сейчас рассмотрим его синтаксис подробнее:

```
case переменная in
значение_1) команды_1 ;;
...
значение_N) команды_N ;;
*) команды_по_умолчанию;;
esac
```

Значение указанной переменной по очереди сравнивается с приведенными значениями (*значение_1*, ..., *значение_N*). Если есть совпадение, то будут выполнены команды, соответствующие значению. Если совпадений нет, то будут выполнены команды по умолчанию. Пример использования `case` был приведен в листинге 15.2.

30.9. Функции

В `bash` можно использовать функции. Синтаксис объявления функции следующий:

```
имя() { список; }
```

Вот пример объявления и использования функции:

```
list_txt()
{
echo "Выводим текстовые файлы"
ls *.txt
}
```

30.10. Примеры сценариев

30.10.1. Сценарий мониторинга журнала

Начнем с простого сценария мониторинга журнала (листинг 30.3). Системные журналы постоянно обновляются, и наш сценарий станет каждые 3 секунды выводить последние 15 строк выбранного вами журнала. Сценарий будет полезен при настройке системы, когда нужно постоянно просматривать журналы, чтобы понять, как система реагирует на новые настройки. Для прекращения работы сценария нужно нажать комбинацию клавиш <Ctrl>+<C>.

Листинг 30.3. Мониторинг системного журнала

```
#!/bin/bash
# Интервал обновления в секундах
INT=3

while [ true ]
do
# Выводим последние 15 строк журнала
tail -n 15 $1
# Ждем
sleep $INT
# Две пустые строки
echo; echo
done
```

Формат вызова следующий:

./сценарий файл_журнала

Например:

```
./script /var/log/messages
```

30.10.2. Переименование файлов

Следующий сценарий сложнее: он ищет в текущем каталоге файлы, в именах которых есть пробелы, и заменяет пробелы символами подчеркивания (листинг 15.4).

Листинг 30.4. Сценарий rename_blanks

```
#!/bin/bash
#

num=0                          # Сколько файлов мы переименовали
```

```
for filename in *           # Перебираем все файлы в текущем каталоге
do
# Передаем имя файла фильтру grep
# Если имя файла содержит пробел, то код завершения
# последней операции равен 0
     echo "$filename" | grep -q " "
     if [ $? -eq 0 ]
     then
# Если код завершения равен 0, переименовываем файл
       fname=$filename
       n='echo $fname | sed -e "s/ /_/g"'
       mv "$fname" "$n"
       let "num += 1"
     fi
done

echo "Переименовано файлов: $num"

exit 0
```

30.10.3. Преобразование систем счисления

Сейчас мы напишем простенький сценарий, преобразующий десятичное число в шестнадцатеричное с помощью программы `dc` (листинг 30.5). Самим преобразованием будет заниматься программа `dc`, а наш сценарий только подготовит данные для этой программы.

Листинг 30.5. Сценарий dec_hex

```
#!/bin/bash
B=16   # Основание системы счисления
# Проверяем, является ли параметр $1 числом
if [ -z "$1" ]
then
echo "Использование: dec_hex число"
exit 1
fi
# Передаем число ($1), систему счисления программе dc
echo ""$1" "$B" o p" | dc
```

30.10.4. Проверка прав пользователя

Для сценариев, требующих полномочий root, сначала нужно проверить, какой пользователь запустил сценарий. UID пользователя root всегда равен 0. Проверка, является ли пользователь, запустивший сценарий, пользователем root, может выглядеть так, как показано в листинге 30.6.

Листинг 30.6. Проверка полномочий пользователя

```
#!/bin/bash

ROOT_UID=0

if [ "$UID" -eq "$ROOT_UID" ]
then
  echo "Вы — root"
else
  echo "Вы — обычный пользователь"
fi

exit 0
```

30.10.5. Генератор имени временного файла

Довольно часто в процессе обработки данных есть необходимость в создании временных файлов. Чтобы имя временного файла всегда было уникальным, мы можем применить программу mcookie, генерирующую случайную строку для утилиты авторизации Xauth, но в своих целях. В листинге 30.7 приведен сценарий, генерирующий имя временного файла вида "tmp_*случайная_строка*".

Листинг 30.7. Генератор случайного имени файла

```
#!/bin/bash

prefix="tmp_"
suffix=`mcookie`

temp_filename=$prefix.$suffix

echo "Имя файла = \"$temp_filename\""

exit 0
```

30.10.6. Проверка свободного дискового пространства с уведомлением по электронной почте

Следующий сценарий проверяет свободное дисковое пространство сервера и отправляет администратору уведомление по e-mail, если осталось меньше 500 Мбайт дискового пространства. Сценарий целесообразно запускать через планировщик заданий (например, через cron) с заданной периодичностью (например, каждый час). Код сценария приведен в листинге 30.8.

Листинг 30.8. Проверка свободного дискового пространства

```
#!/bin/bash
# В переменную freespace будет записано свободное пространство
# на контролируемом диске - /dev/sda1 (в Мб)
freespace=`df -m | grep "/dev/sda1" | awk '{print $4}'`

# Если места на диске < 500 Mb, то отправляем письмо администратору
if [ $freespace -lt 500 ];
then
echo "Warning!!! Free space on server < 500 MB" | mail -s "Freespase is out!"
admin@server.ru
fi
```

> ***ЗАРАНЕЕ НАСТРОЙТЕ ПОЧТОВУЮ СИСТЕМУ***
>
> Для работы этого сценария нужно, чтобы почтовая система была настроена, иначе команда `mail` не сможет отправить сообщение!

30.10.7. Проверка контрольных сумм

Этот пример демонстрирует всю мощь использования `bash`. Представьте, что нам нужно написать систему проверки контрольных сумм. В эту систему будут входить два сценария: первый — станет сохранять в текущем каталоге (в файле checksum.txt) контрольные суммы всех файлов текущего каталога и его подкаталогов, второй — загружать этот список и проверять контрольные суммы по нему. Если контрольная сумма какого-то из файлов изменилась, вы получите предупреждение.

Посмотрим, как бы выглядел алгоритм первого сценария, если бы мы писали его на C:

1. Получить список файлов и каталогов текущего каталога.
2. В цикле пройтись по всему списку элементов каталога, и если текущий элемент — файл, вычислить его контрольную сумму и сохранить во временную переменную строку вида `"КС имя_файла"`.
3. Открыть файл checksum.txt для записи.
4. Записать временную переменную в открытый файл.
5. Закрыть файл

И это я еще упростил, поскольку мы не обрабатываем подкаталоги — чтобы не описывать здесь рекурсивный алгоритм. Так вот в `bash`, поскольку мы воспользуемся командой `find`, вся эта функциональность может быть реализована с помощью двух строк (листинг 30.9).

Листинг 30.9. Сценарий mkcksum

```
#!/bin/bash
find . -type f -exec sha256sum {} \; > checksum.txt
```

Эта команда вычисляет контрольные суммы всех файлов в текущем каталоге и во всех его подкаталогах. Если вам нужно контролировать только определенные файлы, задайте опцию `-name` и уточните, какие файлы вы хотите контролировать, например команда `-name "*.docx"` будет вычислять контрольные суммы только DOCX-файлов:

```
find . -type f -name "*.docx" -exec sha256sum {} \; > checksum.txt
```

Посмотрим на вывод команды `find . -type f -exec sha256sum {} \;`:

```
1f0db0fa6ffbf473c5ee7...8db31547360666c533633f6414c7  ./menu2.html
328065d701eed85738a0...262c4285dc8103920c38098a2b3ea  ./menu3.html
d951051a6bed2ac2d3...fb879b8bb2e72d75c937a5c44546487  ./menu1.html
```

Я сократил контрольные суммы, чтобы они помещались в одну строку. Вот такой вывод и будет записан в файл checksum.txt.

Теперь перейдем ко второму сценарию (листинг 3.10).

Листинг 30.10. Сценарий cksum

```
#!/bin/bash
sha256sum -c < checksum.txt | grep FAILED
```

Этот скрипт запускает команду `sha256sum` в режиме проверки (`-c`), а в качестве строк для проверки использует строки из файла checksum.txt. Все это делает `bash` — берет каждую отдельную строку и передает на вход checksum.txt. Далее вывод программы отфильтровывается — нас интересуют файлы, не прошедшие проверку. Если будет встречена строка `FAILED`, значит, контрольная сумма какого-то файла изменилась. Как видите, все достаточно просто.

ЧАСТЬ VII

Сетевые службы

ASTRA LINUX®

ГЛАВА 31

Служба Astra Linux Directory (ALD)

- Установка пакетов
- Подготовка к настройке
- Настройка сервера ALD
- Настройка клиента

31.1. Установка пакетов

Служба Astra Linux Directory (ALD) является некоторым подобием Active Directory от Microsoft. Разумеется, обе эти технологии несовместимы между собой, но суть у них одна и та же — это доменные службы.

Служба ALD представляет собой систему управления Единым пространством пользователей (ЕПП). При этом она является надстройкой над технологиями LDAP, Kerberos 5 и CIFS, обеспечивающей автоматическую настройку всех необходимых файлов конфигурации служб, реализующих указанные технологии, а также предоставляющей интерфейс управления и администрирования.

Для установки ALD вам необходимо установить три пакета:

- ald-server-common — собственно, сам сервер ALD;
- fly-admin-ald-server — графический инструмент для администрирования домена и его клиентов;
- smolensk-security-ald — инструмент для управления мандатными привилегиями в Astra Linux Special Edition.

Для установки вы можете использовать Synaptic или просто ввести в терминале команду:

```
sudo apt install fly-admin-ald-server ald-server-common smolensk-security-ald
```

Во время установки пакета ald-server-common вам нужно будет задать пароль администратора LDAP (рис. 31.1). Исходя из соображений безопасности, он должен отличаться от пароля учетной записи!

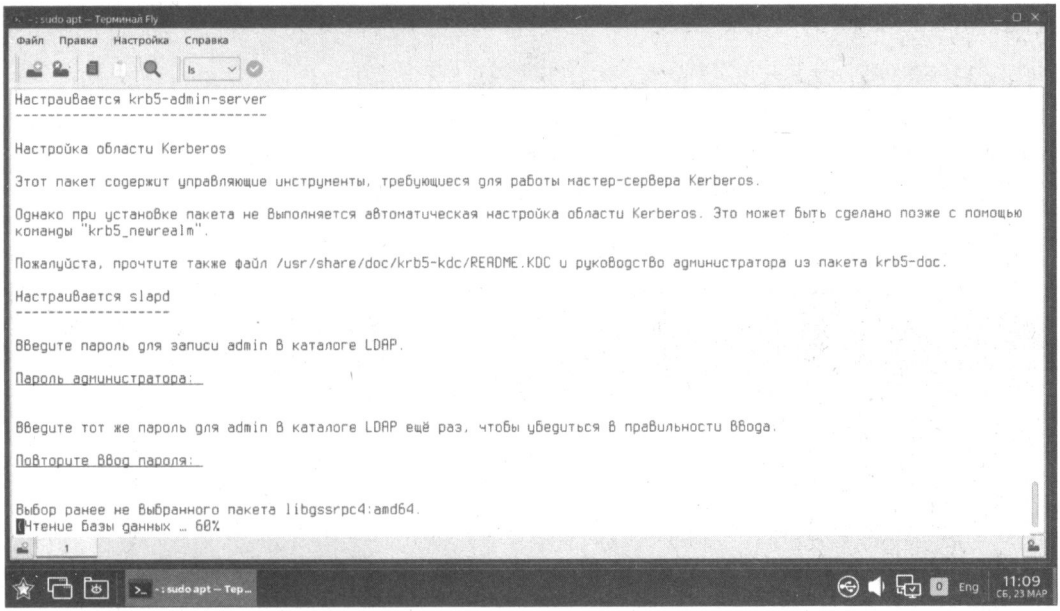

Рис. 31.1. Задаем пароль администратора каталога LDAP

31.2. Подготовка к настройке

Прежде всего необходимо определить постоянный IP-адрес сервера (рис. 31.2) — это должен быть статический IP-адрес, поскольку использование динамического IP-адреса не допускается. Также не допускается использование адреса 127.0.0.1.

