Моей семье.
Боб Корецкий

Содержание

От издательства 10

Предисловие 12

Глава 1. «Быстрый старт» системного администратора Raspberry Pi OS 16

1.0 Цели 16
1.1 Введение 16
1.2 Команды обслуживания файлов и справка по использованию команд Raspberry Pi OS 18
1.2.1 Структура файлов и каталогов 18
1.2.2 Управление содержимым файлов 20
1.2.3 Управление файлами 21
1.2.4 Создание, удаление и управление каталогами 26
1.2.5 Получение помощи с помощью команды man 31
1.2.6 Другие способы вызова справки 34
1.3 Служебные команды 35
1.3.1 Проверка настроек системы 35
1.4 Команды печати 37
1.5 Итоги 39

Глава 2. Основы администрирования Raspberry Pi OS 41

2.0 Цели 41
2.1 Введение 42
2.1.1 Простое системное администрирование 44
2.2 Установка Raspberry Pi OS на различные носители и предварительная настройка системы 44
2.2.1 Загрузка Raspberry Pi Imager и установка 64-битной версии Raspberry Pi OS на карту microSD 45
2.2.2 Установка настольной операционной системы Raspberry Pi на старый ПК с архитектурой x86 46
2.2.3 Как загрузить и запустить Raspberry Pi OS с SSD, подключенного через USB3 47
2.2.4 Как установить Ubuntu Desktop на оборудование Raspberry Pi 49
2.3 Рекомендации и варианты до и после установки 50
2.3.1 Рекомендации перед установкой 51
2.3.2 Варианты действий после установки 54
2.4 Администрирование системных служб, процедуры запуска и завершения работы 60
2.4.1 Процессы загрузки и запуска 60

2.4.2 Подсистема systemd и традиционная перезагрузка или завершение работы системы 62
2.4.3 Предварительные соображения при управлении системными службами с помощью systemd 63
2.4.4 Дополнительные ссылки по управлению системными службами с помощью systemd 64
2.5 Администрирование пользователей 64
2.5.1 Добавление пользователя и группы в текстовом интерфейсе 66
2.5.2 Добавление и ведение групп в текстовом интерфейсе 69
2.5.3 Изменение и удаление учетной записи пользователя и группы из командной строки 71
2.5.4 Метод создания пользователей и групп на втором носителе данных 72
2.6 Базовое управление паролями 75
2.7 Определение и изменение прав доступа к файлам 77
2.7.1 Как раскрыть права доступа к файлам 77
2.7.2 Изменение прав доступа к файлам 79
2.7.3 Права доступа к каталогам 84
2.8 Файловые системы, подключение к постоянным носителям и добавление носителей в вашу систему 85
2.8.1 Типы файловой системы и ext4 88
2.8.2 Постоянные носители и устройства 89
2.8.3 Предварительные соображения при добавлении новых носителей 91
2.8.4 Пять быстрых и простых способов узнать имена логических устройств носителя 92
2.8.5 Добавление новых носителей в систему 95
2.8.6 Добавление дисков с помощью fdisk 98
2.9 Установка ZFS и синтаксис команд zpool и zfs 101
2.9.1 Установка ZFS в системе Ubuntu на оборудовании Raspberry Pi 102
2.9.2 Синтаксис команд zpool и zfs 102
2.9.3 Терминология ZFS 103
2.9.4 Как работает ZFS 105
2.9.5 Важные концепции ZFS 105
2.9.6 Базовый пример ZFS 106
2.10 Настройка принтера 116
2.10.1 Возможности общей системы печати UNIX (CUPS) 117
2.10.2 Локальное управление CUPS с помощью systemd 117
2.11 Резервное копирование и восстановление файловой системы 119
2.11.1 Стратегический обзор средств резервного копирования файлов 120
2.11.2 Linux GNU tar 121
2.12 Другие средства архивирования и резервного копирования Raspberry Pi OS 127
2.12.1 Команда rsync 127
2.12.2 Файлы сценариев для резервного копирования и восстановления 131
2.12.3 Программное обеспечение для резервного копирования и восстановления: Filezilla, SD Card Copier и git 133

