ОГЛАВЛЕНИЕ

Модуль 1. Знакомство с РЕД ОС.

История создания GNU/Linux.

Операционная система GNU/Linux относится к классу UNIX – подобных операционных систем, наследуя от UNIX множество черт.

Операционная система UNIX была создана в Bell Lab фирмы AT&T в 1970 г.

Поэтому время во всех Unix подобных системах отсчитывается с даты первого релиза: 01- 01-1970

Примечание: Схему развития UNIX подобных систем можно посмотреть тут: https://upload.wikimedia.org/wikipedia/commons/7/77/Unix_history-simple.svg

Для тех лет это была одна из самых передовых операционных систем, обеспечивающих многозадачность и возможность одновременной работы многих пользователей.

В настоящее время имеются две основные ветви UNIX систем: UNIX System V (продолжение разработок AT&T) и BSD (Berkeley Software Distribution ранее разрабатывалась в университете Berkeley).

Примечание: То есть все современные UNIX системы можно отнести к первой или второй ветви. Так операционная система SUN Solaris 9 является представителем ветви UNIX System V (кратко SVR4 – System V release 4), а FreeBSD 5.1 – наследницей BSD.

Операционная система GNU/Linux не является прямой наследницей какой-либо из этих двух ветвей UNIX систем. Она сочетает в себе черты, присущие обеим ветвям, поскольку ее разрабатывает множество людей, имеющих, естественно различные предпочтения.

В отличие от, например, команды разработчиков FreeBSD, разработку GNU/Linux в целом, как единой операционной системы, никто не координирует. Поэтому имеется множество различных наборов программного обеспечения (дистрибутивов), являющихся, несмотря на совершенную несхожесть друг с другом, GNU/Linux.

Название Linux является зарегистрированной торговой маркой Линуса Бенедикта Торвальдса.

GNU – наименование проекта GNU is not UNIX (рекурсивный акроним, который можно расшифровывать бесконечно), начатого в 1984 г. в FSF Ричардом Столлмэном (FSF – Free Software Foundation).

Линус Торвальдс, будучи в 1991 г. студентом, экспериментировал с операционной системой MINIX (она была разработана профессором Эндрью Танненбаумом в учебных целях) и ассемблером процессора i386.

25 августа 1991 г. Линус Торвальдс распространил в группе новостей comp.os.minix usenet сообщение о том, что он разработал на основе MINIX новую

операционную систему для AT совместимых компьютеров, и пригласил всех заинтересованных лиц участвовать в продолжении ее разработки.

Примечание: В своем сообщении он специально указал, что его эксперименты не более, чем хобби, и он "... не является таким профессионалом, как специалисты из GNU ...". Действительно, детище Линуса так и осталось бы, вероятно, только экспериментом, но его сообщением заинтересовался Ричард Столлмэн – основатель FSF. К тому моменту в FSF уже семь лет проводились работы в рамках проекта GNU, целью которого было создание свободно распространяемой по лицензии GPL Copy Left UNIX System V совместимой операционной системы (Суть лицензии GPL Copy Left будет пояснена ниже).

В FSF к 1991 г. было создано огромное количество широко используемого программного обеспечения. Например: оболочка Bash (Bourne again shell), знаменитый редактор Emacs, компилятор gcc и прочее.

Однако для реализации проекта GNU не хватало стабильно работающего ядра операционной системы.

Исходно проект GNU был ориентирован на ядро HURD, но работа по созданию этого ядра до сих пор далека от завершения, поэтому тандем: ядро от Линуса Торвальдса плюс утилиты, библиотеки, компилятор, оболочка и прочее от GNU, был отличной альтернативой GNU/HURD.

В течение 1991 – 1992 гг. проект GNU/Linux активно развивался и в 1993 г. появились первые дистрибутивы GNU/Linux: Slackware, Debian и Red Hat.

Примечание: Первые дистрибутивы вряд ли можно было рекомендовать для промышленного использования, но массовый интерес к новой операционной системе как со стороны широких масс разработчиков, так и со стороны производителей аппаратного обеспечения. И к середине 199... - х гг. появились вполне стабильные и надежные дистрибутивы, в которых поставлялось большое количество портированного в GNU/Linux программного обеспечения (например, SMTP агент Sendmail и DNS сервер BIND).

Залогами успеха GNU/Linux явились два аспекта: бесплатность и свобода распространения программного обеспечения дистрибутивов, и достаточная для промышленного использования стабильность множества серверных приложений. Эти две особенности GNU/Linux позволяли строить "малобюджетные" серверы для небольших и средних приложений.

С переносом на GNU/Linux офисных приложений дело обстояло существенно хуже вплоть до конца девяностых годов. Однако на сегодняшний момент имеются отличные графические оболочки для Xfree86: GNOME и KDE, обилие оконных менеджеров, средства офисной работы, редакторы и электронные таблицы.

Примечание: В настоящее время уже есть прецеденты массового перевода на GNU/Linux компьютерных систем крупных фирм и даже муниципальных структур очень больших городов, что доказывает зрелость операционной системы GNU/Linux.

Некоторые из наиболее популярных дистрибутивов GNU/Linux представлены ниже:

Red Hat – наиболее массовый универсальный дистрибутив простой в установке и настройке.

Fedora — проект компании RedHat. В данном дистрибутиве Red Hat

отрабатывает все свои инновации.

CentOS — дистрибутив сообщества построенный на базе Red Hat Enterprise Linux, но без проприентарного программного обеспечения.

Debian – профессиональный дистрибутив с очень большим количеством доступных программных пакетов. Команда разработки этого дистрибутива гарантирует свободу его распространения.

SuSE – платно распространяемый дистрибутив от Novell. Отличается энциклопедической подборкой программного обеспечения и высокой надежностью и удобством.

Gentoo – проект, который по методам установки программного обеспечения тяготеет к FreeBSD. Все программные пакеты в нем необходимо компилировать, что позволяет безгранично широко настраивать систему в соответствии с собственными потребностями.

Ubuntu – один из самых популярных дистрибутивов. Проект поддерживается компанией “Canonical” и является ответвлением от проекта Debian.

История создания РЕД ОС

Компания «РЕД СОФТ» в 2014 году приступила к разработке дистрибутива РЕД ОС.

На август 2023 года выпущено 3 версии РЕД ОС: 7.1, 7.2 и 7.3. Версии РЕД ОС нумеруются двумя цифрами: **A.B**, где **A** - мажорная версия, **B** - минорная версия. РЕД ОС имеет 2 редакции: **стандартную** и **сертифицированную**. Обе редакции в рамках одной версии имеют общую пакетную базу. Сертифицированная редакция РЕД ОС проходит регулярные сертификационные испытания в системе сертификации ФСТЭК России.

Каждая версия и редакция РЕД ОС имеет свой репозиторий пакетов. В период поддержки версии РЕД ОС в репозитории публикуются обновления безопасности и обновления функционала.

Сроки поддержки версий РЕД ОС:

Каждая версия РЕД ОС поддерживается минимум 3 года. При окончании поддержки:

- прекращается выпуск обновлений безопасности и обновлений функционала;
- установочный образ остается доступен на сайте РЕД ОС в архиве, репозиторий остается открытым;
- услуги технической поддержки оказываются только в рамках поддерживаемых версий РЕД ОС.

Стандартная редакция

В период поддержки версии РЕД ОС стандартной редакции, начиная с версии 7.3, выпускаются **корректирующие релизы**, которые нумеруются тремя цифрами: **A.B.x**, где **x** - номер корректирующего релиза в рамках версии **A.B**. Корректирующий релиз представляет собой установочный образ, который включает в себя обновления, накопленные с момента предыдущего релиза. На август 2023 года актуальным корректирующим релизом для версии 7.3 является релиз 7.3.3.

Сертифицированная редакция

Сертификат ФСТЭК России №4060 от 12.01.2019 распространяет свое действие на **единственную версию РЕД ОС**. На август 2023 года сертификат №4060 распространяется на РЕД ОС версии 7.3. Однако с момента переоформления сертификата на новую версию РЕД ОС продолжают выпускаться обновления безопасности в рамках общего срока поддержки для предыдущей версии РЕД ОС.

Что означает свобода распространения программного обеспечения?

Чаще всего программное обеспечение, распространяемое в рамках свободных лицензий таких, как GPL или BSD, доступно совершенно бесплатно.

Свобода программного обеспечения вовсе не подразумевает обязательную бесплатность.

Примечание: Некоторые дистрибутивы Linux предоставляются за деньги, как, например, SUSE.

Основная идея свободы программного обеспечения заключается в свободе его модификации и использования. По классическому определению FSF программное обеспечение является свободным если:

1. Программа может выполняться с любой целью без ограничения сферы применения (свобода 0).
2. Имеется возможность изучения и модификации исходного кода программного обеспечения в соответствии со своими потребностями (свобода 1).
3. Разрешается свободно распространять копии программного обеспечения (свобода 2).
4. Возможно улучшать программное обеспечение и публиковать улучшения для всеобщего блага (свобода 3).

Примечание: Следует отметить, что программное обеспечение, доступное в рамках GPL, не является общественной собственностью (Public Domain). Программа является собственностью ее авторов.

Устройство и функции операционной системы GNU/Linux.

Операционная система является комплексом программных средств, предоставляющим для пользовательских приложений программный интерфейс с аппаратным обеспечением.

Будучи многозадачной и многопользовательской операционной системой, GNU/Linux выполняет следующие функции:

1. Взаимодействует с пользователями системы, позволяя им выполнять их задачи посредством пользовательского интерфейса (например, командной строки оболочки).
2. Обеспечивает управление ресурсами компьютера (например, оперативной памятью), обеспечивая одновременное выполнение в компьютерной системе различных пользовательских и системных процессов.
3. Управляет заданиями в системе и предоставляет возможность обмена информацией между различными процессами.
4. Предоставляет возможности длительного хранения информации с помощью файловой системы.
5. Управляет операциями ввода/вывода.
6. Управляет аппаратным обеспечением.
7. Производит мониторинг состояния системы.
8. Обеспечивает взаимодействие системы с другими вычислительными системами посредством сетевых протоколов.
9. Предоставляет пользователям системы возможность использования системных библиотек для создания программ.
10. Обеспечивает разграничение уровней и прав доступа к данным для различных пользователей системы.