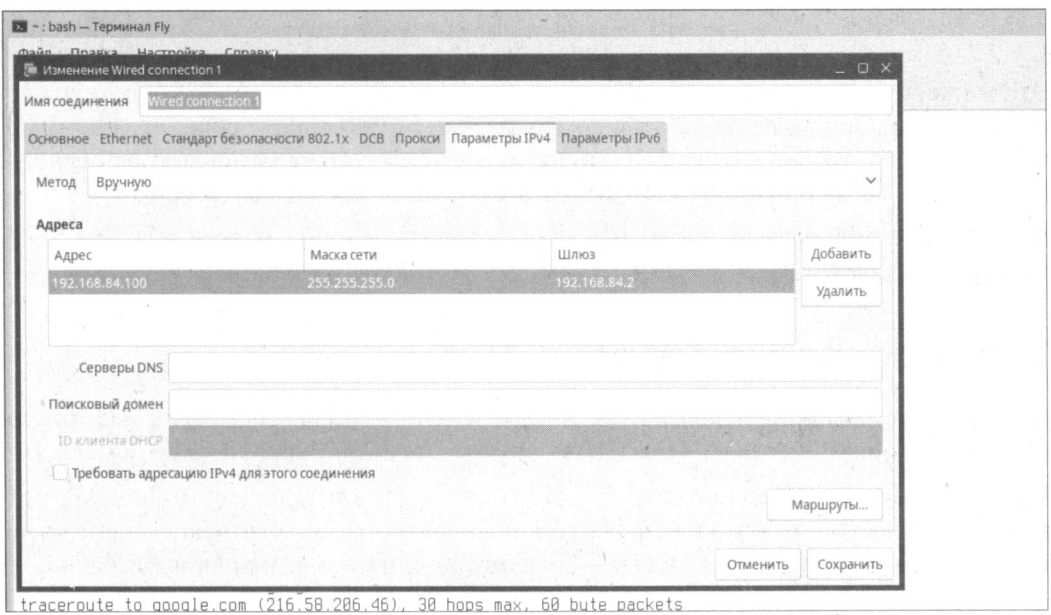

Рис. 31.2. Настройка сети на использование статического IP-адреса

Далее нужно задать полное доменное имя сервера с помощью команды `hostnamectl`:

```
sudo hostnamectl set-hostname server.example.com
```

Это имя должно быть доступно для разрешения. В идеале нужно было бы настроить DNS-службу внутри вашей сети, но, пока вы этого не сделали, откройте файл **/etc/hosts** и укажите IP-адрес для сервера:

```
192.168.84.100 server.example.com
```

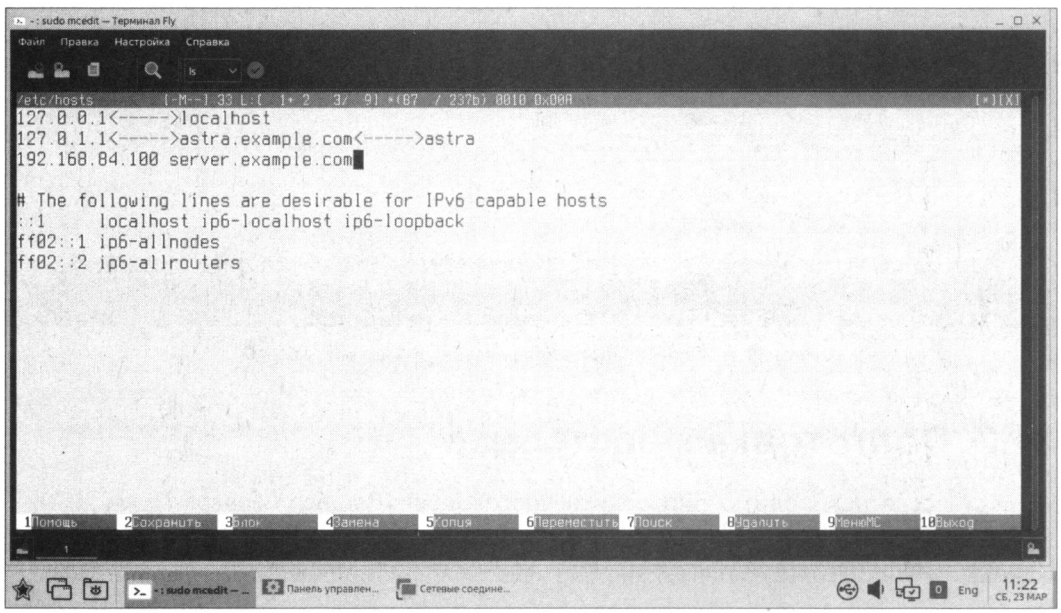

Рис. 31.3. Редактирование файла /etc/hosts

Это же действие нужно выполнить на всех компьютерах, которые вы будете добавлять в домен. Они должны «знать» доменное имя server.example.com. Если компьютеров немного, то есть смысл сделать это вручную. Если же компьютеров несколько десятков, тогда придется настраивать службу DNS для вашего домена.

На этом все подготовительные действия завершены, и мы можем перейти к настройке сервера ALD.

31.3. Настройка сервера ALD

Что означает «выполнить настройку сервера ALD»? Если вы работали с AD, то уже догадались, о чем идет речь. Правильно, нам нужно создать контроллер домена.

Выполнить настройку сервера ALD можно как через консоль, так и с использованием графического инструмента, который вы можете запустить командой меню **Пуск | Панель управления | Сеть | Доменная политика безопасности**. Так написано в документации... Но по факту при запуске этого конфигуратора ничего не происходит, а если запустить его через консоль, то вы увидите множество ошибок,

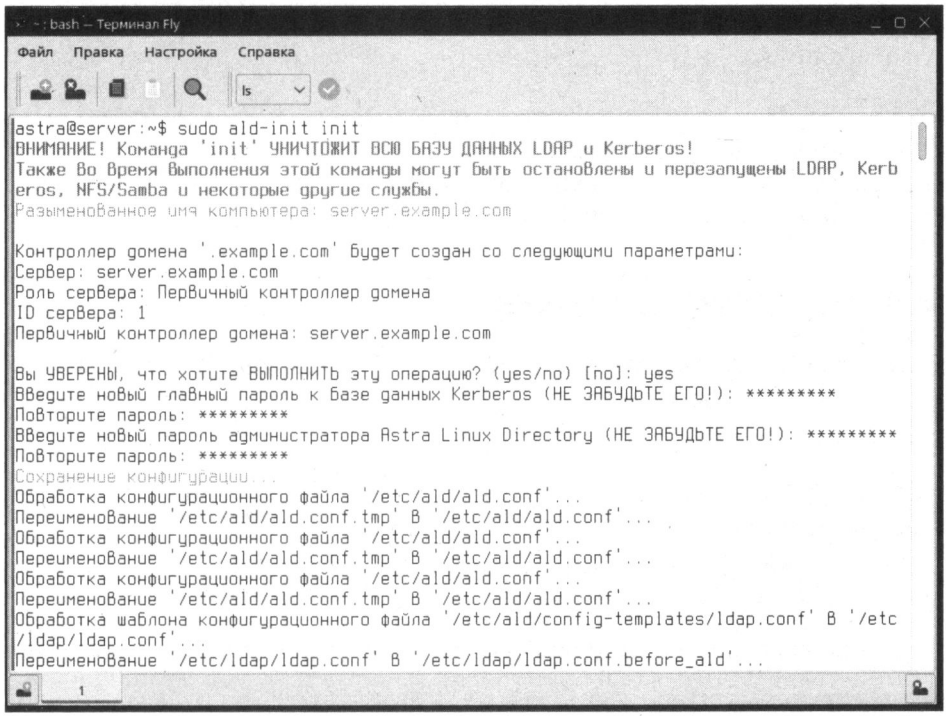

Рис. 31.4. Установка контроллера домена

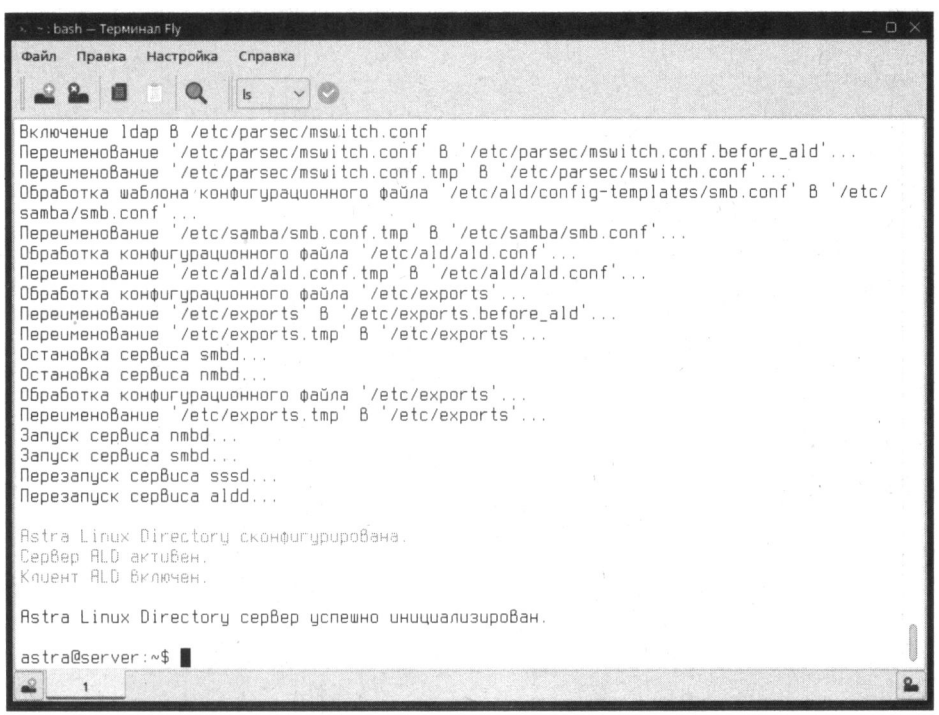

Рис. 31.5. Инициализация завершена

с которыми вам точно (как и мне) не захочется разбираться. Поэтому открываем консоль и вводим команду (рис. 31.4):

```
sudo ald-init init
```

Вам будет предложено задать главный пароль к БД Kerberos, а также новый пароль администратора ALD. Спустя некоторое время вы получите сообщение о том, что ALD-сервер успешно инициализирован (рис. 31.5).

31.4. Настройка клиента

Теперь на каждом компьютере, который нужно будет включить в ALD-домен, надо установить пакет ald-client-common:

```
sudo apt install ald-client-common
```

Его графическую версию можно не устанавливать — есть сомнения относительно того, что она будет работать.

Далее в файле /etc/hosts нужно прописать IP-адрес нашего сервера или же настроить DNS-сервер, что правильнее.

Для подключения клиентов с помощью командной строки используйте команду:

```
sudo ald-client join  <имя_компьютера_контроллера_домена>
[--hostname <имя_компьютера_клиента>]
```

После выполнения этой команды нужно перезагрузить компьютер, чтобы ALD уж совсем был бы похож на AD.

ASTRA LINUX®

ГЛАВА 32

Samba: интеграция с Windows-сетью

- Установка Samba
- Базовая настройка Samba
- Настройка общих ресурсов
- Подключение Linux-компьютера к домену Active Directory

32.1. Установка Samba

Взаимодействие с сетью Microsoft в Linux обеспечивает пакет samba. Если вы хотите использовать общие ресурсы Windows-сети, установите этот пакет, — он позволяет не только пользоваться общими ресурсами сети, но и предоставлять собственные ресурсы Windows-пользователям. Причем все происходит так, что Windows-пользователи даже не заметят разницы.

Кроме того, Samba может использоваться для подключения Linux-компьютера к домену Active Directory, и, более того, с помощью Samba можно построить собственный контроллер Active Directory. Этот вариант мы рассматривать не станем, поскольку в Astra Linux есть служба ALD, которая была рассмотрена ранее (см. *главу 31*). А вот «расшаривание» ресурсов Linux-компьютера и подключение Linux-компьютера к домену AD мы рассмотрим.

Для установки Samba вам нужно установить пакет fly-admin-samba. Он содержит графический инструмент настройки общего доступа к папкам — удобнее использовать его, нежели редактировать конфигурационные файлы (хотя начнем мы все же с редактирования конфигурационных файлов — ну как же в Linux без умения это делать?). В качестве разрешения зависимостей при установке этого пакета будут установлены все необходимые пакеты, в том числе и пакет samba.

32.2. Базовая настройка Samba

Основной конфигурационный файл Samba — /etc/samba/smb.conf. Откройте его и перейдите к секции `[global]`.

Прежде всего, измените в ней параметр WORKGROUP — он задает имя рабочей группы или домена NT:

```
workgroup = WORKGROUP
```

Здесь для рабочей группы задано имя WORKGROUP. Странно конечно, но таково значение по умолчанию. Имя, разумеется, у вас должно быть другим — посмотрите на своих Windows-машинах, как называется ваша рабочая группа, и добавьте сюда это название:

```
workgroup = ВАШЕ_НАЗВАНИЕ
```

Можете также установить параметр server string (он ранее назывался comment) — это описание вашего компьютера:

```
server string = My Linux computer
```

Установите параметр security — если у вас сеть «клиент-сервер», то нужно выбрать параметр server, а если сеть одноранговая (т. е. без выделенного сервера), выберите user или share:

```
security = share
```

Имя гостевой учетной записи установите так:

```
guest account = guest
```

Следует настроить и кодировки:

```
unix charset = UTF-8
dos charset = UTF-8
display charset = UTF-8
```

Для того чтобы Samba работала быстрее, установите следующие опции (что они означают, мы разберемся чуть позже):

```
socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192
dns proxy = no
```

Параметр interfaces определяет интерфейсы, на которых должен работать сервис smb, — укажите те интерфейсы, которые связывают вашу машину с Windows-сетями:

```
interfaces = 192.168.0.22/24
```

А теперь позволю себе несколько комментариев для пользователей более ранних версий Samba:

- параметр server string ранее назывался comment;
- параметры client code page и character set больше не поддерживаются — теперь вместо них используются параметры unix charset, dos charset и display charset:
  - параметр unix charset задает кодировку, в которой хранятся файлы конфигурации Samba;

- параметр `dos charset` — кодировку для Windows-клиентов;
- параметр `display charset` — кодировку для Samba-клиентов;

♦ текущая версия Samba полностью поддерживает кодировку UTF-8 (как и современные версии Linux и Windows), поэтому проблем с UTF-8 возникнуть не должно, и мы спокойно задаем эту кодировку, как и показано чуть ранее.

32.3. Настройка общих ресурсов

Теперь осталось сконфигурировать ресурсы, которые вы хотите предоставить в общее пользование. Фрагмент, приведенный в листинге 32.1, нужно добавить в файл конфигурации Samba.

Листинг 32.1. Секция `[public]`

```
[public]
# общий каталог, комментарий для ресурса задается директивой comment
comment = Public Directory
# путь
path = /var/samba
# не только чтение
read only = no
# разрешить запись
writable = yes
# разрешить гостевой доступ
guest ok = yes
# разрешить просмотр содержимого каталога
browseable = yes
```

В этом случае общим ресурсом нашего компьютера будет каталог /var/samba. В него другие пользователи смогут записывать свои файлы (`read only = no`, `writeable = yes`), естественно, они смогут их и читать (`browseable = yes`). Проверка имени пользователя и пароля для доступа к ресурсу не нужна (`guest ok = yes`) — используется так называемый *гостевой* доступ. Комментарий `Public Directory` увидят другие пользователи Windows-сети при просмотре ресурсов вашего компьютера.