2.13 Обновления программного обеспечения и операционной системы.....137
2.13.1 Предложения по предварительной модели хранения 139
2.13.2 Использование инструмента упаковки Advanced Packaging Tool (APT).....141
2.13.3 Обновление операционной системы.....144
2.14 Мониторинг и настройка производительности системы и программного обеспечения.....145
2.14.1 Управление ресурсами процессов и потоков на уровне приложения.....146
2.14.2 Управление памятью.....154
2.14.3 Оценка использования системного диска.....155
2.14.4 Настройка сети с помощью команды ip.....156
2.15 Безопасность системы.....161
2.15.1 Аутентификация на основе пароля.....163
2.15.2 Модели управления доступом: дискреционная (DAC), принудительная (MAC) и ролевая (RBAC).....164
2.15.3 Команда sudo.....168
2.15.4 Системы обнаружения и предотвращения вторжений.....169
2.15.5 Программное обеспечение безопасности Linux.....171
2.15.6 Безопасность постоянных носителей.....175
2.15.7 Учетные данные процесса.....176
2.15.8 Шифрование диска.....178
2.16 Методологии виртуализации.....179
2.16.1 Приложения виртуализации.....182
2.17 Итоги.....183

Глава 3. Python.....184

3.0 Цели.....184
3.1 Введение.....184
3.2 Быстрый старт в Python 3 с IDE Thonny.....185
3.2.1 Запуск и окно Thonny 186
3.2.2 Создание и запуск простой программы Python из Thonny.....187
3.3 Обзор языка Python 3.....189
3.3.1 Объекты и классы.....191
3.3.2 Модель данных программы Python.....194
3.3.3 Релизы Python и ссылки на ресурсы.....195
3.3.4 Иерархия стандартных типов Python 3.....196
3.3.5 Основные предположения.....198
3.3.6 Запуск Python 3 стандартными способами.....200
3.3.7 Использование Python 3.....204
3.3.8 Информация об установке Python.....205
3.4 Синтаксис Python 3.....209
3.4.1 Ввод текста, комментариев, чисел, групповых операторов и выражений.....209
3.4.2 Переменные и соглашения об именах.....212
3.4.3 Функции.....214
3.4.4 Условное выполнение.....216
3.4.5 Определенные и неопределенные структуры цикла и рекурсия ...218

3.4.6 Загрузка и выгрузка файлов ... 221
3.4.7 Списки и функции списков ... 224
3.4.8 Строки, форматирование строк и операции с последовательностями ... 225
3.4.9 Кортежи ... 231
3.4.10 Множества ... 232
3.4.11 Словари ... 233
3.4.12 Генераторы ... 234
3.4.13 Сопрограммы ... 237
3.4.14 Исключения ... 240
3.4.15 Модули, глобальная и локальная области действия в функциях ... 242
3.5 Практические примеры ... 243
3.5.1 Альтернативный способ написания файлов сценариев оболочки ... 244
3.5.2 Базовый веб-сервер и обслуживание пользовательских файлов ... 249
3.5.3 Графические пользовательские интерфейсы Python 3 с виджетами tkinter ... 256
3.5.4 Многопоточный параллелизм с Python ... 269
3.5.5 Общение между потоками: проблема поставщик–потребитель с использованием модуля очереди ... 277
3.6 Итоги ... 284
3.7 Сокращенный справочный глоссарий ... 284
Приложение 3A Синтаксис Python и сводка команд ... 286

Вопросы, проблемы и проекты ... 291

Предметный указатель ... 309

От издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг, мы будем очень благодарны, если вы сообщите о ней главному редактору по адресу dmkpress@gmail.com. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательство «ДМК Пресс» очень серьезно относится к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты dmkpress@gmail.com.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Этот выпуск, посвященный основам администрирования операционной системы Raspberry Pi OS, представляет собой сборник простых в использовании и необходимых на практике задач администрирования системы Raspberry Pi для начинающих пользователей с особым акцентом на Python и Python 3.

Главной идеей системного администрирования современной системы Linux XXI века, такой как Raspberry Pi OS, является использование подсистемы *systemd* для гарантирования эффективной и действенной работы ядра Linux при обеспечении всех трех краеугольных камней работы современного компьютера: параллелизма систем, виртуализации и безопасной устойчивости. В текст включены упражнения, чтобы закрепить цели обучения читателей с помощью решений и примеров кода, представленных на сопроводительном разделе сайта *GitHub*.

Эта книга предназначена для студентов и практиков, стремящихся максимально эффективно использовать Raspberry Pi OS. Благодаря множеству практических примеров, проектов и упражнений книгу также можно использовать в более формальной среде обучения, чтобы дополнить и расширить базовые знания об операционной системе Linux.