Операционная система GNU/Linux включает в себя следующие программные компоненты:

1. Ядро.
2. Системные библиотеки.
3. Системные утилиты и команды.
4. Системы программирования и отладки программ.
5. Командные оболочки.

Примечание: Большинство известных дистрибутивов GNU/Linux включает в себя помимо перечисленных выше программных компонент огромное количество прикладных библиотек и программ.

Ядро операционной системы GNU/Linux является главной частью операционной системы.

Ядро ответственно за базовый интерфейс с аппаратным обеспечением компьютера, содержит в себе:

1. драйверы файловых систем
2. поддержку файловых систем
3. блок управления процессами, распределяя процессорное время и оперативную память среди процессов в системе (планирование процессов).
4. системные вызовы

Процессы ядра (системные процессы), как и другие процессы, с некоторой периодичностью выполняется процессором.

Однако, в отличие от всех остальных процессов, ядро выполняется в привилегированном режиме работы процессора в адресном пространстве памяти, отличном от адресного пространства остальных процессов.

Поэтому вводятся понятия пространства ядра (kernel space) и пользовательского пространства (user space).

Системные процессы, выполняющиеся в пространстве ядра, плотно взаимодействуют с аппаратным обеспечением и определяют жизнедеятельность системы в целом.

Нестабильная работа любого процесса пространства ядра приводит к неработоспособности всей системы. В то же время, сбой или неправильная работа процесса пользовательского пространства обычно не может привести к неработоспособности всей системы.

Приложения выполняются в пользовательском пространстве и не могут получить доступ к аппаратному обеспечению, а также адресному пространству ядра или других процессов.

Эти процессы не могут выполнять многие инструкции процессора, ну и, естественно, инструкцию перехода в привилегированный режим.

Ядро, в свою очередь, может обращаться к адресному пространству любого процесса, так как в привилегированном режиме работы процессора доступны специальные инструкции для управления указателями на ячейки памяти.

Для того, чтобы процессы пользовательского пространства могли обращаться к функциям ядра, например, для обращения к дисковому ресурсу, ядро предоставляет процессам пользовательского пространства *интерфейс системных вызовов*.

Системные вызовы (system calls) стандартизованы в документах POSIX, в которых они описаны как функции на языке C (хотя это вовсе не требует реализовывать системные вызовы именно на C).

Пользовательская программа может обращаться к этим функциям, а ядро операционной системы заботится о выполнении необходимых процедур, соответствующих этим системным вызовам.

Системные утилиты и команды (например, `df` или `ls`) обеспечивают важнейшие инструменты работы пользователя для работы с операционной системой и прикладным программным обеспечением.

Утилиты и команды предоставляют пользователю основные возможности управления и контроля операционной среды.

Примечание: К системным утилитам относятся также программы, реализующие протоколы сетевого взаимодействия, программы предназначенные для отладки и настройки сети, а также многие программы, реализующие графический интерфейс (например, менеджер X сеанса `xdm` из пакета Xfree86).

Системы программирования и отладки программ, а также компиляторы, компоновщики и т.п., например, компилятор `gcc` или отладчик `gdb`, позволяют разрабатывать и отлаживать как новые программы, так и существующие пользовательские или системные программы.

Примечание: Часто для нормального функционирования операционной системы или пользовательского программного обеспечения требуется пересборка каких-либо программ, что и обеспечивается с помощью систем программирования и отладки.

Оболочки, предоставляя пользователям гибкий и удобный интерфейс командной строки, обеспечивают возможность запуска любых программ в системе, а также предоставляют множество возможностей автоматизации рутинных задач с помощью скриптов.

Примечание: Пользовательский интерфейс вовсе не обязательно должен быть реализован с помощью интерфейса командной строки. Графический интерфейс, предоставляемый современными оболочками GNOME и KDE, прост и интуитивно понятен, позволяя даже начинающим пользователям быстро начать эффективную работу с Linux.

Меню загрузки позволяет выбрать какое ядро загрузить или загрузить систему в аварийном режиме.

В окне входа в систему вы можете выбрать пользователя или настроить параметры работы, либо выключить компьютер.

Структура рабочего стола в целом похожа на то, что используется в Windows.

В Linux в домашней папке сосредоточена работа пользователя. В домашнем каталоге:

• Находятся файлы пользователям.

• По умолчанию сохраняются файлы.

• Располагаются различные файлы и/или каталоги индивидуальных настроек пользователя для приложений.

• При входе пользователя происходит автоматический переход в домашний каталог этого пользователя.

Имеются так же и некоторые исключения, например, почтовые ящики пользователей могут располагаться не в домашней папке, но в другом месте. Или, другой пример, запланированные пользователем задания находятся в специальном каталоге, но не домашнем.

Последовательность процесса загрузки.

При включении питания компьютера автоматически запускаются программы, находящиеся в ПЗУ.

Исторически, начиная с первых IBM совместимых компьютерах, использовался BIOS (Basic Input/Output Services – Базовые службы ввода-вывода).

Микросхема ПЗУ в таких компьютерах выполняется по технологии CMOS (Complemenary Metal-Oxyde Semiconductor – Комплементарный метало-оксидный полупроводник).

Примечание: Особенностью этой технологии является то, что программа, сохраняемая в таком ПЗУ может быть настроена без необходимости полного перепрограммирования ПЗУ.

Программа, которая позволяет настраивать поведение BIOS называется CMOS Setup.

В 1990-х Intel разработал новый стандарт для компьютеров с архитектурой Itanium. Изначальное название — Intel Boot Initiative (Загрузочная инициатива Intel), позже было переименовано в EFI.

Основной мотивацией разработки EFI было преодоление ограничений BIOS: 16-битный исполняемый код, адресуемая память 1 Мбайт, проблемы с одновременной инициализацией нескольких устройств, ограничения на размер дисков.

В 2005 году Intel внесла эту спецификацию в UEFI Forum, который теперь ответственен за развитие и продвижение EFI. EFI был переименован в Unified EFI (UEFI).

Одна из программ, входящих в BIOS/UEFI называется POST (Power On Self Test – Проверка самого компьютера при включении питания). Эта программа выполняет начальную проверку конфигурации компьютера и при наличии проблем выдает специальные сигналы (обычно звуковые).

После удачного завершения программы POST запускается другая программа в BIOS, которая называется Bootstrap Loader (аппаратный загрузчик). Она предназначена для обращения к загрузочному устройству, указанному с помощью CMOS Setup в BIOS, поиска на нем загрузочного сектора и, при удачном поиске, загрузке его.

Если загрузка осуществляется с магнитного диска, то программа загрузки находится в первом по порядку секторе. Для жестких магнитных дисков загрузочный раздел еще называют MBR (Master Boot Record – Главная загрузочная запись).

Аппаратный загрузчик опознает программу загрузки в загрузочном разделе по сигнатуре ее последних двух байт (510 и 511 байты начиная с 0 байта – в шестнадцатеричном виде, соответственно, байты 0x1FE и 0x1FF). Сигнатура 16 бит, составленная этими байтами, является числом 0xAA55. Именно по этой сигнатуре опознается загрузочный сектор, который и загружается аппаратным загрузчиком в ОЗУ.

Если программный загрузчик в MBR отсутствует, то его поиск осуществляется в первом секторе раздела, помеченного в таблице разделов жесткого диска, как активный.

Размер секторов на магнитном диске – 512 байт, поэтому программа, находящаяся в загрузочном секторе, проста и не может выполнять сложных действий.

Примечание: В некоторых загрузчиках (не в GNU/Linux!) программный загрузчик в MBR выдает на экран список разделов, с которых возможно продолжение загрузки.

Для преодоления этого ограничения загрузчики операционных систем разбивают на несколько частей.

Загрузчики, чаще всего применяемые в GNU/Linux – LILO и GRUB, состоят из двух частей.

Одна из них, называемая начальным загрузчиком, помещается либо в MBR, либо в первый сектор активного раздела. Задачей начального загрузчика является поиск и загрузка второй части загрузчика.

Вторая часть загрузчика представляют собой более сложную программу, которая может загрузить ядро Linux (или другое ядро) и выполнить необходимые предварительные действия для загрузки операционной системы (например, создание электронного диска с примитивным образом корневой файловой системы).

Загрузка с UEFI реализована несколько иначе.

С GPT-раздела с идентификатором EF00 (ESP, EFI system partition) и файловой системой FAT32 по умолчанию загружается и запускается файл \efi\boot\boot[название архитектуры].efi, например: \efi\boot\bootx64.efi.

Следующий этап либо запуск загрузчика, либо прямая загрузка ядра Linux. 17.Также в большинстве реализаций UEFI возможна загрузка в режиме совместимости с диска с разметкой MBR.

При использовании UEFI возможна также *Secure Boot*, смысл которой

заключается в проверке цифровых подписей ядра и модулей ядра.

Secure Boot одна из причин почему может не загрузиться Linux, например когда вы скомпилируете новые модули ядра самостоятельно, но не подпишите их.

Загрузчики, используемые в GNU/Linux, способны передавать ядру Linux различные конфигурационные параметры и дополнительные команды.

Примечание: В частности, ядру Linux может быть передана команда считать из файловой системы на электронном диске специальный командный файл, который чаще всего называется `linuxrc`. В этом командном файле можно указать список дополнительных действий, которые необходимо выполнить при инициализации системы GNU/Linux.