> ***Директивы comment и server string***
>
> Как уже было отмечено ранее, начиная с третьей версии Samba в ее конфигурационном файле произошли небольшие изменения. В секции `[global]`, описывающей глобальные параметры Samba, теперь вместо директивы `comment` используется директива `server string`, однако при описании разделяемых ресурсов (т. е. каталогов, к которым вы предоставили общий доступ) используется та же директива `comment`.

Рассмотрим еще один пример, позволяющий сделать общими домашние каталоги пользователей, — секцию `[homes]` (листинг 32.2).

Листинг 32.2. Секция [homes]

```
[homes]
 comment = Home Directories
 browseable = no
 valid users = %S

# запись запрещена, только просмотр
 writable = no

# маска при создании файлов, нужна если writable=yes
create mask = 0600
# маска при создании каталогов, нужна если writable=yes
directory mask = 0700
```

В листинге 32.3 приведен пример предоставления общего доступа к внешнему жесткому диску, который монтируется к каталогу /mnt/ext_usb через файл /etc/fstab.

Листинг 32.3. Пример общего доступа к внешнему HDD

```
[extusb]
   comment = External USB HDD for server backup
   read only = no
   writable = yes
   locking = no
# каталог /mnt/ext_usb должен существовать и являться точкой монтирования
   path = /mnt/ext_usb
   public = yes
# разрешить просмотр содержимого каталога
   browseable = yes
```

Если редактировать конфигурационные файлы вам не хочется, откройте Панель управления, перейдите в раздел **Сеть** и запустите инструмент **Общие папки Samba**. Для добавления общего ресурса нажмите кнопку **+** и в открывшейся панели **Настройка ресурса** (рис. 32.1) укажите путь и режим доступа. Как можно видеть здесь (для примера) прописан путь /tmp (хотя конфигуратор рекомендует создавать общие ресурсы в папке /srv/samba) и установлен режим доступа **Чтение/Запись**.

Теперь нажмите кнопку с красным квадратом для остановки сервиса Samba и кнопку с зеленым треугольником для его запуска. Так вы перезапустите Samba без использования консоли. Перезапуск необходим, чтобы изменения, внесенные конфигуратором в файл smb.conf, вступили в силу.

После этого откройте файловый менеджер, перейдите в группу **Сеть**, выберите вашу рабочую группу, ваш компьютер, и вы увидите «расшаренный» вами общий ресурс (рис. 32.2).

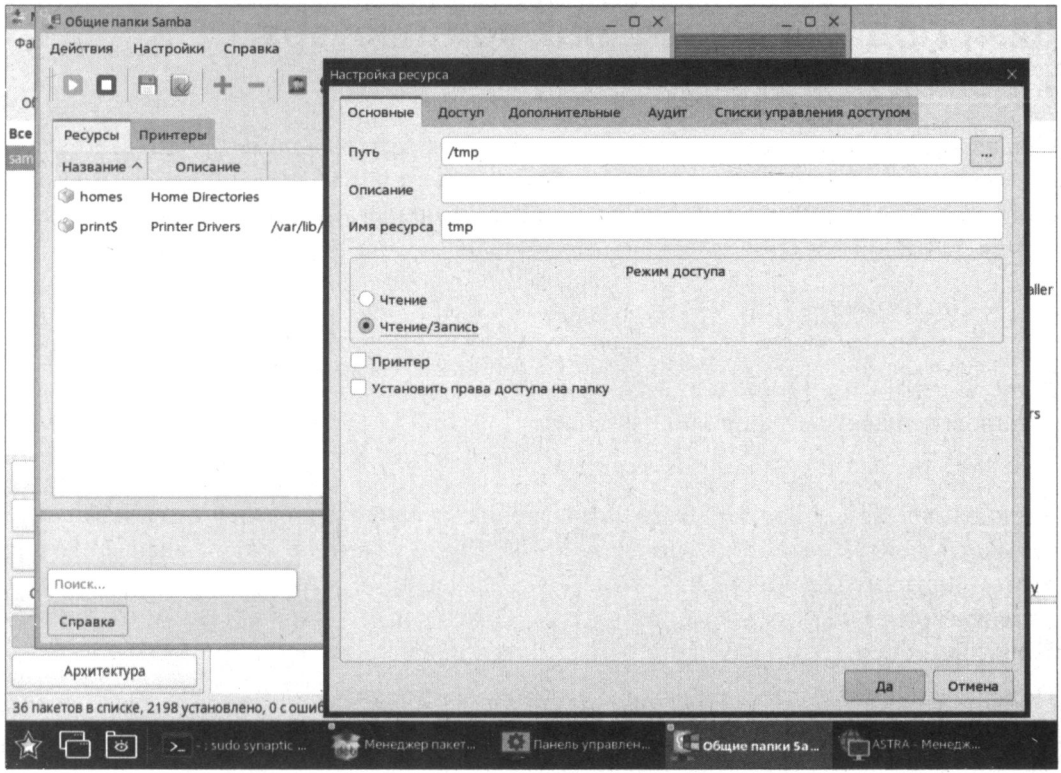

Рис. 32.1. Настройка общего ресурса

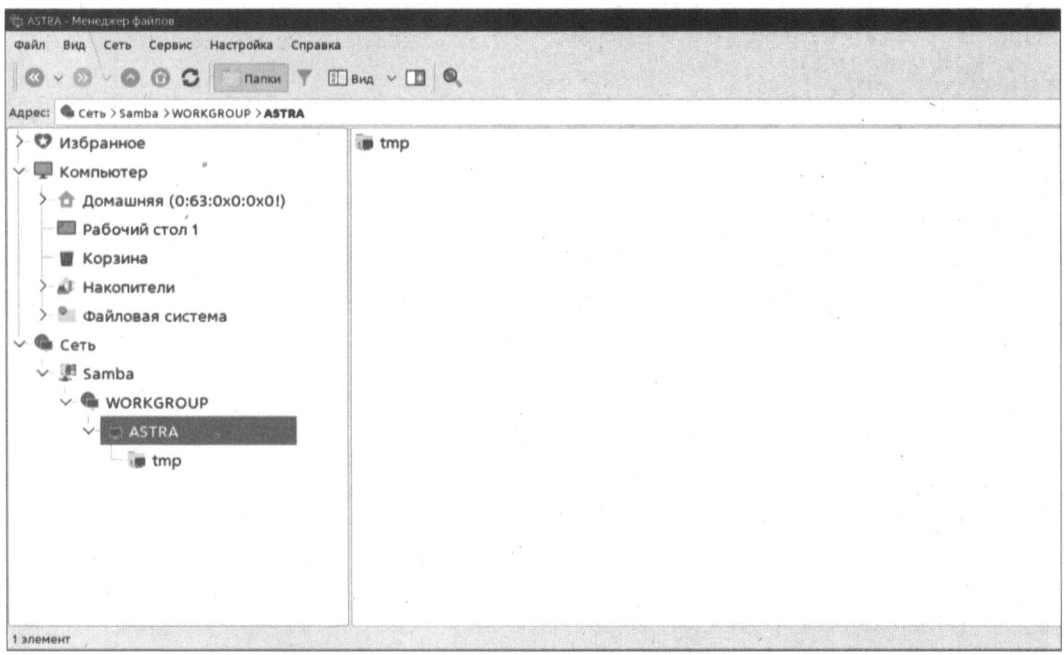

Рис. 32.2. Доступ к общему ресурсу настроен

32.4. Подключение Linux-компьютера к домену Active Directory

Существуют два способа подключения Linux-компьютера к домену ActiveDirectory: с использованием инструмента sssd и с помощью winbind. Второй способ мы рассматривать не станем, поскольку он устарел, и прибегать к нему больше не рекомендуется.

> ***Синхронизируйте часы***
>
> Прежде чем приступить к настройке, убедитесь, что часы контроллера домена и компьютера, который вы хотите включить в состав AD, синхронизированы!

Установите пакет fly-admin-ad-sssd-client:

```
sudo apt install fly-admin-ad-sssd-client
```

Установив пакет, откройте Панель управления, перейдите в раздел **Сеть** и запустите инструмент **Настройка клиента SSSD Fly**. Укажите в открывшейся панели этого инструмента (рис. 32.3) имя домена, адрес сервера времени (например, **time.windows.com**), имя администратора домена и его пароль. Нажмите кнопку **Подключиться**, и ваш компьютер будет подключен к домену.

Подключение можно выполнить и из командной строки:

```
sudo astra-ad-sssd-client -d example.com -u Администратор
```

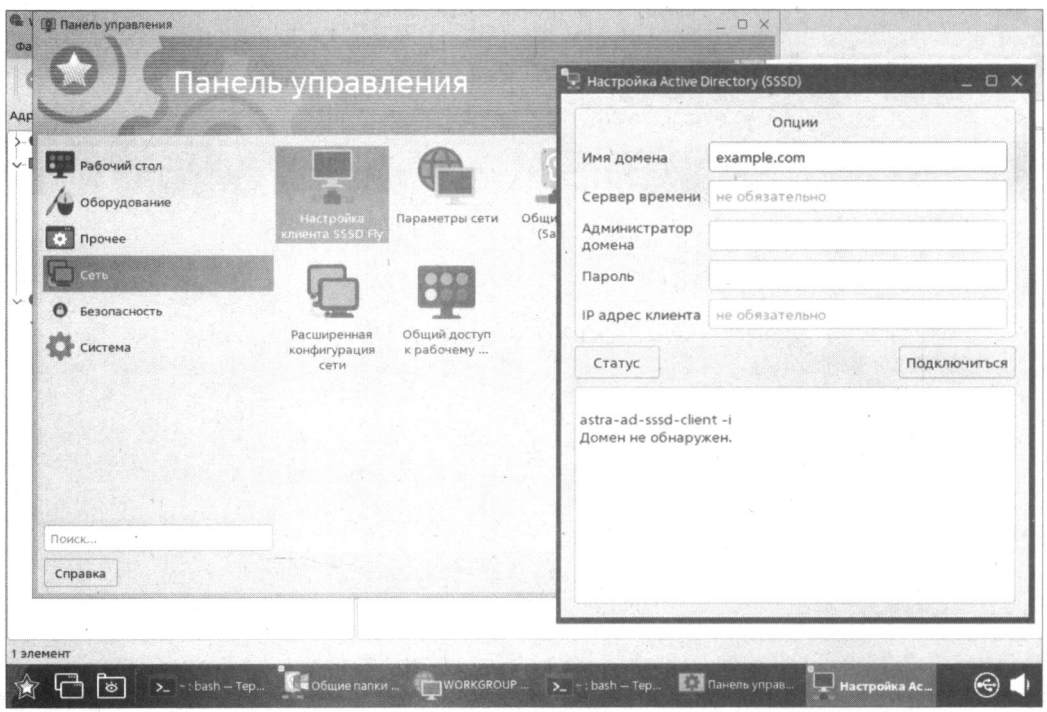

Рис. 32.3. Подключение к домену AD

Здесь параметр `-d` задает имя домена, а параметр `-u` — имя администратора домена. Вывод команды будет таким:

```
compname = astra
domain = example.com
username = admin
введите пароль администратора домена:
ok
продолжать ? (y\N)
y
настройка сервисов...
Завершено.
Компьютер подключен к домену.
Для продолжения работы необходимо перезагрузить компьютер!
```

Для завершения подключения, как вам в выводе команды и рекомендуется, перезагрузите компьютер:

```
sudo reboot
```

ASTRA LINUX®

ГЛАВА 33

DFS: разделяемые ресурсы

- Что такое DFS?
- Монтирование разделяемых ресурсов DFS
- Автоматическое монтирование разделяемых ресурсов DFS

33.1. Что такое DFS?

Распределенная файловая система, или DFS (Distributed File System) — компонент Microsoft Windows, используемый для облегчения доступа и управления файлами, физически распределенными по сети.

Основное назначение DFS — это упрощение доступа к файлам. У пользователей DFS образуется единый сетевой ресурс для доступа к файлам, хотя файлы при этом физически могут находиться на разных серверах. И даже если файлы физически «переедут» на другой сервер, это никак не отразится на пользователях, которые продолжат использовать все тот же единый ресурс.

В этой небольшой главе будет показано, как работать с DFS в Astra Linux. По сути, это еще один инструмент, обеспечивающий интеграцию Astra Linux в существующую Windows-инфраструктуру.

> ***Внимание!***
> Для работы с DFS вам нужно ядро Linux 5.4 или более новое (5.10 или 6.10).

33.2. Монтирование разделяемых ресурсов DFS

Сразу нужно отметить, что настройка самих ресурсов DFS осуществляется на серверах Windows Server и в этой книге не рассматривается.

Для начала работы с DFS вам нужно установить пакет keyutils:

```
sudo apt install keyutils
```

Для монтирования разделяемого ресурса DFS вы можете указать, как полное доменное имя сервера, так и просто имя домена. Примеры:

```
mount -t cifs //server.example.com/dfs1 ~/mnt -o cruid=admin,sec=krb5
mount -t cifs //example.com/dfs1 ~/mnt -o cruid=admin,sec=krb5
```

Параметр `cruid` позволяет указать имя пользователя для аутентификации, а параметр `sec` — тип безопасности (Kerberos 5). Сам ресурс здесь носит имя `dfs2`, а монтируется он к каталогу `~/mnt`. Пароль доступа к нему будет запрошен в процессе выполнения команды.