Роберт М. Корецкий – бывший преподаватель Инженерной школы Портлендского университета. Ранее он работал конструктором автомобильной техники в компании Freightliner Corp. в Портленде, штат Орегон. Роберт женат, у него двое детей и двое внуков.

Предисловие

О чем эта книга

Эта книга представляет собой сборник простых в использовании и важных на практике задач по администрированию системы Raspberry Pi для начинающих. Операционная система Raspberry Pi OS является производной от ветки Linux Debian, и на момент написания текста самой последней версией этой операционной системы была Debian Bullseye. Чтобы представить здесь приемы и команды системного администрирования, я выбрал некоторые самые базовые вещи, а также несколько более сложных концепций, приемов, команд и деталей, которые могут и не появиться в более полной книге по системному администрированию.

Главной идеей системного администрирования современной системы Linux XXI века, такой как Raspberry Pi OS, является использование подсистемы *systemd* для гарантирования эффективной и действенной работы ядра Linux при обеспечении всех трех краеугольных камней работы современного компьютера: параллелизма систем, виртуализации и безопасной устойчивости.

И этот контроль над ядром со стороны «суперъядра», которым, по сути, и является *systemd*, должен также обеспечивать высочайший уровень производительности и скорости работы системы, учитывая варианты использования компьютера и предполагаемые потребности целевого сообщества пользователей, которую обслуживает компьютер. Если начинающий пользователь или даже более опытный системный специалист не обладает не только базовыми, но и более полными знаниями о том, как *systemd* контролирует каждый процесс и операцию в современной системе Linux, он никогда не сможет освоить администрирование и управление реализацией той функциональности, которая в конечном итоге может потребоваться в конкретных сценариях использования системы и удовлетворения требований, которые предъявляет к ней данное сообщество пользователей.

Конечно, из всего множества возможных тем, которые мы могли бы представить подробно, изложенные здесь примеры были выбраны в некоторой степени субъективным образом. Этот избирательный подход в основном объясняет следующими соображениями:

а) безопасное обслуживание с точки зрения параллелизма, виртуализации и устойчивости одной системы Raspberry Pi, которую обычный начинающий пользователь может установить на свое собственное выделенное оборудование;

b) важность выбранной темы в практическом рейтинге основных задач системного администрирования;
c) влияние подсистемы *systemd* на режим обслуживания Raspberry Pi OS и оборудования, на котором она установлена, в соотвествии с потребностями обычного пользователя;
d) общая интеграция выбранных тем по системному администрированию друг с другом;
e) насколько хорошо эти темы помогают подготовить студента к поступлению на любую выбранную профессию в области информационных технологий или компьютерных наук, или могут помочь кому-то, уже работающему в этих областях, использовать эту книгу, чтобы соответствовать лучшим практикам своей профессии. Другими словами, для аудитории учащихся и целей непрерывного образования;
f) в некоторой степени сделать возможным экстраполяцию этих тем из одиночной системной среды Raspberry Pi на более широкую и крупномасштабную вычислительную среду, например на серверы малого и среднего размера или облачные виртуальные вычисления.

Как читать и использовать эту книгу

> **Примечание**
>
> Предпосылкой и обязательным условием изучения этой книги является то, что вы понимаете, какова правильная форма или структура команды Linux и как ее вводить в командной строке консоли либо терминала Raspberry Pi.

Все вводимые вручную команды далее выделены жирным моноширинным шрифтом; ответы системы и код сценариев Python приводятся обычным моноширинным шрифтом.

Для примера: общий синтаксис или структура одной команды Linux (часто называемой простой командой), вводимой в командной строке, выглядит следующим образом:

```
$ command [[-] option(s)] [option argument(s)] [command argument(s)]
```

где:

`$` – это обозначение командной строки (приглашение оболочки Raspberry Pi OS); все, что заключено в квадратные скобки [], не всегда необходимо;
`command` – имя допустимой для этой оболочки команды Linux строчными буквами;
`[-option(s)]` – один или несколько модификаторов, изменяющих поведение команды;
`[option argument(s)]` – один или несколько модификаторов, которые изменяют поведение **`[-option(s)]`**;

`[command argument(s)]` – один или несколько объектов, на которые влияет команда.