После загрузки ядра и его инициализации, если в параметрах, переданных ядру, не указано иное, производится запуск программы `/sbin/init` (или в новых ОС GNU/Linux `systemd`).

Ее конфигурационный файл `/etc/inittab` .

Процесс, соответствующий этой программе всегда имеет PID равный 1.

Эта программа осуществляет дальнейшую инициализацию операционной системы GNU/Linux, зависящую от выбранного режима работы (например, однопользовательский или многопользовательский режим).

В Linux и UNIX подобных системах такие режимы работы называются уровнями исполнения (runlevels).

В зависимости от выбранного уровня исполнения команда `/sbin/init` исполняет тот или иной набор специальных командных файлов, называемых инициализационными скриптами.

Именно инициализационные скрипты определяют какие программы будут работать в фоновом режиме на том или ином уровне исполнения, то есть какова будет функциональность системы.

Пример: однопользовательский режим применяется для обслуживания системы и при работе в нем нет необходимости в запуске сетевых приложений. Подробнее вопросы инициализации системы на различных уровнях исполнения будут рассмотрены позже в этой главе.

Одна из программ, которые запускает процесс `/sbin/init` – это программа `getty` или ее разновидность.

Эта программа выводит приглашение войти в сеанс (login:). Если пользователь вводит что-либо в ответ на это приглашение, то эта строка интерпретируется как имя пользователя, которое передается программе login как аргумент.

Далее выводится приглашение ввести пароль (password:), который проверяется в базе данных паролей (например, в `/etc/shadow` – в зависимости от настроек системы).

Если аутентификация производится успешно, то запускается оболочка, указанная для пользователя в файле `/etc/passwd`, осуществляя, таким образом, вход в сеанс пользователя.

Перед выводом приглашения login: на экран печатается содержимое файла

`/etc/issue` (или `/etc/issue.net`, если подключение осуществляется через сеть). А после входа в сеанс на экран выводится содержимое файла `/etc/motd` (Message Of The Day – Сообщение дня).

Загрузчик GRUB2.

Загрузчик GRUB (GRand Unified Bootloader) является современной альтернативой LILO.

В отличие от загрузчика LILO, который использует данные о точном расположении нужного образа ядра на магнитном носителе, загрузчик GRUB обращается к файловой системе на этом носителе для поиска файла ядра.

Примечание: Загрузчик LILO обращается к заданному номеру сектора диска, считывая определенное количество байт, а загрузчик GRUB обращается к файлу в файловой системе. Загрузчик GRUB распознает различные типы файловых систем, например, ext2, ext3, ReiserFS, XFS, JFS. GRUB способен работать с большими дисками, имеющими более 1024 цилиндров. При этом обеспечивается загрузка ядер ОС с разделов, расположенных в любом месте диска.

GRUB способен загружать ядра разнообразных операционных систем, например, GNU/Linux, GNU/HURD, FreeBSD и других.

Для операционных систем, загрузка ядер которых не поддерживается, загрузчик GRUB обеспечивает цепную загрузку (chainload) , то есть передачу управления загрузчику другой операционной системы.

GRUB поддерживает автоматическую декомпрессию загружаемых файлов, сжатых в формате zip.

GRUB обладает встроенной командной оболочкой, позволяющей гибко управлять процессом загрузки, а также автоматизировать его, задавая в конфигурационном файле загрузчика последовательности команд, которые должны быть выполнены в процессе загрузки.

Загрузчик GRUB поддерживает загрузку через локальную сеть посредством протокола TFTP (Trivial File Transfer Protocol) и поддерживает управление загрузкой с удаленного последовательного терминала.

В современных GNU/Linux как правило применяется загрузчик GRUB2.

GRUB2 — дальнейшее развитие проекта GRUB.

Основное новшество — возможность описать загрузку ОС без привязки к физическому расположению устройств.

В GRUB2 изменился формат файла конфигурации. 12.Конфигурационный файл — `/boot/grub2/grub.cfg`

Настоятельно **не рекомендуется** изменять файл конфигурации вручную. Вместо этого имеется специальная команда `grub2-mkconfig` (иногда `grub-mkconfig`)

`grub2-mkconfig` — создает конфигурацию на основе заготовок в каталоге

/etc/grub.d/, содержимого каталога /boot и /etc/default/grub

Если необходимо поменять параметры загрузки ядер Linux, то изменения необходимо вносить в файл `/etc/default/grub`

Дополнительные пункты меню загрузки определяются в файле /etc/grub.d/40_custom

Инициализация системы с помощью демона systemd

В стандартном подходе к инициализации Unix-подобных систем используется демон `init`, который запускается первым и получает PID=1, затем `init` запускает различные сервисы, в том числе сценарии инициализации `rc`

Такой подход имеет ряд ограничений:

Нет обратной связи между сервисом и демоном `init`. Демон `init` может только отслеживать работает или не работает запущенный им сервис (скрипт).

Нет возможности просмотреть результат загрузки сервисов после старта системы.

Монтирование, запуск служб, запуск сетевых сокетов, запуск виртуальных машин и т. д. в системах с `init` инициализацией это отдельные задачи, которые решаются разными средствами.

Не возможно параллельно запустить несколько сервисов, отложить запуск или запустить в фоновом режиме службу.

Systemd предлагает универсальное средство для инициализации, управления и настройки ОС.

Как правило программа `init` является ссылкой на `systemd` в таких системах.

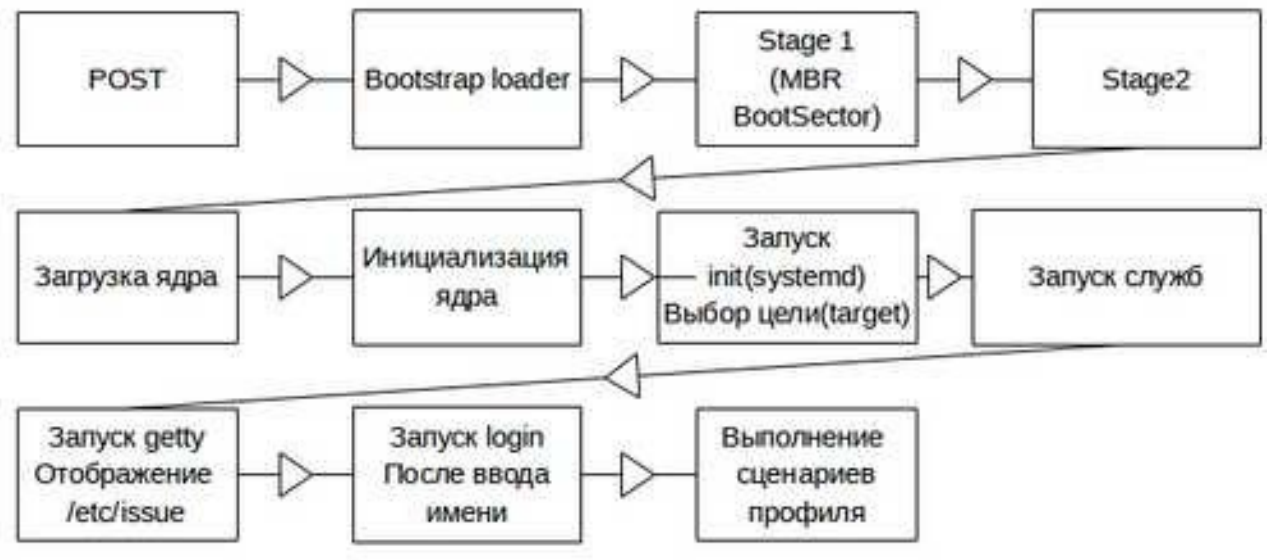


Рисунок 2: Инициализация системы с помощью демона systemd

В `systemd` отказались от понятия уровень исполнения, вместо этого используется понятие цель (`target`).

Цель — некоторое состояние в которое система должна прийти после инициализации нужных сервисов.

Не все цели предназначены для конечной загрузки.

Основополагающее понятие `systemd` — **unit**.

Unit — отдельный элемент для описания и управления в `systemd`.

С каждым юнитом может быть связан конфигурационный файл, который должен находиться в одном из каталогов.

/etc/systemd/system 2./run/systemd/system 3./usr/lib/systemd/system

Каталоги перечислены в порядке убывания приоритета.

Локальные изменения следует проводить в каталоге `/etc/systemd/system`

Юниты делятся на типы:

`target` — цель, используется для описания некоторого состояния системы, в которое мы должны попасть;

`service` — описывает процесс, который контролирует `systemd`;

`socket` — сетевой сокет, для каждого сокета должен существовать соответствующий ему сервис;

`timer` — описывает какой сервис должен запуститься в определенное время;

`device` — создает файл устройства;

`mount` — точка монтирования, дополнительная к `/etc/fstab`;

`automout` — то же, что `mount`, но монтирование производится по требованию;

`swap` — область подкачки;

`slice` — описание виртуальной машины или контейнера;

`snapshot` — юнит без юнит-файла, позволяет создать копию конфигурации и затем вернуться к ней;

`scope` — юнит без юнит-файла, создается программно для группировки процессов.

Команда `systemctl` управляет работой демона `systemd`

`systemctl list-dependencies` — показать от чего зависит юнит.

`systemctl list-units` — показывает список известных (включенных) юнитов.

`systemctl list-unit-files` — показывает список всех юнитов, в том числе выключенных, и их состояние.

`systemctl get-default` — показывает цель загрузки по умолчанию.

`systemctl set-default target` — устанавливает цель загрузки по умолчанию.

`systemctl {enable|disable} unit` — включение (разрешение на запуск) или выключение (запрет) юнита.

Примечание: некоторые сервисы все равно запускаются несмотря на запрет, т. к. это могут потребовать зависимые сервисы.

`systemctl {start|stop} unit` — запуск или остановка юнита.