33.3. Автоматическое монтирование разделяемых ресурсов DFS

Для автоматического монтирования DFS необходимо отредактировать файл **/etc/fstab**. Формат записи следующий:

```
//IP/имя_ресурса  /каталог  cifs  credentials=/etc/win-credentials,
file_mode=0755,dir_mode=0755 0       0
```

Здесь мы используем одну хитрость. Вместо того чтобы записать в **/etc/fstab** параметры доступа (логин и пароль), мы указываем, что они будут храниться в файле **/etc/win-credentials**. Пример содержимого этого файла приведен далее (вам нужно прописать в нем свой домен, имя пользователя и пароль):

```
username=user
password=password
domain=domain
```

В завершение мы ограничиваем доступ к этому файлу, чтобы никто не прочитал наши реквизиты:

```
sudo chown root: /etc/win-credentials
sudo chmod 600 /etc/win-credentials
```

ASTRA LINUX®

ГЛАВА 34

Настройка веб-сервера Apache

- Самый популярный веб-сервер
- Установка веб-сервера и интерпретатора PHP
- Тестирование настроек
- Файл конфигурации веб-сервера
- Управление запуском сервера Apache
- Оптимизация Apache

34.1. Самый популярный веб-сервер

Apache — это веб-сервер с открытым исходным кодом. История его развития началась в 1995 году — тогда Apache был всего лишь «заплаткой», устраняющей ошибки популярного в то время веб-сервера NCSA HTTPd 1.3. Считается, что отсюда произошло и название Apache (a patchy, заплатка). Сейчас Apache — самый популярный веб-сервер в Интернете.

Основные достоинства Apache — надежность, безопасность и гибкость настройки. Apache позволяет подключать различные модули, добавляющие в него новые возможности, — например, можно подключить модуль, обеспечивающий поддержку PHP или любого другого веб-ориентированного языка программирования.

Но есть у Apache и недостатки — без этого никак, у любой медали всегда есть оборотная сторона. Основной недостаток — отсутствие удобного графического интерфейса администратора. Да, настройка Apache осуществляется путем редактирования его конфигурационного файла. В Интернете можно найти простые конфигураторы Apache, но их возможностей явно не хватает для настройки всех функций этого веб-сервера.

Казалось, что может быть сложного в установке веб-сервера? Он везде ведь одинаковый. Но у Astra Linux — свой путь. Его мы и рассмотрим в этой главе.

34.2. Установка веб-сервера и интерпретатора PHP

Долгое время в репозиториях большинства дистрибутивов Linux параллельно содержались две версии Apache: 1.3 и 2.2. Сейчас версия 1.*x* больше не поддерживается, а вместо версии 2.2 обычно используется версия 2.4. Поэтому теперь можно не ломать себе голову вопросом, какую версию лучше установить.

Итак, приступим к установке Apache. В большинстве современных дистрибутивов нужный нам пакет называется apache2, а в некоторых устаревших может называться httpd. Для установки Apache введите команду:

```
sudo apt install apache2
```

Менеджер пакетов сообщит вам, что нужно установить дополнительные пакеты (рис. 34.1).

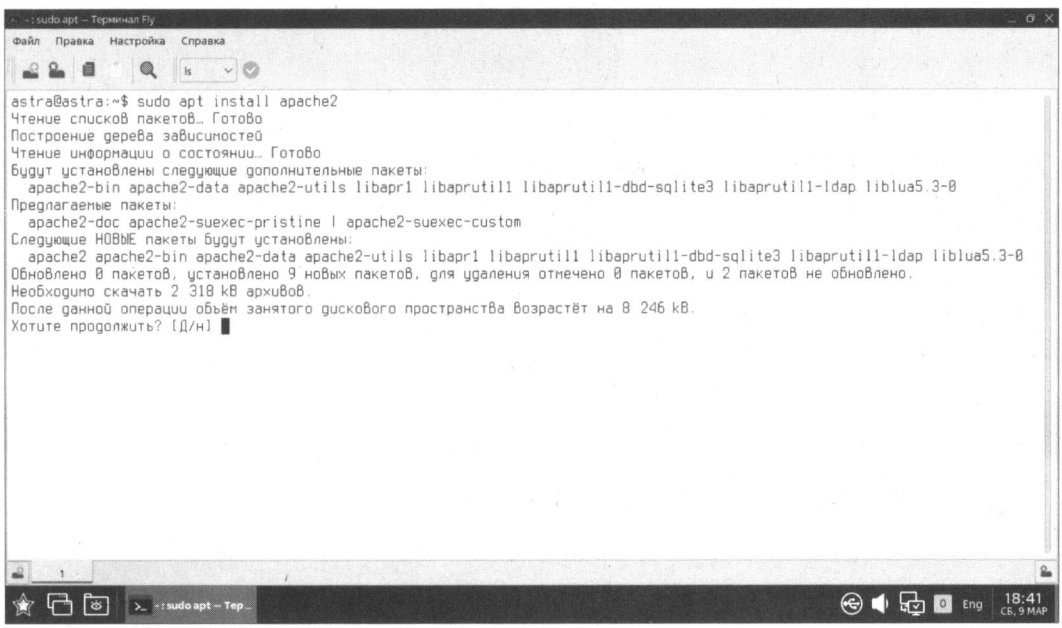

Рис. 34.1. Дополнительные пакеты для установки Apache

Чтобы сразу убить двух зайцев, установите еще и пакет libapache2-mod-php. Этот пакет — виртуальный, и при его установке система попросит вас уточнить номер версии PHP (рис. 34.2). Как можно видеть, предлагается установить PHP версии 8.1 или 7.3. Разумеется, лучше устанавливать версию, самую новую из доступных. Поэтому вводим команду:

```
sudo apt install libapache2-mod-php8.1
```

Теперь надо установить дополнительные расширения PHP. Как минимум вам понадобятся пакеты php8.1-mysql (поддержка СУБД MySQL) и php8.1-gd (графи-

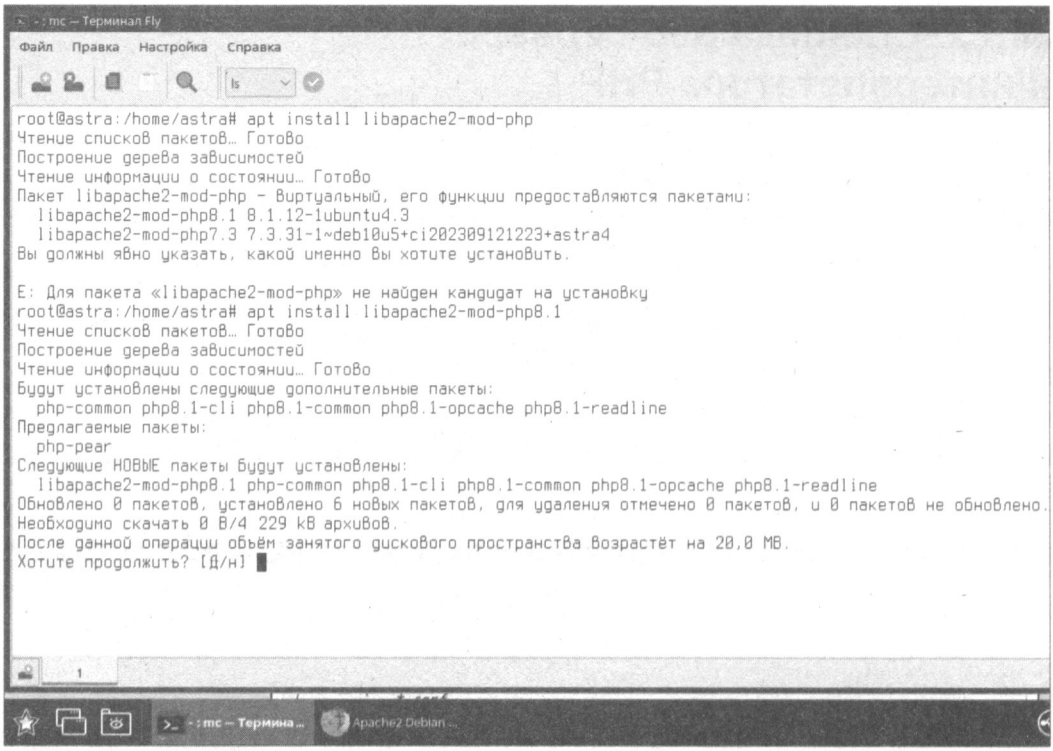

Рис. 34.2. Установка PHP

ческая библиотека), поэтому вводим команду (номер версии можно заменить на 7.3, если вы выбрали эту версию):

```
sudo apt install php8.1-mysql php8.1-gd
```

Если система вам сообщит, что таких пакетов нет, значит «идем» в файл /etc/apt/sources.list, раскомментируем закомментированные строки (рис. 34.3), после чего вводим команду `sudo apt update` и повторяем попытку установки дополнительных расширений PHP. Как показано на рис. 34.4, дополнительные пакеты теперь доступны для установки.

Но мало установить PHP — его нужно еще и настроить. Конфигурация PHP хранится в файле php.ini, вот только в системе существуют три файла php.ini, и каждый из них содержит конфигурацию, подходящую для того или иного случая использования:

- /etc/php/<*номер версии*>/apache2/php.ini — конфигурация скриптов, запущенных посредством веб-сервера;
- /etc/php/<*номер версии*>/cli/php.ini — конфигурация скриптов, запускаемых из консоли (когда вы вводите команду `php <имя скрипта.php>`);
- /etc/php/<*номер версии*>/fpm/php.ini — конфигурация для FPM, т. е. когда PHP запускается через nginx.

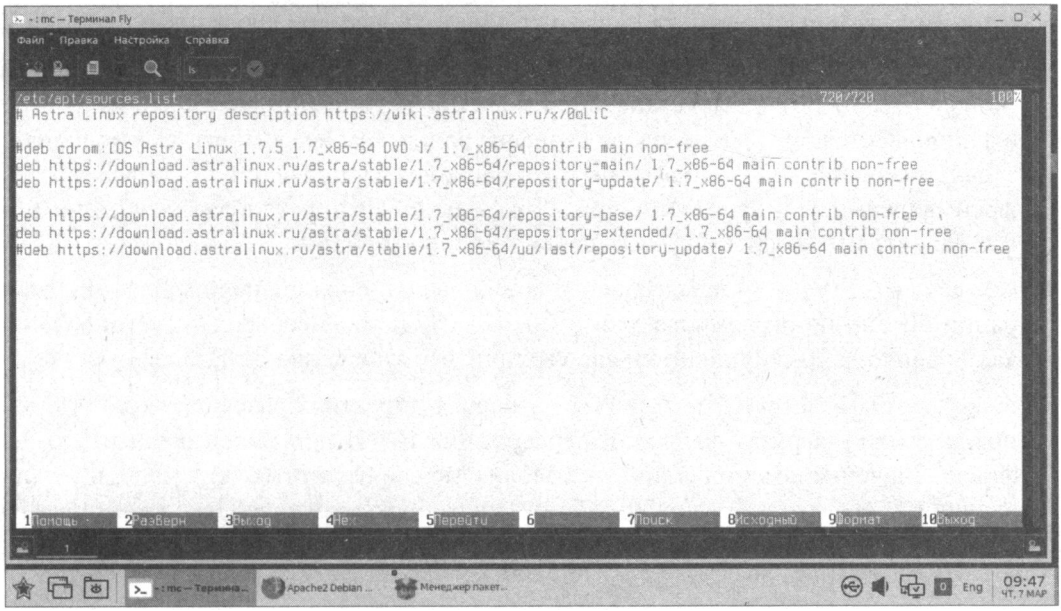

Рис. 34.3. Раскомментируйте строки для установки дополнительных пакетов

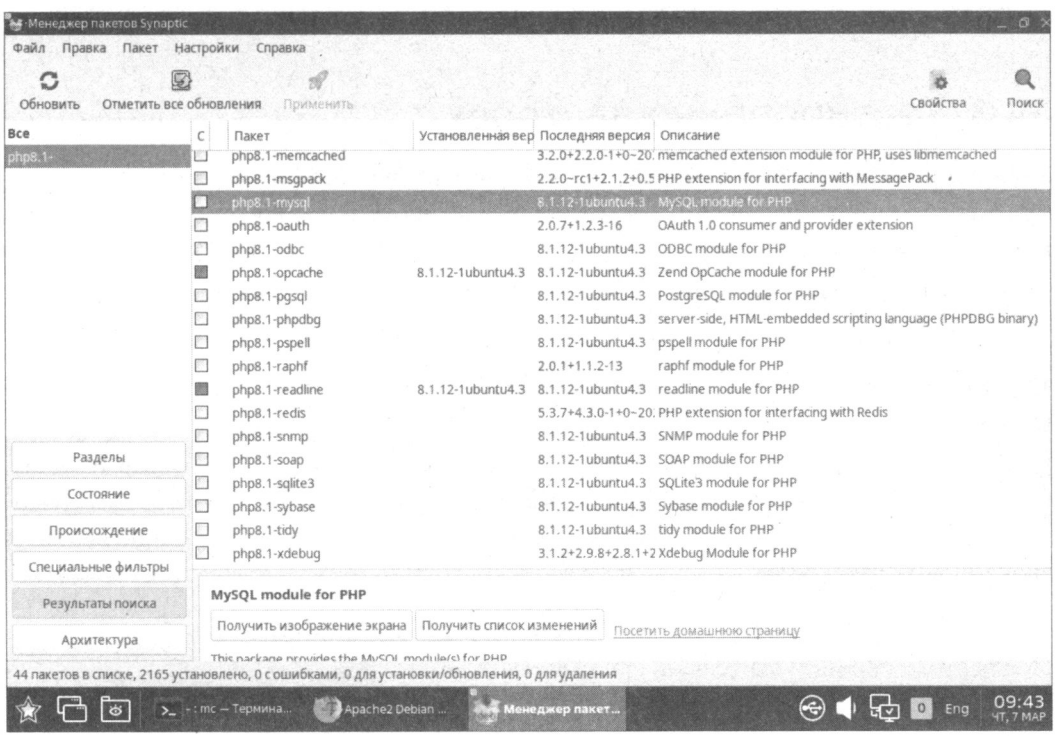

Рис. 34.4. Дополнительные пакеты доступны для установки

Как минимум вам нужно изменить в этих файлах следующие директивы:

- `memory_limit` — лимит памяти, доступный скрипту. Для консольных скриптов (конфигурация cli) здесь установлено значение `-1`, а это означает, что память не ограничивается. Для веб-сервера (конфигурации apache2 и fpm) по умолчанию задано значение 128 Мбайт, но для сложных приложений, основанных на фреймворках Zend, Symfony, Laravel, этого мало. Надо установить как минимум 512 Мбайт, а то и больше — все зависит от приложения;
- `max_execution_time` — максимальное время выполнения сценария. Для конфигурации cli оно не ограничивается (значение 0), для веб-сервера — установлено, как правило, в 30 секунд, но лучше увеличить это значение до 120 секунд;
- `file_uploads` и `upload_max_filesize` — первая директива разрешает (`On`) или запрещает (`Off`) загрузку файлов на сервер, вторая — задает максимальный размер файла. Значение по умолчанию — 2 Мбайт, но сейчас этого очень мало, и нужно увеличить его хотя бы до 20 Мбайт (здесь нужно руководствовать здравым смыслом и функциональностью приложения. Если, например, приложение подразумевает загрузку видеофайлов, то и 20 Мбайт может не хватить).