Обратите внимание на следующие семь основных моментов:

1) команда **`command`**, опции **`option(s)`**, аргументы опций **`option argument(s)`** и аргументы команды **`command argument(s)`** разделяются пробелом, но между несколькими опциями или несколькими аргументами опций пробел не обязателен;
2) порядок опций или аргументов опций не имеет значения;
3) символ пробела между опцией и аргументом опции необязателен;
4) нажатие клавиши <**Enter**> отправляет команду на интерпретацию и выполнение;
5) опциям может предшествовать один или два дефиса, в зависимости от формы опции. Краткой форме опции предшествует один дефис (-), длинной форме опции – два дефиса (--). Между дефисом(ами) и опцией(ями) не должно быть пробелов;
6) небольшой процент команд (например, **`whoami`**) не принимает никаких опций, аргументов опций или аргументов команды;
7) все символы в командной строке чувствительны к регистру!

Кроме того, очень часто можно ввести несколько команд Linux (иногда называемых составными командами, чтобы отличать их от простых команд) в одной командной строке перед нажатием клавиши <**Enter**>. Компоненты нескольких команд Linux разделяются символами перенаправления ввода и вывода (>, <), чтобы направить вывод одной команды на ввод другой.

Итак, основные предпосылки для изучения этой книги:

1) знание того, как вводить синтаксически правильную команду Linux в командной строке;
2) наличие доступа к выделенному компьютеру Raspberry Pi с уже установленной и работающей последней версией операционной системы Raspberry Pi OS;
3) наличие в системе прав привилегированного пользователя, который может выполнять команду **`sudo`**, чтобы получить статус суперпользователя;
4) базовые знания о том, как редактировать и сохранять текстовые файлы в текстовом редакторе *nano*. В этой книге мы не даем инструкций по использованию текстового редактора *nano*, но их можно найти в других книгах[1].

Для этой книги предусмотрен онлайн-раздел сайта *Github* с дополнительными материалами и обновлениями, программным кодом, решениями как для упражнений в главах, так и для задач, вопросов и проектов в конце главы, а также других дополнений. Его можно найти по адресу www.github.com/bobk48/RaspberryPiOS.

---

[1] На русском языке см., например, https://help.ubuntu.ru/wiki/nano. – *Прим. перев.*

Все инструкции в этой книге были протестированы на Raspberry Pi 4B или Raspberry Pi 400, оба с 4 Гбайтами памяти и последней на момент написания версией Raspberry Pi OS.

Порядок изучения книги

- Просмотрите оглавление.
- Выберите тему, которая вас интересует.
- Проделайте примеры или все образцы командной строки, представленные по этой теме.
- Возможно, выберите другую тему, которая вас интересует, и разместите там примеры и все образцы командной строки.
- Наконец, вернитесь к началу книги. Делайте все, от начала до конца.
- Проверьте и повторите вышеописанное при необходимости.
- Наслаждатесь!

1

«Быстрый старт» системного администратора Raspberry Pi OS

В этой вводной главе мы рассмотрим основные команды Raspberry Pi OS, которые позволяют системному администратору выполнять обслуживание файлов и другие полезные операции. Это обязательный набор основ, которые необходимо знать даже обычному пользователю без прав администратора для эффективной работы с символьным (текстовым) интерфейсом операционной системы. После прочтения данной главы читателю должно быть очевидно, что правильно развернутые текстовые команды являются основным средством, которое системный администратор имеет в своем распоряжении для поддержания целостности системы. Здесь мы приводим набор основных примеров и показываем базовый формат основных команд и примитивов.

1.0 Цели

1. Объяснить, как управлять и обслуживать файлы и каталоги.
2. Показать, где можно получить общесистемную справку по командам Raspberry Pi OS.
3. Продемонстрировать использование набора служебных команд для начинающих.

   Охват основных команд и операторов:

```
cat cd cp exit hostname -I ip login lp lpr ls man mesg mkdir more mv passwd PATH pwd rm
rmdir telnet unalias uname whatis whereis who whoami
```

1.1 Введение

Чтобы начать продуктивно работать с системным администрированием в Raspberry Pi OS, новичку необходимо ознакомиться со следующими последовательными темами:

1) как сохранять и организовывать файлы в файловой структуре операционной системы. Создание древовидной структуры папок (также на-

зываемых каталогами) и логическое хранение файлов в этих папках имеет решающее значение для эффективной работы в Raspberry Pi OS;

2) как получить справку по текстовым командам и их использованию. Для эффективной работы с клавиатуры в текстовом пользовательском интерфейсе (Character User Interface, CUI) на основе командной строки необходима возможность быстро узнавать, как использовать команду, ее параметры и аргументы, правильно набрать ее на клавиатуре;
3) как выполнить небольшой набор основных служебных команд для установки или настройки вашей рабочей среды. Как только новичок освоит правильный способ создания команд обслуживания файлов, добавление набора служебных команд сделает каждый сеанс более продуктивным.