Когда запускается `systemd` он определяет цель по умолчанию (`default.target`), заданную в настройках системы или во время загрузки ядра (параметр ядра `systemd.unit=нужная_цель.target`)

$ ls -l /etc/systemd/system/default.target

lrwxrwxrwx 1 root root 40 Nov 27 2016 /etc/systemd/system/default.target ->

/usr/lib/systemd/system/graphical.target

$ systemctl get-default graphical.target

После определения цели для загрузки определяется последовательность

запуска юнитов:

$ systemctl list-dependencies | grep target default.target

- └─multi-user.target
- ├─basic.target
- │ ├─paths.target
- │ ├─slices.target
- │ ├─sockets.target
- │ ├─sysinit.target
- │ │ ├─cryptsetup.target
- │ │ ├─local-fs.target
- │ │ └─swap.target
- │ └─timers.target
- ├─getty.target
- ├─nfs-client.target
- │ └─remote-fs-pre.target
- └─remote-fs.target
- └─nfs-client.target
- └─remote-fs-pre.target

$ systemctl list-dependencies swap.target swap.target

- ├─dev-disk-by\x2duuid-1496f7c8\x2d3bb8\x2d4395\x2d97f3\ x2d31f8a6083d9f.swap
- ├─dev-disk-by\x2duuid-4cc362d5\x2db46d\x2d4e20\x2d8d75\ x2dc4225eba0ea9.swap
- └─dev-zram0.swap

Подробнее это можно почитать в `man 7 bootup`.

Когда все нужные юниты будут запущены, то система считается работающей:

$ systemctl is-system-running

running

Если имеются проблемы, то мы увидим состояние `degraded`. Чтобы выяснить, что не так можно выполнить команду `systemctl --failed`:

$ systemctl is-system-running

degraded

$ systemctl --failed

UNIT LOAD ACTIVE SUB DESCRIPTION

kdump.service loaded failed failed Crash recovery kernel arming

LOAD = Reflects whether the unit definition was properly loaded.

ACTIVE = The high-level unit activation state, i.e. generalization of SUB. SUB = The low-level unit activation state, values depend on unit type.

1 loaded units listed. Pass --all to see loaded but inactive units, too. To show all installed unit files use 'systemctl list-unit-files'.

Попробуем понять в чем дело:

$ systemctl status kdump.service

kdump.service - Crash recovery kernel arming

Loaded: loaded (/usr/lib/systemd/system/kdump.service; enabled; vendor preset: enabled)

Active: failed (Result: exit-code) since Fri 2021-02-05 15:08:54 +05; 1h 57min ago

Process: 957 ExecStart=/usr/bin/kdumpctl start (code=exited, status=1/FAILURE) Main PID: 957 (code=exited, status=1/FAILURE)

Feb 05 15:08:54 cent8-stream systemd[1]: Starting Crash recovery kernel arming...

Feb 05 15:08:54 cent8-stream kdumpctl[957]: kdump: No memory reserved for crash kernel

Feb 05 15:08:54 cent8-stream kdumpctl[957]: kdump: Starting kdump: [FAILED]

Feb 05 15:08:54 cent8-stream systemd[1]: kdump.service: Main process exited, code=exited, status=1/FAILURE

Feb 05 15:08:54 cent8-stream systemd[1]: kdump.service: Failed with result 'exit-code'.

Feb 05 15:08:54 cent8-stream systemd[1]: Failed to start Crash recovery kernel arming.

Теперь у нас есть причина: `No memory reserved for crash kernel.` Чтобы исправить мы можем или установить параметр ядра `crashkernel=auto` или просто запретить запуск службы при старте системы:

$ sudo systemctl disable kdump

Removed /etc/systemd/system/multi-user.target.wants/kdump.service.

$ sudo systemctl stop kdump

В `systemd` есть три состояния у юнита:

`enabled` – стартовать зависимость автоматически

`disabled` – не запускать автоматически, но вручную можно

`masked` – никогда не запускать юнит.

Systemd имеет свою особую систему журналов работы, как, собственно, `systemd`, так и всех служб им запущенных.

Просмотр журналов `systemd` производится командой `journalctl`.

Примечание: Описание команды будет в главе посвященной системным журналам.

Остановка и перезагрузка системы.

Для немедленной остановки или перезагрузки системы можно использовать команды, соответственно, `/sbin/init 0` или `/sbin/init 6`. Однако для этого удобнее использовать команды `halt` для остановки или `reboot` для перезагрузки.

Команда `halt` вносит в файл `/var/log/wtmp` запись о том, что система была остановлена в это время.

Далее для останова вызывается команда `shutdown -h now`, останавливающая систему.

Опция `-f` (force) команды `halt` , заставляет систему остановиться без вызова `shutdown`.

Если команда `halt` вызвана с опцией `-n`, то перед остановом не будет произведена операция сброса содержимого кэша на диск.

При использовании команды `halt -d` запись в файл `/var/log/wtmp`

произведена не будет.

Остановка системы с последующим отключением питания будет произведена в результате выполнения команд `halt -p` или `poweroff`.

Примечание: Обычно команды `poweroff` и `reboot` реализованы в виде символических ссылок на файл команды `/sbin/halt`.

Основной командой для безопасной остановки или перезагрузки системы является /sbin/shutdown.

С помощью нее можно осуществлять как немедленную, так и отложенную остановку системы.

Эта команда посылает пользователям предупреждение о том, что система останавливается. Процессам, работающим в этот момент, посылается сигнал SIGTERM, получив который приложения могут корректно завершить свою работу.

Для останова системы команда `shutdown` посылает сигнал процессу `init` для перехода на 0-й или 6-й уровень исполнения при вызове с опцией, соответственно, `-h` или `-r`.

Пример:

/sbin/shutdown -h now

Примечание: Данная команда осуществит немедленную остановку системы, так как в качестве времени останова системы указан параметр `now`. Если же необходимо остановить или перезагрузить систему в заданное время, то его следует указать в качестве аргумента: /sbin/shutdown -r 17:00 'System will be rebooted at 17:00!'В этом примере перезагрузка (`-r`) системы будет произведена в 17:00, причем пользователи будут оповещены об этом с помощью строки сообщения, указанной в качестве аргумента.

Вместо использования точного указания времени можно указывать время задержки перед остановом. Если задержка измеряется секундами, то количество секунд следует указать после опции `-t`.

Количество минут задается опцией `+n` где `n` количество минут

Пример:

/sbin/shutdown -h +10

Примечание: В данном случае останов будет предпринят через 10 минут. Реально между этими двумя вариантами задания задержки существует разница: после опции `-t` задается время задержки в секундах дотого, как `shutdown` передаст сигнал `init` для перехода на другой уровень исполнения. Если же используется указание либо времени, либо задержки в минутах, то при этом реальное действие самой команды `shutdown` будет произведено с заданной задержкой. При этом пользователи, вошедшие в сеанс могут продолжать работать до начала останова, но новые сеансы не будут открыты.

1. В таблице, приведенной ниже, указаны часто используемые опции команды shutdown.

Опция	Назначение
-c	Отменить начавшийся останов системы.
-f	создает файл /fastboot, наличие которого позволяет не проверять файловую систему при загрузке.
-F	создает файл /forcefsck, наличие которого вынуждает проверять файловую систему при загрузке.
-h	Остановка системы.
-r	Перезагрузить систему.
-k	Послать пользователям сообщение, но не останавливать систему.

2. В системах с `systemd` выключением и перезагрузкой системы управляет сам `systemd`, командой `systemctl`:

1. systemctl reboot — перезагрузка;
2. systemctl halt — остановка;
3. `systemctl poweroff` — выключение питания.

Модуль 2. Установка РЕД ОС на ПК

Системные требования РЕД ОС

Минимальные системные требования для установки операционной системы **РЕД ОС** в конфигурациях «**Рабочая станция**» и «**Сервер**» для версий **7.2** и **7.3 и старше**.

Системные требования для **РЕД ОС 7.3**:

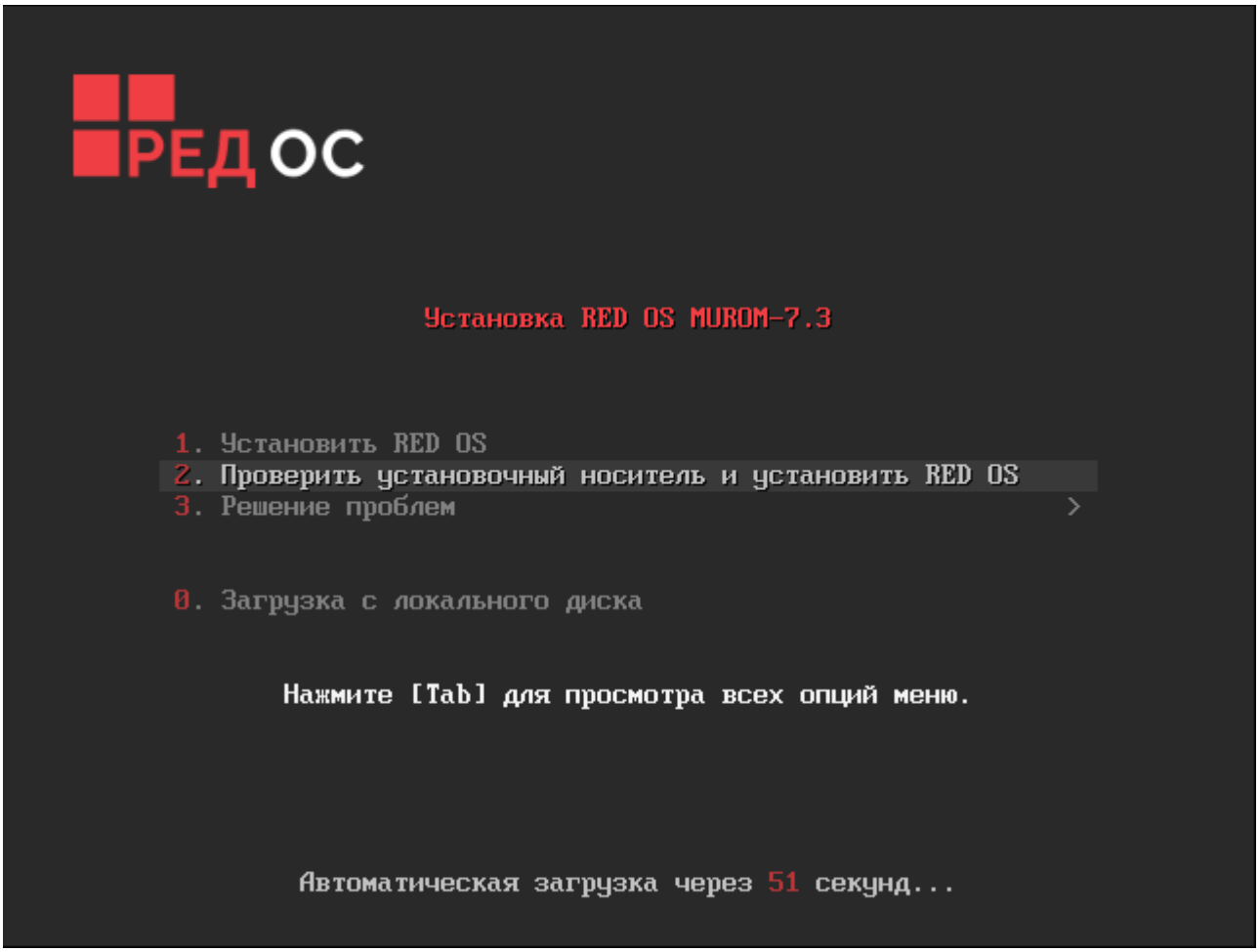
Конфигурация	**Рабочая станция**	**Сервер**
Процессор	X86_64 1.6 ГГц 2 ядра	X86_64 1.6 ГГц 2 ядра
Объем оперативной памяти	2 Гб	2 Гб
Объем свободного дискового пространства	20 Гб	20 Гб
Видеоадаптер	Поддержка режима SVGA800x600	Поддержка режима SVGA800x600

Системные требования для **РЕД ОС 7.2**:

Конфигурация	**Рабочая станция**	**Сервер**
Процессор	X86_64 1.6 ГГц 2 ядра	X86_64 1.6 ГГц 2 ядра
Объем оперативной памяти	2 Гб	2 Гб
Объем свободного дискового пространства	20 Гб	20 Гб
Видеоадаптер	Поддержка режима SVGA800x600	Поддержка режима SVGA800x600

Начало установки

Перед началом установки РЕД ОС необходимо настроить BIOS рабочей станции на загрузку с носителя информации (DVD-диска или флеш-накопителя), куда записан дистрибутив. Способ входа в меню BIOS и расположение конкретных настроек может сильно отличаться, в зависимости от используемой материнской платы оборудования рабочей станции. Чаще всего для входа в BIOS необходимо нажать функциональную клавишу *Delete* на стационарных рабочих станциях или функциональную клавишу *F1 (F9)* на мобильных рабочих станциях (ноутбуках) в момент начала загрузки компьютера. Для получения более подробных сведений по настройке необходимо обратиться к документации на используемое оборудование.



Начальное меню установки РЕД ОС

Загрузка с установочного носителя начинается с меню, в котором перечислено несколько вариантов загрузки, причем установка системы - это только одна из возможностей. Из данного меню можно:

запустить тестирование носителя данных и только после этого перейти к процедурам установки ОС;

используя меню «**Решение проблем**», запустить проверку памяти и диагностику оборудования;

запустить уже установленную ОС на жестком диске;

загрузить ОС в аварийном режиме.

Манипулятор графической информации (мышь) на этом этапе установки не поддерживается, поэтому для выбора различных вариантов и опций установки необходимо воспользоваться функциональными клавишами клавиатуры - стрелками. Можно скорректировать параметры запуска любого пункта начального меню установки РЕД ОС, нажав функциональную клавишу клавиатуры *TAB*.

Установка РЕД ОС автоматически осуществляется в графическом режиме с выводом текстовой информации на выбранном в начале процесса установки языке.

По умолчанию при установке используется графический видеорежим (разрешение экрана) 800x600. При возникновении проблем запуска установки в графическом режиме существует возможность альтернативной установки системы - не графический (текстовый) режим.

Чтобы начать процесс установки, нужно клавишами перемещения курсора «вверх»/«вниз» выбрать пункт меню «**Установить RED OS**» и нажать *Enter*. В начальном загрузчике установлено небольшое время ожидания продолжительностью 1 минута. Если в этот момент не предпринимать никаких действий, то будет загружена установка системы с проверкой установочного носителя. Если пропущен нужный момент, нужно перезагрузить компьютер и до истечения установленного времени ожидания выбрать нужный пункт в меню загрузчика.

Начальный этап установки не требует вмешательства пользователя: происходит автоматическое определение оборудования и запуск компонентов программы установки. Прервать начальный этап установки и вернуться в начальное меню установки РЕД ОС можно, нажав клавишу *ESC*.

Для штатной установки дистрибутива РЕД ОС используется загрузочный оптический носитель информации - DVD-диск из комплекта поставки дистрибутива РЕД ОС. Если производится установка с такого диска, можно пропустить этот раздел и сразу перейти к следующему разделу.

Установка с оптического носителя информации (DVD-диска) - это лишь один из возможных способов установки РЕД ОС. Он подходит для большинства случаев, но не работает, например, в случае отсутствия на компьютере накопителя на оптических носителях информации - DVD-привода. Для таких случаев поддерживаются альтернативные методы установки. Важно понимать, что для начала установки необходимо иметь две вещи: возможность загрузить компьютер и доступ к установочным файлам. В случае установочного дистрибутивного DVD-диска эти две возможности предоставляются самим диском: он является загрузочным и содержит все необходимые для установки файлы. Однако вполне допустим и такой вариант: первоначальная загрузка происходит со специально подготовленного flash-диска, а установочные файлы берутся с FTP-сервера сети.

Таким образом, для установки дистрибутива необходимо:

Выбрать способ первоначальной загрузки компьютера;

Выбрать источник установки.

Для загрузки компьютера с целью установки операционной системы необходимо воспользоваться носителем, содержащим начальный загрузчик.

После первоначальной загрузки с одного из поддерживаемых носителей можно выбрать *источник установки* - место, откуда программа установки будет

брать все необходимые при установке данные (прежде всего, устанавливаемое ПО). Так как установка системы возможна не только с лазерного диска, то можно выбрать один из поддерживаемых альтернативных источников установки.

Источники установки:

Сетевые:

FTP-сервер;

NFS-сервер;

HTTP-сервер.

Локальные:

Загрузочный флеш-диск.

Условием для всех способов установки является доступность дерева файлов, аналогичного содержимому установочного диска.

До того, как будет произведена установка РЕД ОС на жесткий диск СВТ, программа установки работает с образом системы, загруженным в оперативную память компьютера.

Если инициализация оборудования СВТ завершилась успешно, будет запущен псевдографический интерфейс программы-установщика (**anaconda**). Процесс установки реализован в виде «мастера» установки, который представляет из себя интерактивный графический интерфейс, в котором пользователю предлагается отвечать на вопросы и указывать необходимые опции РЕД ОС. Мастер установки разделен на шаги, каждый шаг посвящен настройке или установке определенного сервиса системы. Шаги нужно проходить последовательно, переход к следующему шагу происходит по нажатию кнопки «**Далее**». При помощи кнопки «**Назад**», при необходимости, можно вернуться к уже пройденному шагу и изменить настройки. На некоторых этапах установки возможность перехода к предыдущему шагу ограничена теми шагами, где нет зависимости от данных, введенных ранее.

Если по каким-то причинам возникла необходимость прекратить установку, нажмите ***Reset*** на системном блоке компьютера. Помните, что совершенно *безопасно* прекращать установку только до шага «**Подготовка диска**», поскольку до этого момента не производится никаких изменений на жестком диске. Если прервать установку между шагами «**Подготовка диска**» и «**Установка загрузчика**», вероятно, что после этого с жесткого диска не сможет загрузиться ни одна из установленных систем.

Технические сведения о ходе установки можно посмотреть, нажав *«Ctrl+Alt+F1»*, вернуться к программе установки - *«Ctrl+Alt+F7»*. По

нажатию *«Ctrl+Alt+F2»* откроется отладочная виртуальная консоль.