Если вы изменили конфигурацию для Apache, то, чтобы изменения вступили в силу, его нужно перезапустить. А если вы изменили конфигурацию PHP-FPM, то нужно выполнить его перезапуск командой:

```
sudo systemctl restart php-fpm
```

34.3. Тестирование настроек

Теперь протестируем наш веб-сервер. По идее, после установки сервер должен запуститься автоматически, но в некоторых дистрибутивах его придется запустить вручную (см. *разд. 34.5*).

Запустите сервер или убедитесь, что он запущен, откройте браузер и введите адрес:

```
http://localhost
```

Должна открыться тестовая страница Apache, но вместо этого вы видите ошибку 500 (рис. 34.5).

Цель этой книги — быть вам максимально полезной, а не заставлять вас читать документацию и гуглить. В общем, чтобы не вдаваться в подробности, вам нужно открыть файл /etc/apache2/apache2.conf, найти директиву `AstraMode` и присвоить ей значение `off` (рис. 34.6):

```
AstraMode off
```

После этого перезапустите веб-сервер командой:

```
sudo systemstl restart apache2
```

и обновите страницу в браузере. Вы должны увидеть то, что увидели бы в любом другом дистрибутиве сразу, — стартовую страницу Apache (рис. 34.7).

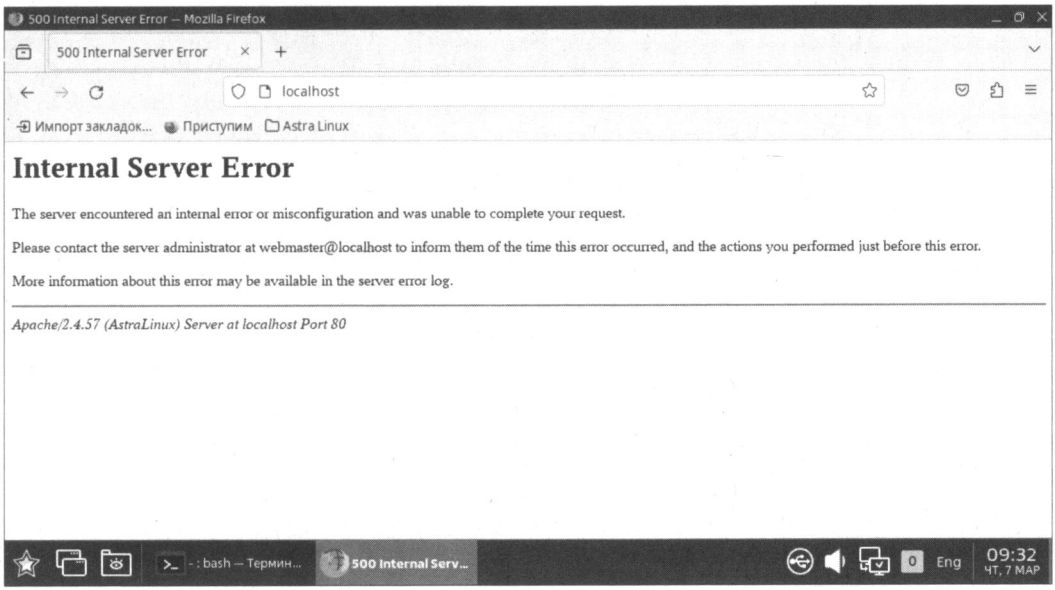

Рис. 34.5. Ошибка 500

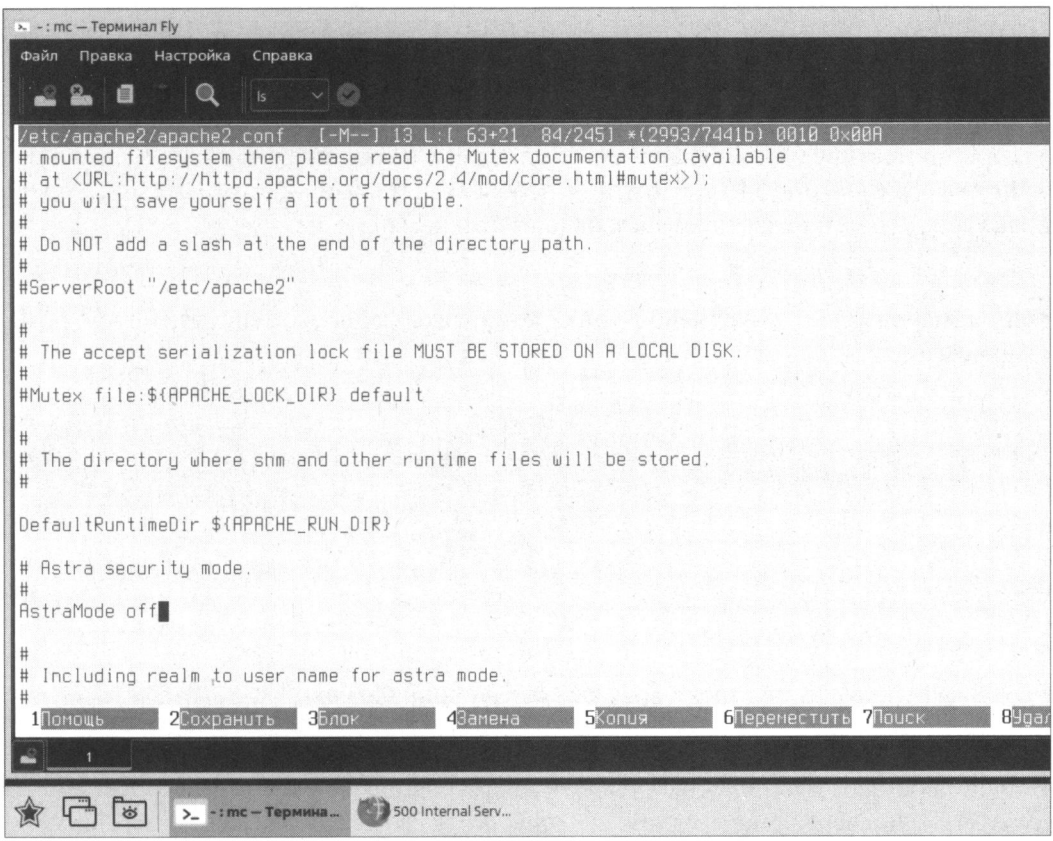

Рис. 34.6. Редактирование файла /etc/apache2/apache2.conf

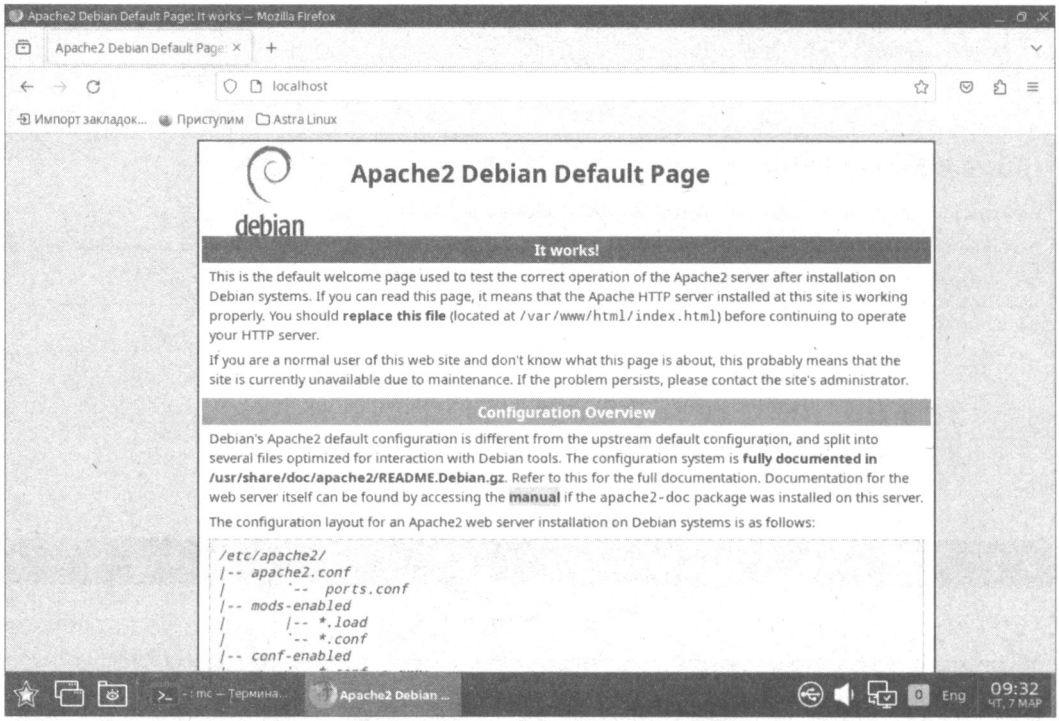

Рис. 34.7. Стартовая страница Apache

Теперь протестируем поддержку PHP. Для этого поместите в каталог /var/www/html/ файл test.php (листинг 34.1). Учтите — чтобы создать файл в этом каталоге, нужны права root. Для его редактирования надо ввести команду:

```
sudo mcedit /var/www/html/test.php
```

Если команда `mcedit` у вас недоступна, установите файловый менеджер mc с помощью команды:

```
sudo apt install mc
```

Листинг 34.1. Файл test.php

```
<?

phpinfo();

?>
```

Создав файл, введите в строке браузера следующий адрес:

```
http://localhost/test.php
```

В окне браузера вы должны увидеть информацию о своем сервере и о PHP (рис. 34.8). Да, версия почему-то из Ubuntu. Хотя все установлено в Astra Linux, что и видно на скриншоте.

Совет

Если вместо отображения тестовой страницы, показанной на рис. 34.8, браузер предложит вам сохранить файл *test.php*, перезапустите веб-сервер (см. *разд. 34.5*).

Как вы уже догадались, каталог /var/www/html является корневым для нашего сервера, и если создать в нем файл test.html, то он будет доступен по адресу: **http://localhost/test.html**.

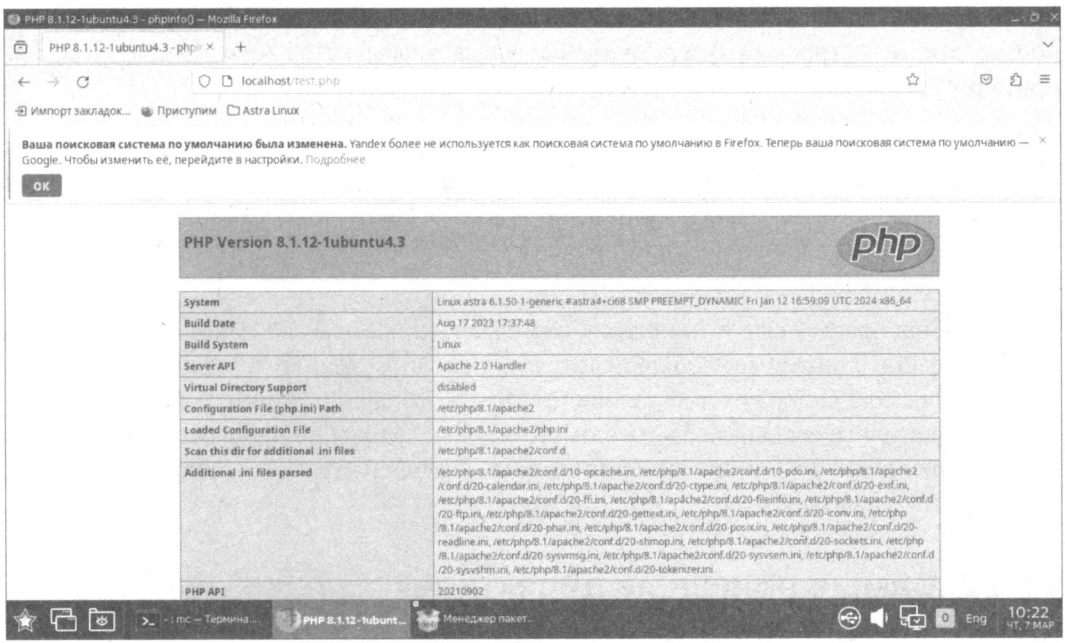

Рис. 34.8. Информация о веб-сервере и PHP

34.4. Файл конфигурации веб-сервера

34.4.1. Базовая настройка

Конфигурационные файлы Apache находятся в каталоге /etc/apache2. Основной конфигурационный файл веб-сервера называется apache2.conf.

Внимание!

После каждого изменения конфигурационных файлов сервера его нужно перезапустить (см. *разд. 34.5*)!

В предыдущих версиях Linux все настройки хранились в одном огромном файле конфигурации. Сейчас этот файл чаще содержит Include-инструкции подключения других файлов конфигурации (более компактных). Все это сделано для удобства администраторов — проще работать с несколькими компактными файлами, чем с одним огромным, поэтому смотрите на все такие дополнительные файлы как на части основного файла конфигурации. Синтаксис у них такой же.