Чтобы успешно использовать эту главу в качестве трамплина к остальной части книги, вам следует внимательно прочитать и последовательно выполнить инструкции и сессии командной строки, которые мы предоставляем, в указанном порядке. Каждый раздел этой главы, а также две последующие главы основаны на предшествующей информации. Они дадут вам концепции, командные инструменты и методы, которые позволят вам выполнять системное администрирование с помощью Raspberry Pi OS.

На протяжении этой книги мы иллюстрируем материал с использованием следующей версии Raspberry Pi OS на следующем оборудовании:

- **система**: raspberrypi Kernel: 6.1.21-v8+ aarch64 bits: 64 compiler: gcc v: 10.2.1 Console: tty0 Distro: Debian GNU/Linux 11 (bullseye);
- **тип процессора**: ARM Device System: Raspberry Pi 400 Rev 1.0.

В этой главе основные команды, которые мы хотим проиллюстрировать, сначала определяются с помощью сокращенного описания синтаксиса, которое поясняет общие компоненты этих команд. Порядок описания синтаксиса следующий:

- **синтаксис** (Syntax): точный синтаксис того, как команда, ее параметры и аргументы правильно вводятся в командной строке;
- **цель** (Purpose): конкретная цель команды;
- **результат** (Output): краткое описание результатов выполнения команды;
- **часто используемые параметры/опции** (Commonly used options/features): список наиболее популярных и полезных параметров и их аргументов.

Для получения дополнительной информации: следующая веб-ссылка ведет на сайт, который позволит вам ввести одну или несколько команд Raspberry Pi OS и получить подробное объяснение компонентов этой команды: https://explainshell.com/.

Упражнения к этой главе

1. Введите следующие команды в командной строке вашей системы Raspberry Pi и запишите результаты. Какие из них синтаксически неверны? Почему? (Приглашение

среды Bash отображается в виде символа **$** в каждом случае, и мы предполагаем, что **file1** и **file2** существуют.)

```
$ la -ls
$ cat
$ more -q file1
$ more file2
$ time
$ lsblk-a
```

2. Как отличить команду Raspberry Pi OS от ее параметров, аргументов опций и аргументов команды?
3. В чем разница между одной командой Raspberry Pi OS и несколькими командами Raspberry Pi OS, введенными в командной строке перед нажатием <**Enter**>?
4. Если после ввода команды Raspberry Pi OS вы не получаете сообщения об ошибке, как узнать, что она действительно выполнила то, что вы хотели?

1.2 Команды обслуживания файлов и справка по использованию команд Raspberry Pi OS

После первого входа в новую Raspberry Pi OS одним из ваших первых действий будет создание и организация рабочей среды и файлов, которые будут в ней содержаться. Обслуживание файлов и состоит в операциях организации файлов в соответствии с некоторой логической схемой. Логическая схема, используемая для организации ваших файлов, может заключаться в создании ячеек для хранения файлов в соответствии с тематикой содержимого файлов или датами их создания. В следующих разделах вы будете вводить команды создания и обслуживания файлов, которые создают структуру, аналогичную той, что показана на рис. 1.1. Выполните операции в следующих разделах в том порядке, в котором они представлены, чтобы получить лучшее представление о том, что на самом деле значит обслуживание файлов. Кроме того, очень важно просмотреть то, что было представлено в предисловии относительно структуры команды Raspberry Pi OS, чтобы, когда вы начнете вводить команды для обслуживания файлов, вы понимали, насколько синтаксис того, что вы вводите, соответствует общему синтаксису любой команды Raspberry Pi OS.