Во время установки РЕД ОС выполняются следующие шаги:

Выбор типа накопителя CBT;

Присвоение имени компьютера в сети и настройка сетевых интерфейсов;

Выбор часового пояса;

Задание пароля администратора системы;

Подготовка разделов диска;

Выбор типа установки: сервер или рабочая станция;

Установка системы;

Установка загрузчика;

Перезагрузка системы;

Лицензионный договор;

Создание системного пользователя;

Установка даты и времени;

Сохранение настроек;

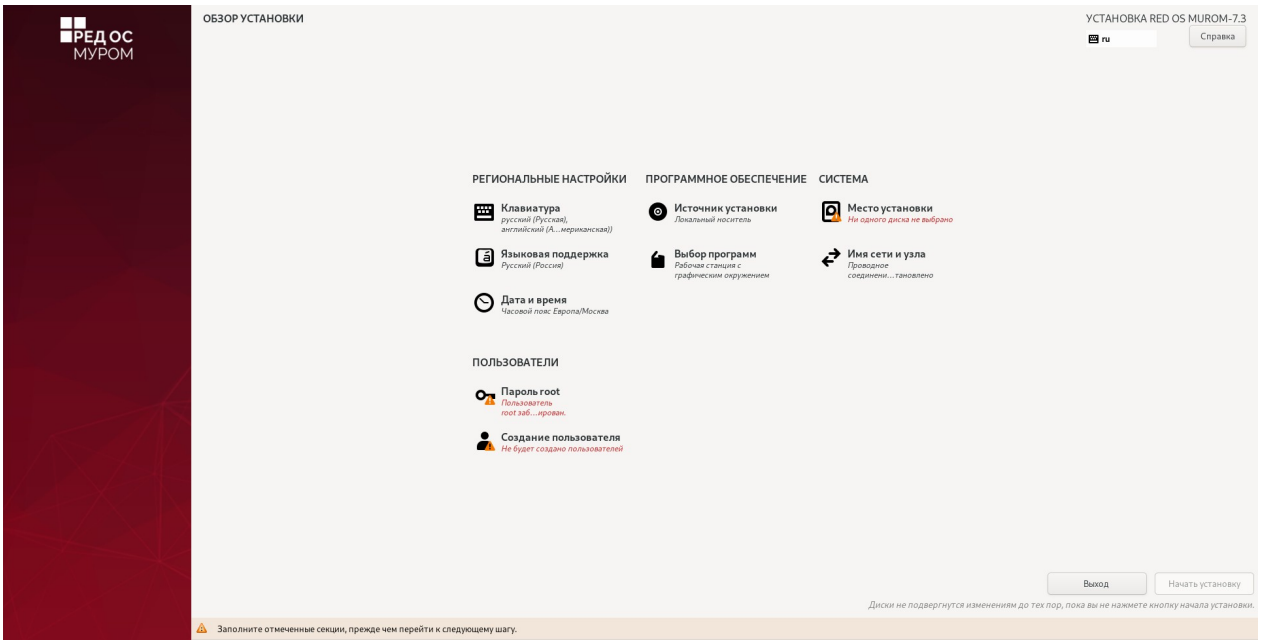
Завершение установки.

Язык

Язык интерфейса программы установки можно выбрать на самом первом этапе процесса установки. Язык устанавливаемой РЕД ОС будет русским, если был выбран русский язык, или английским, если был выбран любой другой язык.

Дополнительным языком РЕД ОС является английский язык. Другие дополнительные языковые пакеты можно установить из репозитория после завершения установки РЕД ОС.
Переключение раскладки клавиатуры при установке РЕД ОС и в графическом интерфейсе РЕД ОС выполняется нажатием комбинации функциональных клавиш *«Alt+Shift»*.

После выбора языка необходимо произвести первоначальную конфигурацию установщика и параметров будущей системы.

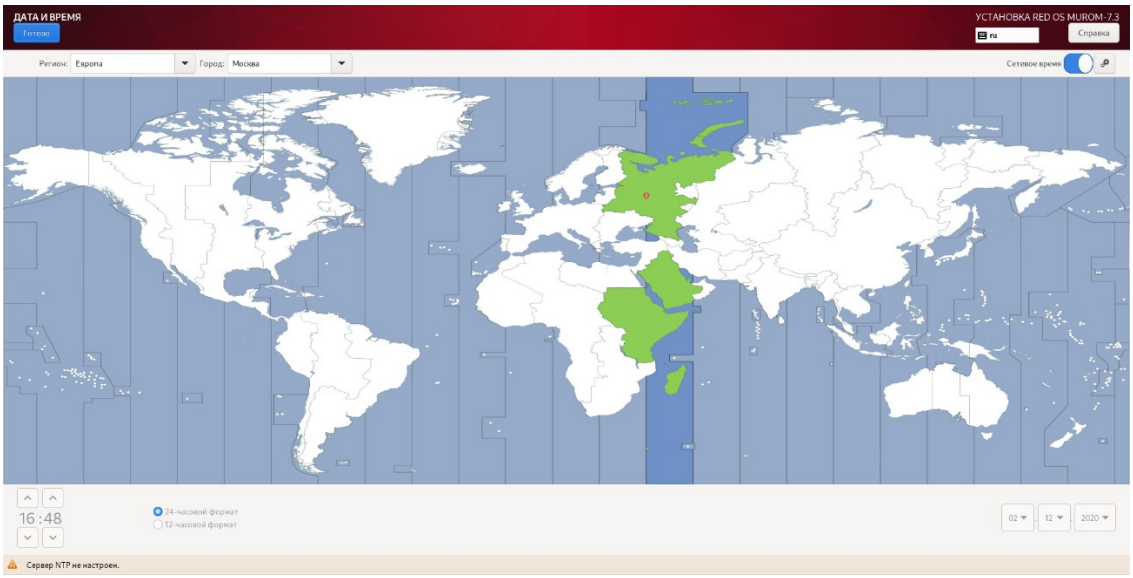


В этом окне необходимо задать региональные настройки, состав программного обеспечения и системные настройки. Здесь и в последующих окнах установщика красным цветом выводятся подсказки у тех вкладок, которые должны быть обязательно заполнены до перехода к следующему шагу установки.

Рассмотрим подробнее каждую вкладку окна.

Дата и время

Во окне настройки даты и времени можно выбрать текущий регион и город и установить используемое локальное время, дату и формат времени.



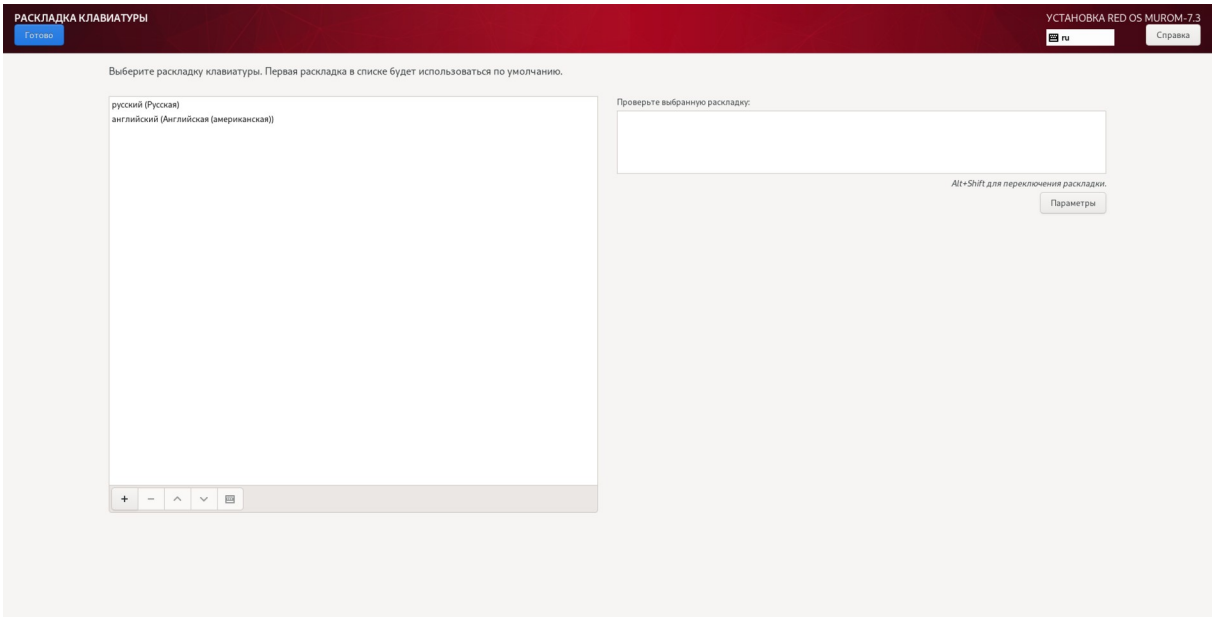
Если установлена сеть, и есть доступ к глобальной сети, можно разрешить автоматическую настройку времени с помощью службы **ntp** — сетевое время.

Данную настройку можно изменить после завершения установки РЕД ОС. По умолчанию устанавливается часовой пояс UTC+03:00 (Европа/Москва).

Здесь и далее возврат в предыдущее меню осуществляется с помощью кнопки «Готово».

Клавиатура

Во окне настройки клавиатуры можно выбрать используемые в ОС раскладки.

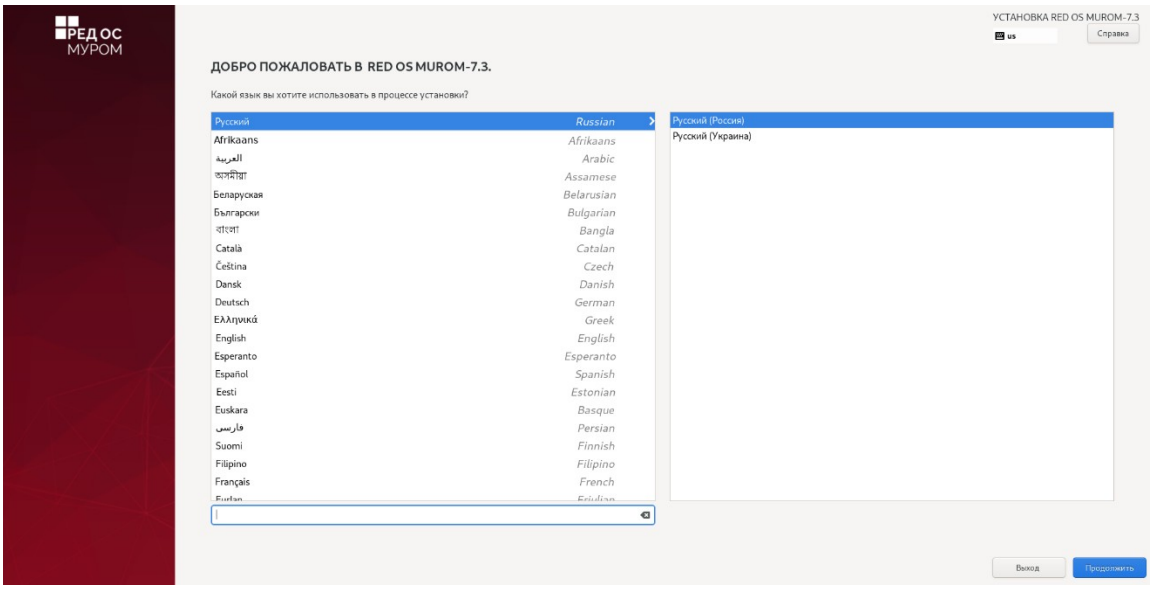


В отдельном поле ввода можно проверить корректность отображения вводимых символов.