В современных дистрибутивах Linux файл /etc/apache2/apache2.conf содержит общие настройки веб-сервера. А вот настройки, специфичные для сайта, хранятся в каталоге /etc/apache2/sites-available. По умолчанию в нем находится файл 00-default.conf — там и хранятся параметры сайта по умолчанию.

Итак, прежде всего откройте файл 00-default.conf и найдите директиву:

```
#ServerName new.host.name
```

Ее следует раскомментировать и указать в ней имя сервера, которое будут задавать пользователи в строке браузера. Обычно здесь указывается имя компьютера, например:

```
ServerName user-desktop
```

После этого можно будет обращаться к серверу по адресу: **http://user-desktop/**. Имя сервера должно быть зарегистрировано либо в DNS-сервере вашей сети, либо в файле /etc/hosts каждого компьютера сети — чтобы система знала, какой IP-адрес соответствует имени `user-desktop`.

Кроме файла apache2.conf, в подкаталогах каталога /etc/apache2 находятся дополнительные файлы, которые подключаются в основном файле конфигурации. Так, в файле ports.conf содержится описание портов, которые должен прослушивать сервер, в подкаталоге conf-enabled — вспомогательные файлы конфигурации (и все они с «расширением» conf), а описание различных модулей Apache находится в файлах из подкаталога mods-enabled.

34.4.2. Самые полезные директивы файла конфигурации

Понятно, что для полноценной настройки сервера одной директивы `ServerName` недостаточно. В табл. 34.1 приведены самые полезные директивы файла конфигурации Apache. Нужно отметить, что в нее не включены некоторые директивы (например, `Port`, `BindAddress`), которые не используются во второй версии Apache.

***Таблица 34.1.** Директивы файла конфигурации*

Директива	Описание
`ServerName` *имя*	Задает имя веб-сервера — оно должно быть зарегистрировано на DNS-сервере, т. е. обычно — это доменное имя сервера
`ServerAdmin` *e-mail*	Задает e-mail администратора сервера
`ServerRoot` *каталог*	Определяет каталог с конфигурационными файлами сервера
`PidFile` *файл*	Определяет имя файла, в котором будет храниться PID исходного процесса веб-сервера. Обычно изменять эту директиву не нужно
`DocumentRoot` *каталог*	Позволяет задать каталог, в котором хранятся документы веб-сервера, — это корневой каталог документов. Обычно это /var/www

Таблица 34.1 *(окончание)*

Директива	Описание
`StartServers N`, `MaxSpareServers N`, `MinSpareServers N`, `MaxClients N`	Директивы, непосредственно влияющие на производительность сервера. Мы их рассмотрим отдельно в *разд. 34.6*
`KeepAlive On | Off`, `KeepAliveTimeout N`	Управляют постоянными соединениями (будут рассмотрены в *разд. 34.6*)
`DirectoryIndex` *список*	Задает имена файлов, которые могут использоваться в качестве главной страницы (индекса). Значение по умолчанию `index.html index.cgi index.pl index.php index.xhtml`
`HostnameLookups On|Off`	Если директива включена (`On`), то IP-адрес клиента перед записью в журнал будет разрешен (т. е. веб-сервер вычислит доменное имя клиента перед записью информации о попытке доступа в журнал). Выключение (`Off`) этой опции позволяет повысить производительность сервера, поскольку ему не нужно будет тратить время на разрешение IP-адресов в доменные имена
`ErrorLog` *файл*	Задает журнал ошибок
`TransferLog` *файл*	Задает журнал обращений к серверу
`Timeout N`	Тайм-аут в секундах (время, на протяжении которого сервер будет ждать возобновления прерванной попытки передачи данных)
`User` *пользователь* `Group` *группа*	Директивы `User` и `Group` задают имя пользователя и группы, от имени которых запускается веб-сервер
`FancyIndexing on | off`	Если пользователь в запросе не укажет имя документа, а только каталог, но в нем не окажется главной страницы, заданной директивой `DirectoryIndex`, сервер передаст пользователю оглавление каталога. Эта директива определяет, в каком виде будет передано оглавление каталога: в более красивом, со значками каталогов и описаниями файлов (значение `On`), или в более простом (`Off`)
`AddIcon` *картинка список*	Если `FancyIndexing` включена, то `AddIcon` позволяет связать графическую картинку с типом файла, например: `AddIcon /images/graphics.gif .gif, .jpeg, .bmp, .png, .tiff`
`DefaultIcon` *картинка*	Позволяет задать картинку по умолчанию (`AddIcon`, `FancyIndexing`)
`ErrorDocument N` *файл*	Позволяет задать файл, содержащий сообщение об ошибке. Для ошибки с номером 404, например: `ErrorDocument 404 /errors/file_not_found.html`
`Directory`, `Limit`, `Location`, `Files`	Это так называемые *блочные* директивы, которые нельзя описать одной строкой, поэтому о них мы поговорим отдельно (см. *разд. 34.4.3*)

34.4.3. Директивы *Directory, Limit, Location, Files*

Рассмотрим сначала блочные директивы `Directory` и `Limit`.

- С помощью блочной директивы `Directory` можно установить параметры отдельного каталога. Внутри директивы `Directory` могут использоваться директивы `AllowOverride`, `Limit`, `Options`. Вот пример определения параметров корневого сервера:

```
<Directory />
AllowOverride None
Options None
</Directory>
```

Значения `None` для обеих директив (`AllowOverride` и `Options`) считаются самыми безопасными. `None` для `AllowOverride` запрещает использование файлов .htaccess, которые могут переопределять директивы конфигурационного файла Apache. К тому же `AllowOverride None` позволяет повысить производительность сервера.

Допустимые опции каталога (значения директивы `Options`) приведены в табл. 34.2.

***Таблица 34.2.** Опции каталога*

Опция	Описание
`None`	Запрещены все опции
`All`	Все опции разрешены
`Indexes`	Если указана эта опция, при отсутствии файла, заданного `DirectoryIndex`, будет выведено оглавление каталога. Если `Options` установлена в `None` (или `Indexes` не указана в списке опций), то оглавление каталога выводиться не будет
`Includes`	Разрешает использование SSI (Server Side Includes)
`IncludesNoExec`	Более безопасный режим SSI: разрешает SSI, но запрещает запускать из включений внешние программы
`ExecCGI`	Разрешает выполнение CGI-сценариев
`FollowSymLink`	Разрешает использование символических ссылок. Довольно опасная опция, поэтому лучше ее не использовать

- Блочная директива `Limit` позволяет ограничить доступ. Внутри этой директивы можно использовать директивы `order`, `deny` и `allow` (вообще-то, есть еще и директива `require`, но она применяется очень редко). Директива `order` задает порядок выполнения директив `deny` и `allow`:

```
# сначала запретить, потом разрешить
order deny, allow
# сначала разрешить, потом запретить
order allow, deny
```

Директивы `allow` и `deny` нужно использовать так:

```
# запрещаем доступ всем
deny from all
# разрешаем доступ только нашей сети
allow from firma.ru
```

Пример использования директив `Directory` и `Limit` представлен в листинге 34.2.

Листинг 34.2. Фрагмент файла конфигурации Apache

```
<Directory />
AllowOverride None
Options None
<Limit>
       order deny, allow
       # запрещаем доступ всем
       deny from all
       # разрешаем доступ только нашей сети
       allow from firma.ru
</Limit>

</Directory>
```

В качестве параметра директиве `Limit` можно передать метод передачи данных (`GET`, `POST`), например:

```
<Limit GET>
<Limit POST>
```

Теперь обратимся к блочным директивам `Location` и `Files`.

- Директива `Location` очень похожа на директиву `Directory`. Только если `Directory` ограничивает доступ к каталогу, то `Location` предназначена для ограничения доступа к отдельным адресам (URL) сервера:

  ```
  <Location URL>
  директивы ограничения доступа
  </Location>
  ```

  К директивам ограничения доступа относятся `order`, `deny`, `allow`.

- Директива `Files` предназначена для ограничения доступа к отдельным файлам:

  ```
  <Files файл>
  директивы ограничения доступа
  </Files>
  ```

  Вы можете указать как отдельный файл, так и регулярное выражение, которому должны соответствовать файлы:

```
# запрещаем доступ к файлу privat.html всем, кроме нашей сети
<Files privat.html>
order deny, allow
deny from all
allow from firma.ru
</Files>

# запрещаем доступ к файлам .ht* всем
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
</Files>
```

Мы рассмотрели все самые полезные директивы конфигурационного файла Apache. Напомню, что директивы, непосредственно влияющие на производительность сервера, рассмотрены в *разд. 34.6*.

34.4.4. Работа сервера на нескольких портах

Как правило, веб-сервер «слушает» порт 80. Но в некоторых сложных конфигурациях нужно, чтобы он «слушал» несколько портов. Далее будет показано, как разместить на одном сервере три независимых приложения, каждое из которых будет работать на собственном порту: 80, 3128 и 8080.

Сначала нужно отредактировать файл /etc/apache2/ports.conf — задать в нем порты, которые будет слушать сервер (по одному порту в строке):

```
Listen 80
Listen 3128
Listen 8080
```

Затем создать конфигурацию для каждого приложения (для каждого порта). Вы можете использовать разные файлы конфигурации, а можете поместить все настройки в один файл — как вам больше нравится. В листинге 34.3 настройки хранятся в одном большом файле etc/apache2/sites-available/000-default.conf (чтобы не делать три разных листинга).

Листинг 34.3. Файл /etc/apache2/sites-available/000-default.conf

```
<VirtualHost *:80>
    DocumentRoot    /var/www/landing

    <Directory /var/www/landing>
        Options -Indexes
        AllowOverride All

        # Apache 2.4.x
        <IfModule mod_authz_core.c>
            Require all granted
        </IfModule>
    </Directory>
```

```
    CustomLog   /var/log/apache2/landing.access.log "Combined"
    ErrorLog    /var/log/apache2/landing.error.log
</VirtualHost>

<VirtualHost *:3128>
    DocumentRoot    /var/www/sonerezh

    <Directory /var/www/sonerezh>
        Options -Indexes
        AllowOverride All

        # Apache 2.4.x
        <IfModule mod_authz_core.c>
            Require all granted
        </IfModule>
    </Directory>

    CustomLog   /var/log/apache2/demo.sonerezh.bzh-access.log "Combined"
    ErrorLog    /var/log/apache2/demo.sonerezh.bzh-error.log
</VirtualHost>

<VirtualHost *:8080>
    DocumentRoot    /var/www/sonerezh-ru

    <Directory /var/www/sonerezh-ru>
        Options -Indexes
        AllowOverride All

        # Apache 2.4.x
        <IfModule mod_authz_core.c>
            Require all granted
        </IfModule>
    </Directory>

    CustomLog   /var/log/apache2/demo.sonerezh.ru-access.log "Combined"
    ErrorLog    /var/log/apache2/demo.sonerezh.ru-error.log
</VirtualHost>
```

Принцип здесь следующий: мы указываем номер порта в директиве `VirtualHost`, а затем готовим разную конфигурацию для разных узлов.

34.5. Управление запуском сервера Apache

Для управления веб-сервером можно использовать команду `systemctl`:

- `sudo systemctl start apache2` — запуск сервера;
- `sudo systemctl stop apache2` — останов сервера;
- `sudo systemctl restart apache2` — перезапуск сервера.

Понятно, что веб-сервер запускается автоматически, поэтому каждый день вам не придется вводить команду sudo `systemctl start apache2`.

34.6. Оптимизация Apache

В конфигурационном файле сервера Apache apache2.conf, находящемся в каталоге /etc/apache2, имеется ряд директив, позволяющих оптимизировать работу сервера:

- директива `MaxClients` позволяет ограничить число одновременно работающих клиентов.

  Чтобы правильно установить это значение, нужно знать, сколько пользователей может одновременно зайти на сервер. При небольшой посещаемости вполне хватит значения 30–50, при большой загрузке количество одновременно работающих клиентов может исчисляться сотнями. Следите за посещаемостью вашего сервера и корректируйте это значение, иначе какая-то часть пользователей может остаться «за бортом», а им это очень не понравится (или же, наоборот, ресурсы сервера будут использоваться нерационально);

- директива `StartServers` задает количество экземпляров сервера, которые будут созданы при запуске исходной копии сервера.

  Для этой директивы можно установить значение, равное 10% от `MaxClients`. Устанавливать большое значение не следует во избежание нерационального использования ресурсов компьютера;

  Рассмотрим обычную ситуацию. Для `MaxClients` вы установили значение 200, а для `StartServers` — 20. Запросы первых 20 клиентов будут обрабатываться очень быстро, поскольку сервисы уже запущены. Запрос 21-го клиента будет обслужен чуть медленнее, поскольку понадобится запустить еще одну копию Apache. И тем не менее не нужно устанавливать в нашем случае (`MaxClients = 200`) для `StartServers` значение больше 20 — ведь не всегда даже 20 человек одновременно заходят на сервер. Если же на сервере постоянно находится как минимум 20 человек, тогда нужно увеличить значения и `MaxClients`, и `StartServers`.

  Впрочем, бывают и исключения — например, если сервер обслуживает внутреннюю корпоративную сеть. В этом случае вы точно знаете, сколько клиентов в вашей сети, а следовательно, можете точно определить, какое значение установить для `MaxClients` и `StartServers`. Но все равно для `MaxClients` нужно установить чуть большее значение, чем для `StartServers`, — на всякий случай:

```
MaxClients 150
StartServers 100
```

- чтобы еще эффективнее оптимизировать работу веб-сервера, нужно понимать, как он работает: клиент посылает запрос, веб-сервер его обрабатывает и посылает клиенту ответ. После этого соединение можно закрывать и завершать копию Apache, обслуживающую это соединение. Но зачем завершать копию веб-сер-

вера, если сейчас же на сайт зайдет другой пользователь, и опять нужно будет запускать еще одну копию сервера, что только увеличит загрузку процессора.