1.2.1 Структура файлов и каталогов

Когда вы впервые открываете окно терминала или консоли, вы оказываетесь в домашнем каталоге или папке автономного пользователя, связанного с именем пользователя и паролем, которые использовались для входа в систему. В каком бы каталоге вы ни находились в данный момент, он называ-

ется *текущим рабочим каталогом*, причем в любой момент времени активен только один текущий рабочий каталог. Полезно визуализировать структуру ваших файлов и каталогов с помощью блок-схем. На рис. 1.1 показан пример домашнего каталога и файловой структуры пользователя с именем *bob*. На этом рисунке каталоги представлены в виде параллелограммов, а простые файлы (например, файлы, содержащие текстовые или двоичные инструкции) представлены в виде прямоугольников. Путь (path) — это просто текстовый способ обозначения местоположения каталога или файла в полной файловой структуре системы Raspberry Pi, над которой вы работаете.

Например, путь к файлу *myfile2* на рис. 1.1 — */home/bob/myfile2*. Обозначение пути начинается с корня (/) всей файловой системы, спускается до папки с именем *home*, а затем спускается дальше к домашнему каталогу пользователя с именем *bob*.

Как показано на рис. 1.1, файлы с именами *myfile*, *myfile2* и переименованный файл *renamed_file* хранятся в каталоге *bob*. Под *bob* находится подкаталог с именем *first*. В следующих разделах вы создадите эти файлы и структуру подкаталогов в домашнем каталоге пользователя, под которым вы вошли в систему Raspberry Pi.

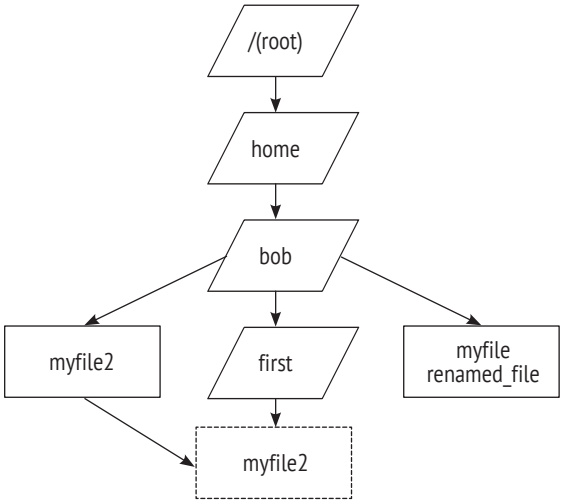

Рис. 1.1 Пример структуры каталогов

Упражнение по теме

5. Введите следующие две команды в вашей Raspberry Pi OS:

```
$ cd /
$ ls
```

   По аналогии с рис. 1.1 нарисуйте диаграмму каталогов и файлов, имена которых вы видите в выводе второй команды. Сохраните эту диаграмму для дальнейшего использования.

1.2.2 Управление содержимым файлов

Чтобы начать работу с файлами, вы можете легко создать новый текстовый файл с помощью команды **cat**. Синтаксис команды **cat** следующий:

```
cat [options] [file-list]
```

Назначение

Последовательно объединить один или несколько файлов или отобразить их в окне консоли.

Результат

Содержимое файлов в списке файлов **file-list** отображается на экране по одному файлу за раз.

Часто используемые параметры/опции

+E – отображать $ в конце каждой строки.
-n – показать номера строк в отображаемых файлах.
--help[1] – отобразить цель команды и краткое объяснение каждой опции.

Команда **cat** (сокращение от concatenate, *связывать*) позволяет объединять файлы. В примере далее вы соедините то, что набираете на клавиатуре, с новым файлом, создаваемым в текущем рабочем каталоге. Это достигается с помощью символа перенаправления **>**, который принимает то, что вы вводите со стандартного ввода (в данном случае с клавиатуры), и направляет его в файл с именем *myfile*. Вы можете рассматривать клавиатуру и поток информации, которую она предоставляет, как файл. Как указано ранее, такое использование является примером команды **cat** без параметров, т. е. опций или аргументов команды. Он просто использует команду, символ перенаправления и цель или место назначения с именем **myfile**, куда будет идти перенаправление.