Первая раскладка в списке будет использоваться по умолчанию.

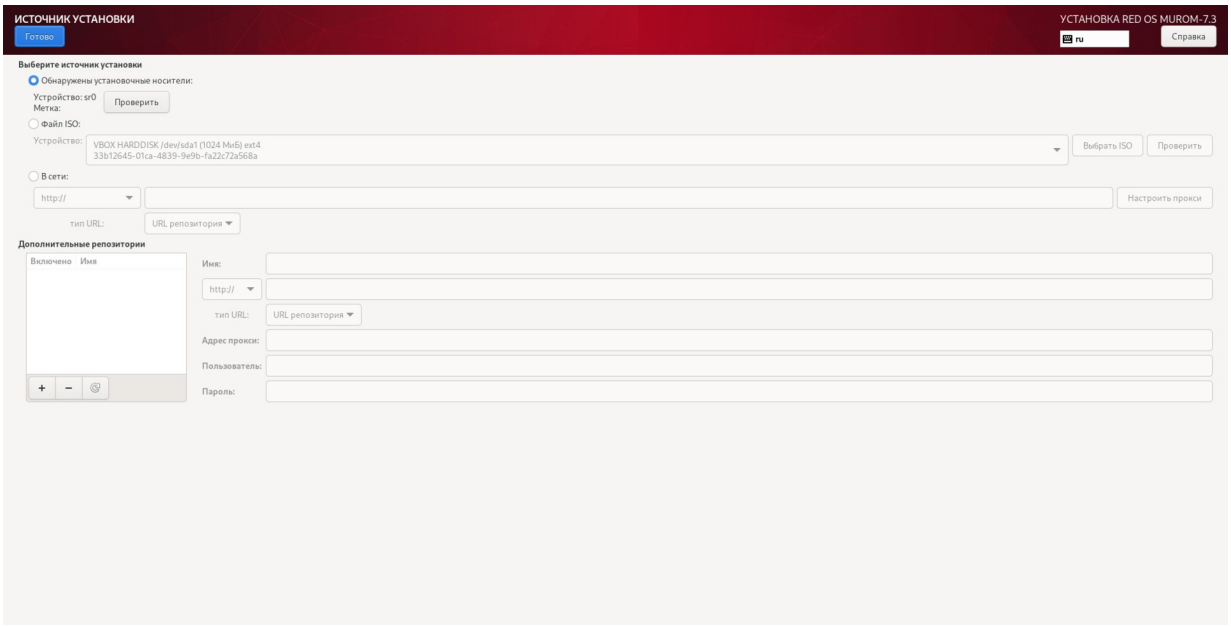
Языковая поддержка

В окне языковой поддержки можно задать языки, которые будут установлены в системе. Настройка и выбор соответствующего языка будут доступны в настройках системы после установки. Также после установки системы можно будет добавить дополнительные языки, которые не были установлены ранее. По умолчанию используется русский язык интерфейса.



Источник установки

В меню выбора источников установки можно задать, откуда система будет получать пакеты в процессе установки.



Источниками установки могут быть локальные носители или сетевые источники.

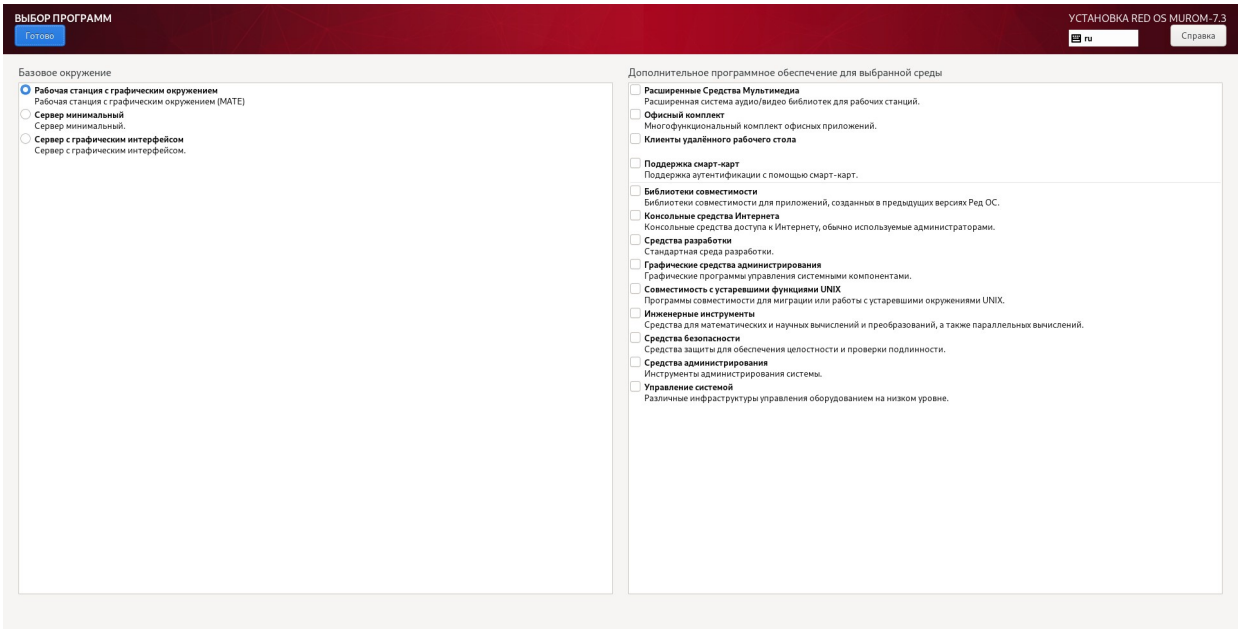
Выбор программ

В меню выбора программ можно задать, какая конфигурация из числа базового окружения будет установлена:

рабочая станция,

сервер минимальный,

сервер с графическим интерфейсом.



Дополнительно для каждой из выбранных конфигураций можно настроить группы пакетов, которые будут установлены.

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

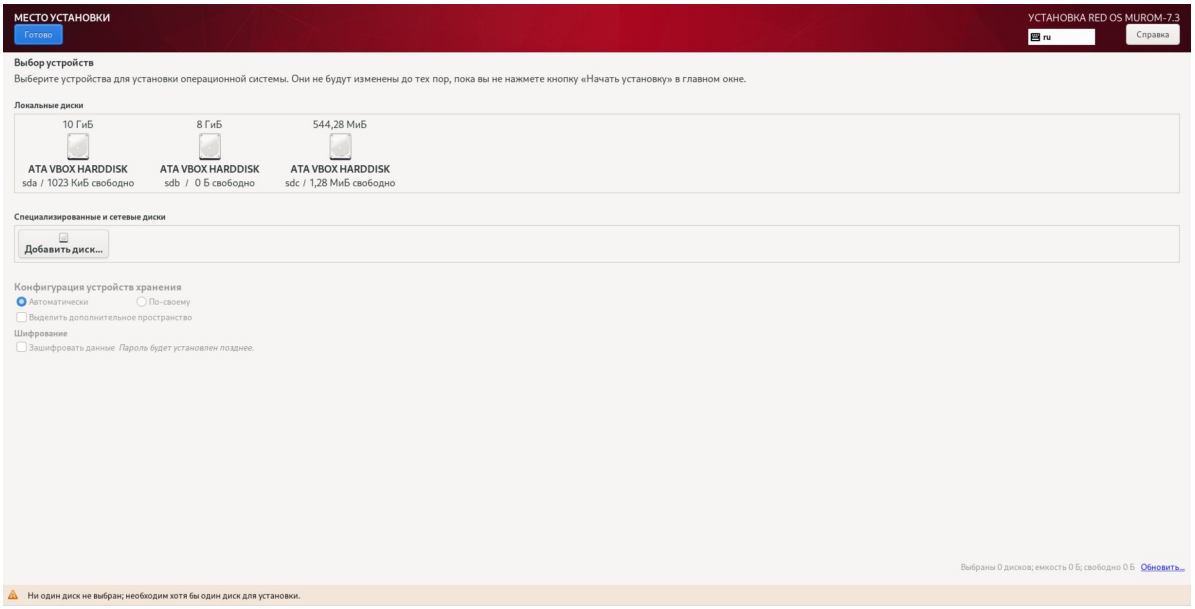
В РЕД ОС все операции установки и удаления производятся над пакетами - отдельными компонентами системы. Пакет и программа соотносятся неоднозначно: иногда одна программа состоит из нескольких пакетов, иногда один пакет включает несколько программ.

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

Расположение установки

В меню выбора расположения установки можно выбрать устройство для

установки операционной системы.



Переход к этому шагу может занять некоторое время. Время ожидания может быть разным и зависит от производительности компьютера, объёма и типов используемых накопителей, количества существующих разделов на них.

На этом этапе подготавливается площадка для установки РЕД ОС, в первую очередь - выделяется свободное место на диске.
Можно выбрать и использовать профили разбиения диска. Профиль - это шаблон распределения места на диске для установки РЕД ОС. Можно выбрать:

"создать разделы автоматически",

"я настрою разделы".

Первый профиль предполагает автоматическое разбиение диска. Будет выбрано оптимальное расположение ОС. Необратимые изменения разделов на жестком диске требуют подтверждения со стороны пользователя. После подтверждения внесенные изменения сохраняются на жестком диске/дисках СВТ.

При необходимости дополнительной защиты информации для разделов жесткого диска, на которые устанавливается РЕД ОС, можно включить шифрование. При краже или утере такого жесткого диска получить доступ к содержимому файловой системы без дополнительных программно-аппаратных решений по дешифрованию не удастся.