Поэтому с помощью директивы `MaxSpareServers` можно установить максимальное число серверов, которые останутся в памяти уже после закрытия соединения с пользователем, — они будут просто ждать своего пользователя. Теоретически, чтобы сбалансировать нагрузку, значение для `MaxSpareServers` можно установить таким же, как и для `StartServers`, т. е. 10% от `MaxClients`;

- вы не задумывались, что если веб-сервер будет работать в режиме постоянного соединения, то это повысит его производительность? Если вы об этом подумали, то мыслите в правильном направлении. Представим, что у нас на сайте есть форум. Человек редко заходит на форум, чтобы посмотреть одну страничку, — обычно он может находиться на форуме часами. Так зачем же закрывать соединение? Чтобы потом опять тратить время и ресурсы на его открытие?

  Разрешить постоянные соединения можно с помощью директивы `KeepAlive`. Она задает максимальное число таких соединений:

  ```
  KeepAlive 5
  ```

- а директива `KeepAliveTimeout` задает тайм-аут для постоянного соединения в секундах:

  ```
  KeepAliveTimeout 15
  ```

Используя все упомянутые в этом разделе директивы, вы сможете добиться существенного повышения производительности своего веб-сервера.

ASTRA LINUX®

ГЛАВА 35

Настройка SSH-сервера

- Протокол SSH
- Использование SSH-клиента
- Настройка SSH-сервера
- Вход по ключу без пароля

35.1. Протокол SSH

Протокол SSH (Secure Shell) — это самый популярный протокол удаленного доступа. В отличие от того же telnet, который использовался еще со времени первых UNIX-версий, все, что передается по протоколу SSH, — передается в зашифрованном виде.

SSH использует для шифрования передаваемых данных следующие алгоритмы: Blowfish, 3DES (Data Encryption Standard), IDEA (International Data Encryption Algorithm) и RSA (Rivest-Shamir-Adelman algorithm). Самыми надежными из них считаются IDEA и RSA. Поэтому, если вы передаете действительно конфиденциальные данные, лучше использовать один из этих алгоритмов.

В состав любого дистрибутива Linux входят SSH-сервер и SSH-клиент. В Astra Linux нужные пакеты называются openssh-client (клиент для подключения по протоколу SSH, установлен по умолчанию) и openssh-server (сервер, нужно установить при необходимости).

Используя apt или Synaptic (рис. 35.1), установите пакет openssh-server:

```
sudo apt install openssh-server
```

35.2. Использование SSH-клиента

SSH-клиент установлен по умолчанию практически в каждом дистрибутиве Linux. В Windows можно использовать любые клиенты вроде Bitvise SSH Client, PuTTY и др.

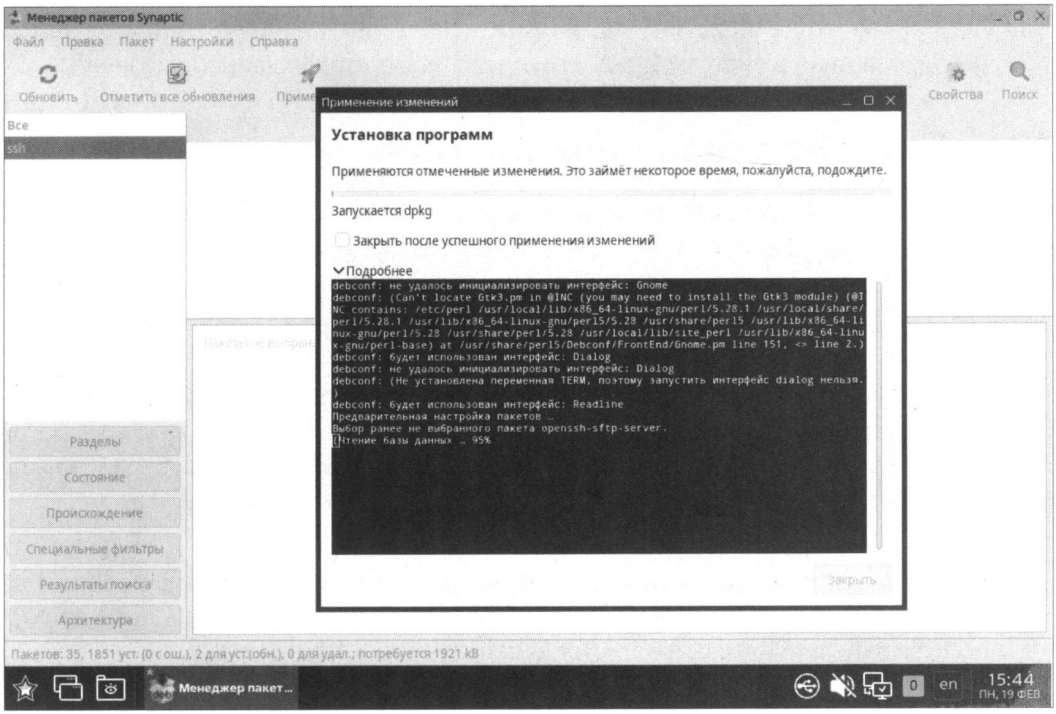

Рис. 35.1. Установка пакета openssh-server с помощью Synaptic

Синтаксис вызова команды `ssh` выглядит так:

`ssh [`*опции*`] <`*адрес_удаленного_компьютера*`>`

Как правило, опции программы можно не указывать. Самая часто используемая опция — это `-p <`*порт*`>`, задающая номер порта, если он отличается от стандартного (22).

35.3. Настройка SSH-сервера

По сути, настройка SSH-сервера сводится к установке пакета openssh-server и запуску службы. Сервер сразу же готов к использованию, однако настоятельно рекомендуется исправить файл /etc/ssh/sshd_config. В нем обратите внимание на директивы `Port` и `PermitRootLogin`. Рекомендуемые значения:

```
Port 8022
PermitRootLogin no
```

Первая устанавливает нестандартный номер порта. Вы, конечно же, можете использовать другой номер порта. Вторая отключает возможность входа как root — из соображений безопасности.

После редактирования файла конфигурации перезапустите сервис:

```
sudo systemctl restart ssh
```

Для удаленного входа в систему вы можете использовать любую учетную запись, зарегистрированную в системе (кроме root и тех, для которых вход отключен).

35.4. Вход по ключу без пароля

Существует практика аутентификации по ключу (Public Key Authentication), а не по паролю. Ее преимущества и недостатки мы рассматривать не станем, так как здесь — одни преимущества: пароль никто не подберет, так как его нет. Это самый надежный способ аутентификации. Жаль, что на практике (из личного опыта) используется он реже обычной аутентификации по паролю.

Принцип аутентификации по ключу следующий: создается пара ключей (открытый и закрытый ключи). Закрытый ключ хранится на стороне клиента, и в идеале он недоступен ни для кого. Открытый — загружается на сервер, к которому нужно получить доступ.

В идеальном мире ключи должны генерировать сами пользователи. Сгенерировать ключи можно с помощью команды `ssh-keygen`, а также задействовав функциональность самого SSH-клиента. Например, возможность создания ключевой пары есть в Bitvise SSH Client и во многих других клиентах (рис. 35.2 и 35.3).

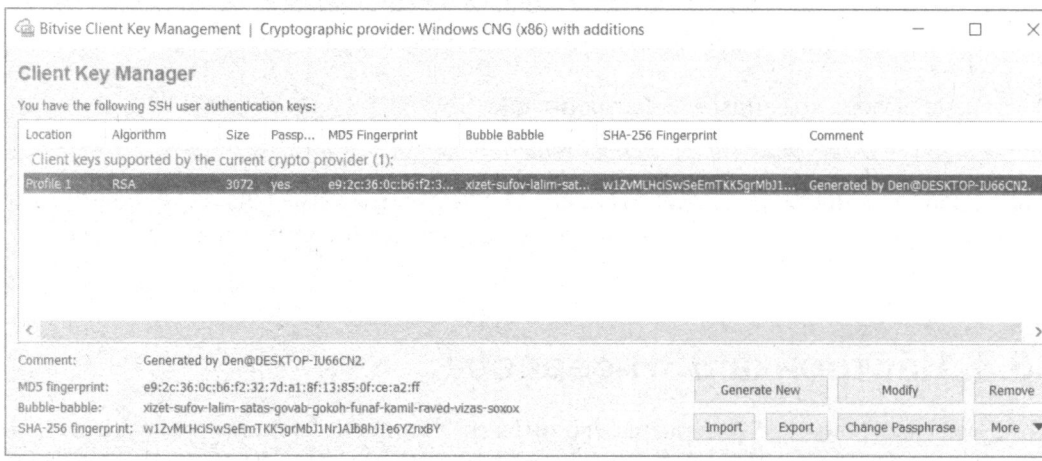

Рис. 35.2. Генерирование ключей средствами SSH-клиента

Если у вас Linux, никакой дополнительный софт устанавливать не требуется. Просто введите команду:

```
$ mkdir ~/.ssh; ssh-keygen -t rsa -b 2048 -f ~/.ssh/rsa
```

Здесь мы генерируем RSA-ключи длиной 2048 битов. После генерации ключей публичный ключ вы найдете в файле с именем ~/.ssh/rsa.pub. Этот файл надо передать администратору на сервер. Как это сделать — выбирайте сами: можно отправить по e-mail, можно загрузить командой `scp` прямо на сервер:

```
$ scp ~/.ssh/rsa.pub  user@example.com:~
```

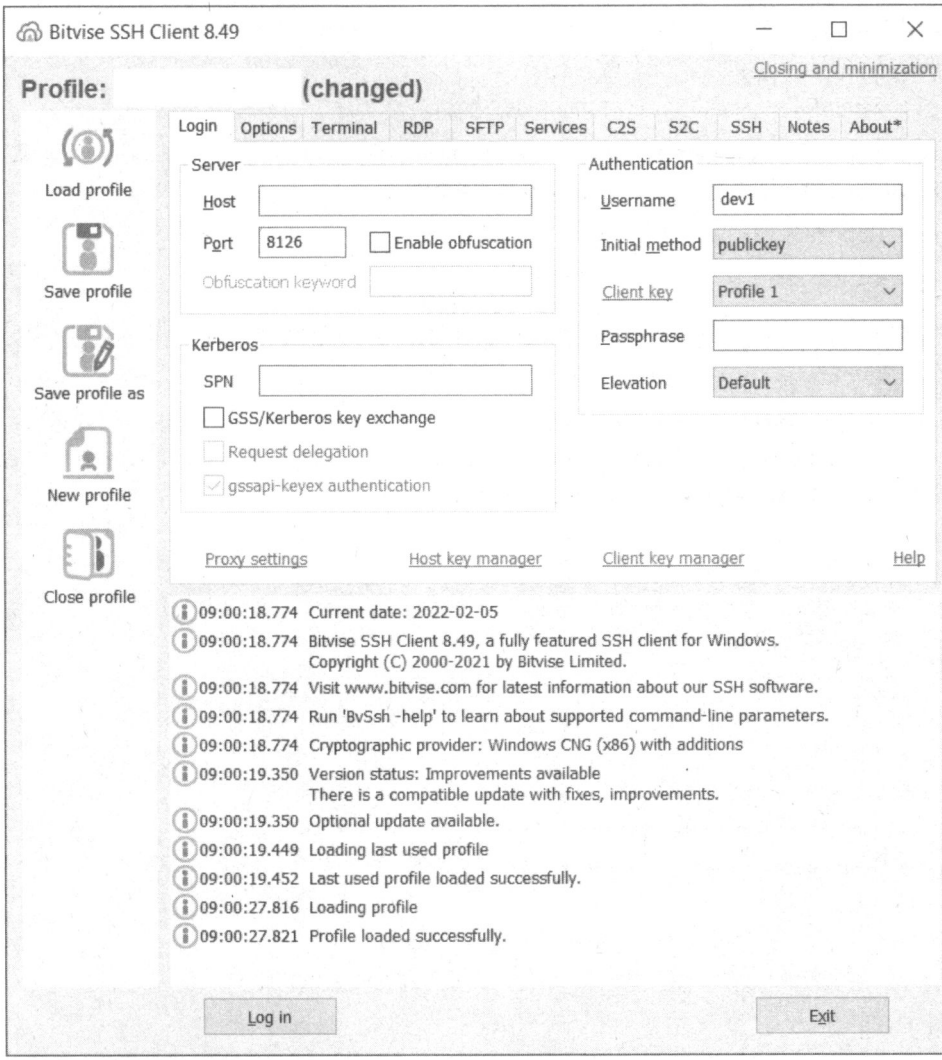

Рис. 35.3. Сгенерировав ключ, нужно выбрать **publickey** в качестве метода аутентификации

Эта команда загрузит открытый ключ в домашний каталог пользователя `user` — но только в том случае, если администратор еще не успел выключить аутентификацию по паролю. Скорее всего, администратор уже запретил входить по SSH посредством паролей, поэтому просто отправьте ему ключ на e-mail.

Далее ключом должен заниматься администратор. Ему нужно поместить содержимое этого файла в файл ~/.ssh/authorized_keys того пользователя, вход по ключу для которого настраивается, то есть в нашем случае ~ будет означать /home/<*имя*>.

Если вы сами себе администратор, то прямо сейчас запретите вход по паролю и перезапустите SSH. Для этого откройте в любом текстовом редакторе файл /etc/ssh/sshd_config и установите в значение `No` для директивы `PasswordAuthentication`:

```
PasswordAuthentication no
```

Также нужно убедиться, что две следующие директивы установлены так:

```
PubkeyAuthentication yes
AuthorizedKeysFile %h/.ssh/authorized_keys
```

Первая включает аутентификацию по публичному ключу, а вторая — указывает имя файла, в котором должны храниться ключи.