Это самый простой пример ввода нескольких команд в командной строке, в отличие от одной команды, как говорилось в начале главы. В составе нескольких команд вы можете объединить отдельные команды Raspberry Pi OS в цепочку с соединяющими операторами, например показанным здесь символом перенаправления.

```
$ cat > myfile
This is an example of how to use the cat command to add plain text to a file
<Ctrl+D>
$
```

Вы можете набирать столько строк текста, нажимая <**Enter**> на клавиатуре, чтобы различать строки в файле, сколько захотите. Затем в новой строке, когда вы нажмете <**Ctrl+D**>, файл создается в текущем рабочем каталоге с помощью введенной вами команды. Вы можете просмотреть содержимое

[1] Напомним, что двойной дефис (--) перед опцией вместо одинарного употребляется в случае, если опция представлена в полной (несокращенной) форме. Это правило для Linux общепринятое, но соблюдается не всегда. – *Прим. перев.*

этого файла, поскольку это обычный текстовый файл, созданный с помощью клавиатуры, выполнив следующие действия:

```
$ more myfile
This is an example of how to use the cat command to add plain text to a file
$
```

Это простой пример синтаксиса одной команды Raspberry Pi OS.

Общий синтаксис команды **more** следующий:

```
more [options] [file-list]
```

Назначение

Объединить/отобразить файлы в списке файлов на экране по одному экрану за раз.

Результат

Содержимое файлов в списке файлов отображается на экране по одной странице за раз.

Часто используемые параметры/опции

+E/str – вставить две строки перед первой строкой, содержащей **str**.
-nN – отобразить N строк на экране/странице.
+N – начать отображение содержимого файла со строки номер N.

Команда **more** по умолчанию показывает один полный экран с содержимым файла за раз. Если файл состоит из нескольких страниц, перейти к просмотру последующих страниц можно, нажав клавишу пробела на клавиатуре; нажатие клавиши <**Q**> прекращает просмотр вывода.

Упражнение по теме

6. Используйте команду **cat**, чтобы создать еще один текстовый файл с именем *testfile*. Затем объедините содержимое *myfile* и *testfile* в один текстовый файл с именем *myfile3* с помощью команды **cat**.

1.2.3 Управление файлами

Чтобы скопировать содержимое одного файла в другой, используйте команду **cp**. Общий синтаксис команды следующий:

```
cp [options] file1 file2
```

Назначение

Скопировать *file1* в *file2*; если *file2* – каталог, сделать копию *file1* в этом каталоге.

Результат

Скопированный файл.

Часто используемые параметры/опции

-i – если пункт назначения существует, запросить запрос перед перезаписью.

-p – сохранять режимы доступа к файлам и время изменения скопированных файлов.

-r – рекурсивно копировать файлы и подкаталоги[1].

Например, чтобы создать точную копию файла с именем **myfile** с новым именем **myfile2**, введите следующее:

```
$ cp myfile myfile2
$
```

Чтобы изменить имя файла или каталога, вы можете использовать команду **mv**. Общий синтаксис команды следующий:

```
mv [options] file1 file2
mv [options] file-list directory
```

Назначение

Первый вариант – переименовать *file1* в *file2*.

Второй вариант – перенести все файлы в списке **file-list** в каталог *directory*.

Результат

Переименованные или перенесенные файлы.

Часто используемые параметры/опции

-f – принудительное перемещение независимо от режимов доступа к файлу назначения.

-i – запрашивать у пользователя перед перезаписью существующего файла в месте назначения.

В следующем примере первым аргументом команды **mv** является исходное имя файла *myfile2*, вторым аргументом – вновь назначенное *renamed_file*:

```
$ mv myfile2 renamed_file
$
```

На этом этапе важно обратить внимание на использование пробелов в командах Raspberry Pi OS. Что, если вы получите файл из системы Windows, в одном из имен которого есть один или несколько пробелов? Как можно работать с этим файлом в Raspberry Pi OS? Ответ простой: всякий раз, когда

---

[1] В программировании рекурсия – вызов функции (процедуры) из нее же самой. В данном случае опроса вложенных файлов/каталогов некая функция А по очереди опрашивает объекты, находящиеся в указанном каталоге, и выполняет заданное действие. Обнаружив, что очередной опрашиваемый объект также является каталогом, функция А вызывает саму себя уже с этим обнаруженным каталогом в качестве аргумента, и процесс повторяется по отношению ко вложенным в этот каталог объектам, до тех пор, пока имеется хоть один вложенный объект нижележащего уровня, также являющийся каталогом. Рекурсия применяется там, где количество уровней вложенности не определено заранее, так как не требует предварительного подсчета уровней и организации вложенных функций с раздуванием объема кода. Хороший пример рекурсии см. также в разделе 2.7.2. – *Прим. перев.*

вам нужно использовать подобное имя файла в команде в качестве аргумента, заключите имя файла в двойные простые кавычки ("").

Например, вы можете получить по электронной почте файл от кого-то из системы Windows под названием *latest revisions october.txt*. Чтобы работать с этим файлом в Raspberry Pi OS, то есть использовать имя файла в качестве аргумента в какой-либо команде, заключите все имя в двойные кавычки. Правильная команда для переименования этого файла во что-то более короткое выглядит так:

```
$ mv "latest revisions october.txt" laterevs.txt
$
```

Чтобы удалить файл, вы можете использовать команду **rm**. Общий синтаксис команды следующий:

```
rm [options] file-list
```

Назначение

Удаление файлов в списке **file-list** из файловой структуры (и с диска[1]).

Результат

Удаление файлов.

Часто используемые параметры/опции

- **-f** – удалить независимо от режима доступа к файлам.
- **-i** – запрашивать у пользователя перед удалением файла.
- **-r** – рекурсивно удалить файлы в списке **file-list**, если список **file-list** является каталогом; использовать с осторожностью!

Например, чтобы удалить файл *renamed_file* из текущего рабочего каталога, введите:

```
$ rm renamed_file
$
```

Упражнение по теме

7. Используйте команду **rm**, чтобы удалить файлы *testfile* и *myfile3*.

---

[1] Команда удаления **rm** в Linux, как и аналогичные команды в других операционных системах, конечно, не удаляет файлы с диска полностью – она всего лишь делает их недоступными для файловой системы. Записанная на диск информация, составляющая содержание файла, остается на своем месте – оно просто помечается как «свободное», то есть разрешенное для новых записей. Старая информация будет действительно уничтожена только после перезаписи на то же место. Фрагменты содержимого могут задерживаться на диске довольно долго, особенно если не производится интенсивных операций записи и перезаписи, и эти фрагменты можно восстановить, хотя и довольно сложным путем (процедура восстановления для Unix и Linux подробно описана здесь: https://www.osp.ru/lan/2002/12/135557). Для гарантированного уничтожения «стертой» информации с дисков и флешек необходимо использовать специальные утилиты, принудительно заполняющие нулями (или любыми случайными символами) все области диска, помеченные как «свободные». – *Прим. перев.*

Самая важная команда для обслуживания файлов – это команда **ls**. Общий синтаксис команды следующий:

```
ls [options] [pathname-list]
```

Назначение

Выводит имена файлов и каталогов в каталоге, указанном в списке **pathname-list**, на экран дисплея.

Результат

Имена файлов и каталогов в каталоге, указанном в списке **pathname-list**, или только имена, если список **pathname-list** содержит лишь имена файлов.

Часто используемые параметры/опции

- **-F** – отображать косую черту (/) после имен каталогов, звездочку (*) после двоичных исполняемых файлов и символ «at» (@) после символических ссылок.
- **-a** – отображать имена всех файлов, включая скрытые файлы.
- **-i** – отображать номера индексных дескрипторов.
- **-l** – отобразить длинный список, включающий режимы доступа к файлу, количество ссылок, владельца, группу, размер файла (в байтах) и время модификации.

Команда **ls** выведет список имен файлов или папок в вашем текущем рабочем каталоге либо папке. Кроме того, как и в случае с другими командами, которые мы использовали до сих пор, если вы включите в команду полную спецификацию пути для аргумента **pathname-list**, то сможете перечислить имена файлов и папок по этому пути. Например, чтобы увидеть имена файлов в вашем текущем рабочем каталоге, введите команду без параметров и получите следующее:

```
$ ls
Desktop Documents Downloads Dropbox Music Pictures Public Templates
Videos
$
```

Обратите внимание, что вы, вероятно, не получите список тех же самых имен файлов, показанных выше. В примере, который мы использовали, система автоматически поместила некоторые файлы в ваш домашний каталог, а мы еще создали свои с именами *myfile*. и *myfile2*. Также обратите внимание, что список имен файлов не будет включать имя *renamed_file*, поскольку ранее мы удалили этот файл.

Следующая команда, которую вы выполните, представляет собой альтернативный (модифицированный) способ выполнения команды **ls**, включающий имя команды и параметры. Как указывалось в предисловии, команда Raspberry Pi OS может иметь опции, которые можно ввести в командной строке вместе с самой командой, чтобы изменить ее поведение. В случае команды **ls** опции **l** и **a** создают более длинный список всех обычных и системных (обозначенных точкой перед именем) файлов, а также предоставляют другую сопутствующую информацию о файлах.