Осталось перезагрузить SSH:

```
sudo systemctl ssh restart
```

На стороне клиента войти по ключу на SSH-сервер можно так:

```
ssh -i ~/.ssh/rsa.pub user@example.com
```

Здесь нужно указать имя файла ключа, имя пользователя и имя сервера — это в Linux. А в Windows вам придется настраивать используемый SSH-клиент — соответствующие инструкции вы найдете в Сети.

ASTRA LINUX®

ГЛАВА 36

Удаленное VNC-подключение

- Что такое VNC?
- Подготовка сервера
- Подключение к удаленному рабочему столу

36.1. Что такое VNC?

VNC (Virtual Network Computing) — система удаленного доступа к рабочему столу компьютера. В предыдущей главе мы рассматривали SSH-подключение. Как вы могли заметить — это был удаленный консольный вход, в этой же главе мы рассмотрим графический вход в удаленную систему.

Пользователям Windows знаком протокол удаленного доступа к рабочему столу (RDP, Remote Desktop Protocol), и они знают, как его использовать. Можно сказать, что VNC является аналогом RDP, но для Linux. На самом деле VNC и RDP имеют мало чего общего, за исключением концепции, — мы можем подключиться к удаленному компьютеру и увидеть его графический рабочий стол.

Изначально система VNC была разработана исследовательской лабораторией Olivetti & Oracle Research Lab, которая в то время принадлежала Olivetti и Oracle Corporation. В 1999 году эта лаборатория была приобретена компанией AT&T, которая в 2002 году закрыла разработку VNC.

Технология хоть и стара, но работает. VNC состоит из сервера и клиента. Сервер — это программа, предоставляющая доступ к экрану компьютера, на котором она запущена, а клиент — это программа, получающая изображение экрана сервера и отправляющая ему команды по протоколу RFB.

Протокол RFB (Remote Framebuffer) — это простой клиент-серверный сетевой протокол прикладного уровня для удаленного доступа к рабочему столу компьютера. В подробности протокола мы углубляться не станем. Для нас, как для администратора, важно знать два момента: первый — это номера портов, второй — безопасность протокола.

Поговорим сначала о портах. RFB по умолчанию использует диапазон TCP-портов от 5900 до 5906 — каждый порт представляет собой соответствующий экран X-сервера, то есть порты 5900–5906 соответствуют экранам :0–:6. Экраном по умолчанию является экран :0. Кстати, RFB/VNC можно использовать и для Windows — там также экран по умолчанию имеет номер :0.

Теперь о безопасности. VNC не использует шифрование трафика, что, конечно же, печально. Единственная защита — это DES-шифрование пароля, то есть пароль не передается в открытом виде, а шифруется алгоритмом DES с длиной ключа 56 битов. Во многих реализациях также существует ограничение на длину пароля в 8 символов, а если длина превышает 8 символов, пароль урезается, а лишние символы игнорируются. Такая длина пароля, а также небольшой размер ключа делают этот протокол ненадежным с точки зрения безопасности. Поэтому было бы неплохо на сервере развернуть OpenVPN-сервер для шифрования трафика. В этом случае вы сначала подключаетесь к VPN, а затем к VNC.

Если вам не с руки этим заниматься, можно использовать реализации VNC, поддерживающие шифрование, — например, EchoVNC, UltraVNC, RealVNC (коммерческая версия).

Далее мы рассмотрим стандартную реализацию VNC, входящую в состав Astra Linux. При этом мы не станем отвлекаться на шифрование соединения, однако если вы планируете предоставлять VNC-доступ компьютерам со всего мира, то вам все же придется настроить VPN/PPTP-туннель, иначе его отсутствие ничем хорошим не закончится.

36.2. Подготовка сервера

Чтобы к компьютеру можно было подключиться по VNC, необходимо установить и запустить специальное ПО — сервер VNC, который в Astra Linux содержится в пакете vino (рис. 36.1):

```
sudo apt install vino
```

Установив vino, запустите команду `vino-preferences` для задания опций подключения. В частности, нужно включить параметр **Требовать от пользователя ввести следующий пароль** и задать пароль для доступа к удаленному рабочему столу (рис. 36.2).

Запустить сервер можно командой:

```
/usr/lib/vino/vino-server
```

Да, сервер запускается командой, а не через systemd. Все-таки не забываем о почтенном возрасте протокола — тогда systemd еще не было. Протокол запуска показан на рис. 36.3. Как можно здесь видеть, протокол работает на порту 5900.

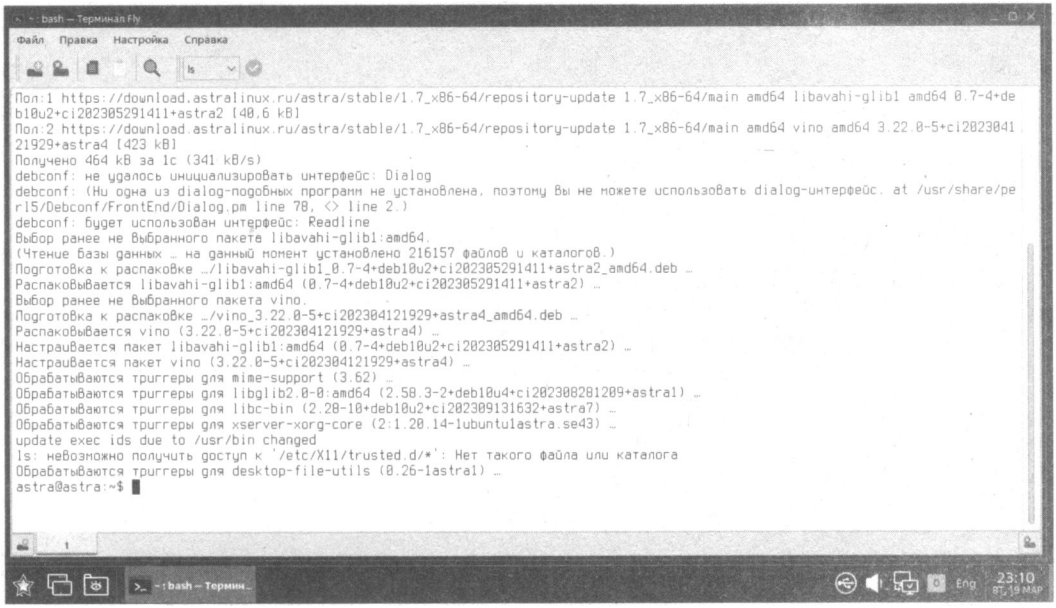

Рис. 36.1. Установка пакета vino

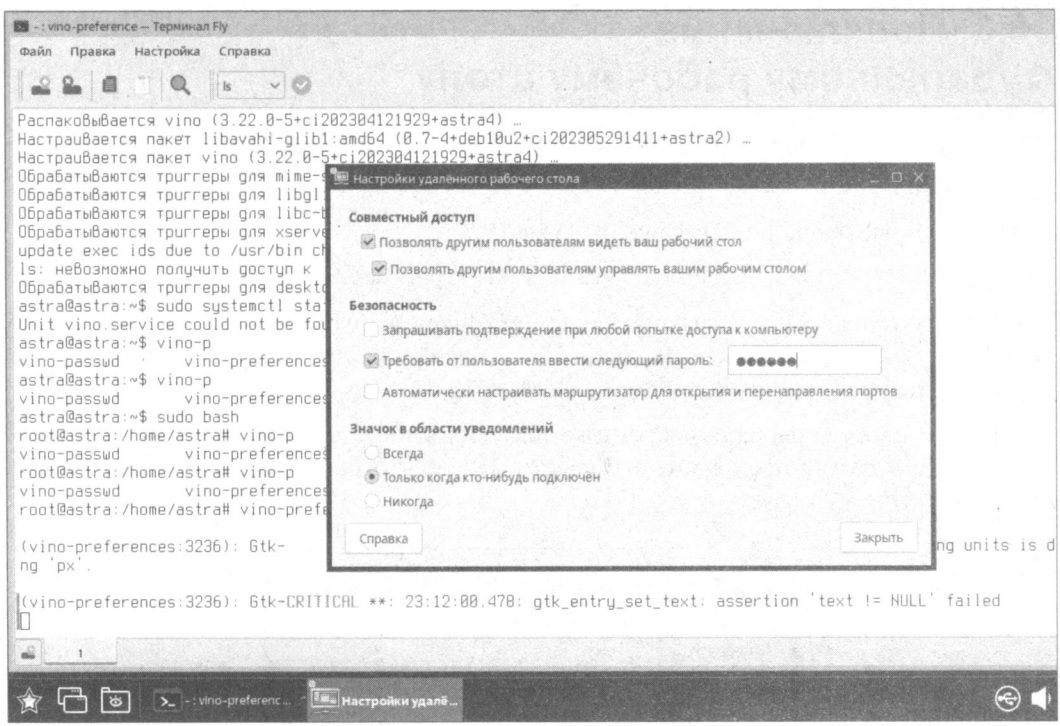

Рис. 36.2. Параметры доступа к удаленному рабочему столу

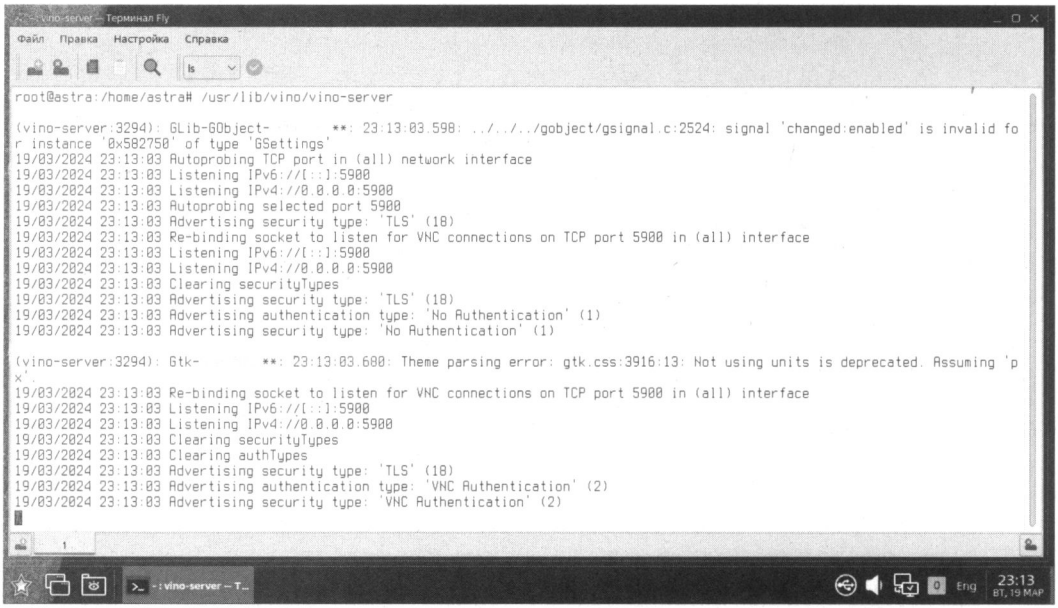

Рис. 36.3. Сервер VNC запущен

36.3. Подключение к удаленному рабочему столу

На клиентском компьютере нужно установить пакет tigervnc-viewer:

```
sudo apt install tigervnc-viewer
```

А для установки подключения ввести команду:

```
vncviewer -ViewOnly=0 -DotWhenNoCursor=1 IP_адрес_хоста
```

В общем-то, можно не указывать какие-либо параметры, а просто ввести команду `vncviewer`. Тогда в графическом окне вы сможете указать все эти параметры, используя интерфейс пользователя (рис. 36.4).

Введите IP-адрес сервера и нажмите кнопку **Подключ.** Если соединение защищено паролем, программа предложит его ввести. После этого вы получите доступ к удаленному рабочему столу.

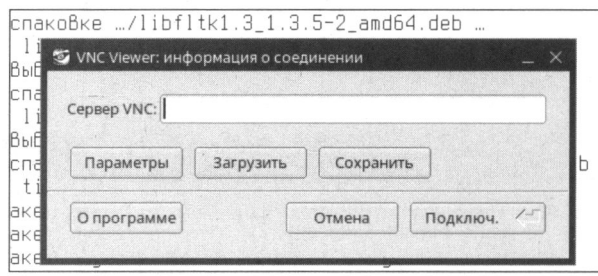

Рис. 36.4. VNC-клиент

Предметный указатель

А

Б

В

Г

Д

З

И

К

Л

М

Н

О

П

P

C

T

У

Ф

Ц

Ч

Ш

Администрирование Astra Linux

Евгений Андреев, инженер-программист и системный администратор. Имеет более чем двадцатилетний опыт работы с операционной системой Linux в самых различных вариантах — от встраиваемых систем до серверов.

Отечественный дистрибутив Astra Linux широко внедряется в последнее время с целью замены Microsoft Windows. Книга ориентирована на системного администратора, который впервые сталкивается с Astra Linux. Она позволит начать работу с Astra Linux даже тем администраторам, которые никогда ранее вообще не работали ни с одним другим дистрибутивом Linux. В книге рассмотрены установка Astra Linux, настройка после установки, процесс загрузки системы, в том числе система инициализации systemd, основы командной строки, настройка сети, беспроводного и проводного соединения с Интернетом, настройка хранилища, работа с файловой системой, а также настройка самых необходимых сетевых служб — Samba, Apache (веб-сервер), SSH. Сделан акцент на специфические особенности Astra Linux, а именно: мандатный контроль целостности, мандатное управление доступом, системный киоск, служба каталогов ALD и учет сменных накопителей.

КАТЕГОРИЯ: операционные системы

ISBN 978-5-9775-1993-9

191036, Санкт-Петербург,
Гончарная ул., 20
Тел.: (812) 717-10-50,
339-54-17, 339-54-28
E-mail: mail@bhv.ru
Internet: www.bhv.ru