Если для применения одного из профилей автоматической разметки доступного места окажется недостаточно, будет выведено сообщение о невозможности выполнения операции разбиения диска.

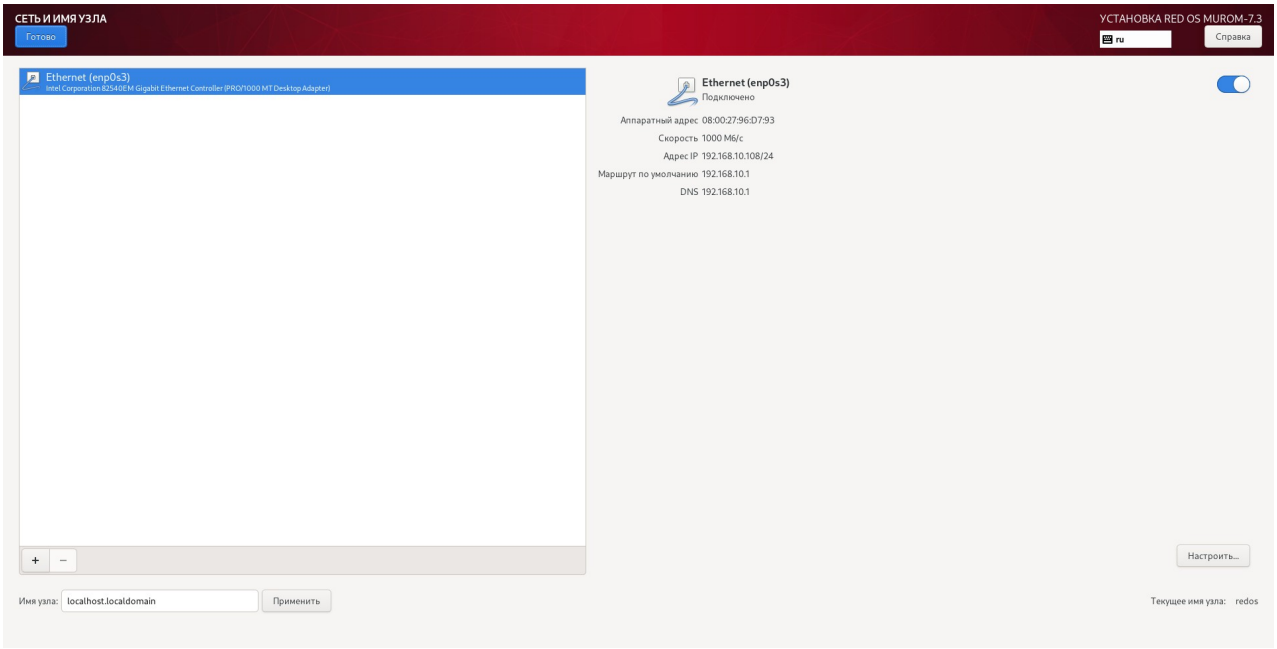
При необходимости освободить часть дискового пространства, следует воспользоваться профилем разбиения вручную. Можно удалить некоторые из существующих разделов или содержащиеся в них файловые системы. После этого

можно создать необходимые разделы самостоятельно или вернуться к шагу выбора профиля. Выбор этой возможности требует знаний об устройстве диска и технологиях его разбиения.

По нажатию «Готово» будет произведена запись новой таблицы разделов на диск и форматирование разделов. Разделы, только что созданные на диске программой установки, пока не содержат данных и поэтому форматируются без предупреждения. Уже существовавшие, но изменённые разделы, которые будут отформатированы, помечаются специальным значком в колонке «Файловая система».

Сеть и имя узла

В меню выбора настройки сети можно активировать соединение с сетью и задать имя узла.



На данном этапе необходимо задать имя компьютера в сети (хоста). При наличии сети имя компьютера используется для однозначного определения каждого компьютера. Имя компьютера состоит непосредственно из имени компьютера и имени домена в сети, к которому принадлежит компьютер. Имя компьютера и имя домена разделяются знаком «.» . При наличии домена сети имя должно даваться полностью. При отсутствии сети домену также может быть дано произвольное имя.

В качестве букв в имени компьютера разрешаются только буквы латиницы. В имени компьютера не допускается использование заглавных букв, пробелов и специальных символов.

Также на данном этапе можно сконфигурировать параметры настройки сетевых интерфейсов: автоматическое включение интерфейсов, MAC-адреса сетевых интерфейсов, параметры сетевых протоколов.

Общая информация

Прежде чем приступить к разметке дисковой подсистемы при установке ОС, вспомним про устройство файловой системы РЕД ОС.

Рисунок 1 - Описание иерархии каталогов

Как мы можем видеть из рисунка, каждый каталог используется под определенные задачи. При установке РЕД ОС в ручном режиме разметки диска можно определенным образом организовать дисковое пространство под ОС, например, для основного корневого раздела **/** ,
директорий **/home**, **/boot** или **/var** выделить отдельные разделы диска или вообще вынести какой-либо каталог на отдельный диск.

Такое разбиение изолирует их один от другого, что может быть полезным в случае, например, если на домашнем (**/home**) каталоге, вынесенном на отдельный раздел, закончится место, то система все еще сможет нормально работать, потому что это никак не касается корневого раздела. Также, в целях удобства, выделение для **home** отдельного раздела позволит переустановить операционную систему, сохранив пользовательские данные. Таким образом, каталоги могут быть

разграничены не только по задачам, но и по своему местонахождению: на разных разделах одного диска или разделах других дисков. Кроме того, ручная разметка дает возможность создавать программный **RAID** и более гибко настраивать файловую подсистему, чем в автоматическом режиме.

Приступаем к разметке диска. В мастере установки ОС выберем жесткие диски для установки РЕД ОС. В нижнем правом углу виден признак количества выбранных дисков. Укажите «**Я настрою разделы**». В результате мы перейдем к ручной разметке дисков.

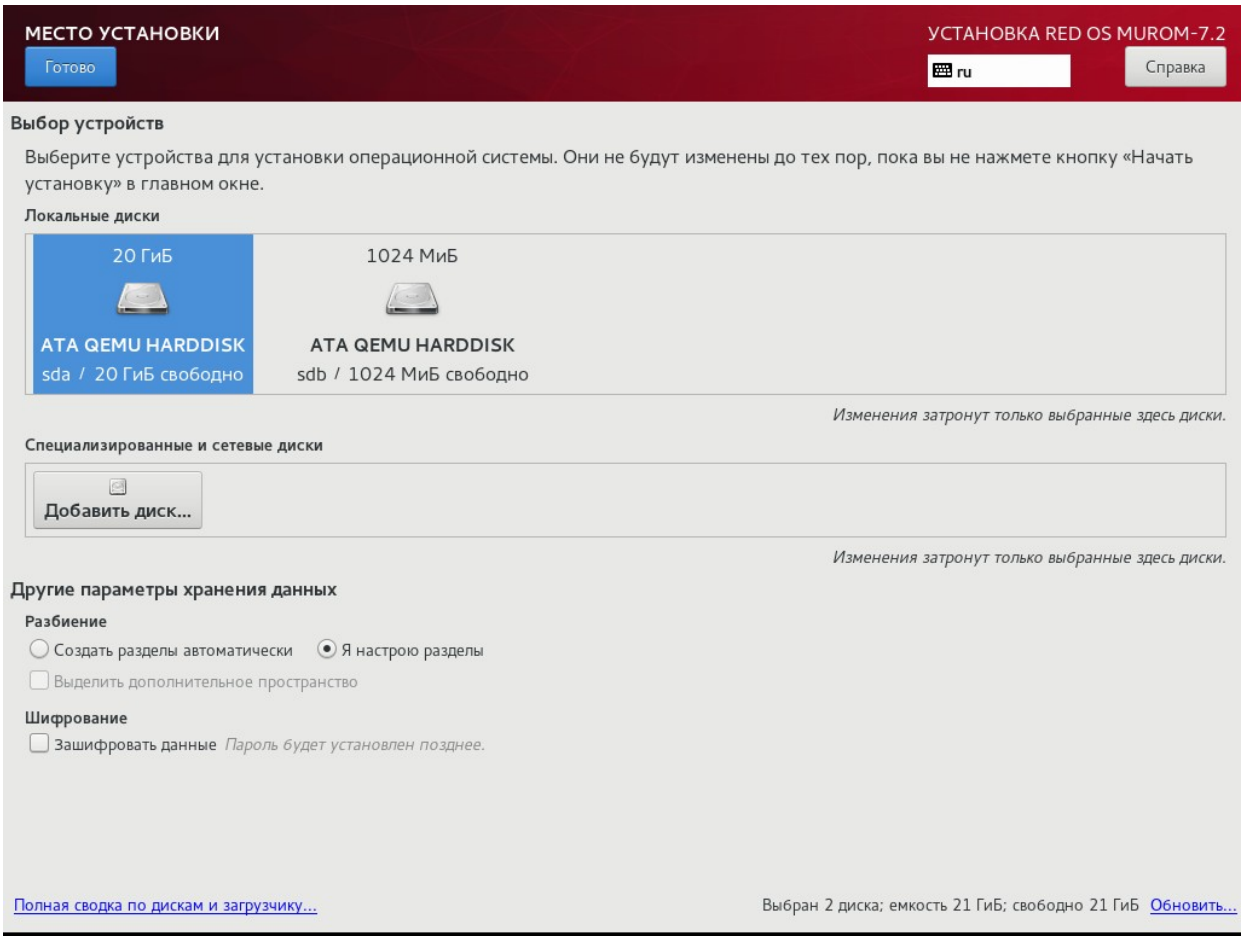


Рисунок 2 — Выбор дисков

На следующем шаге предлагается выбирать схему разбиения диска.

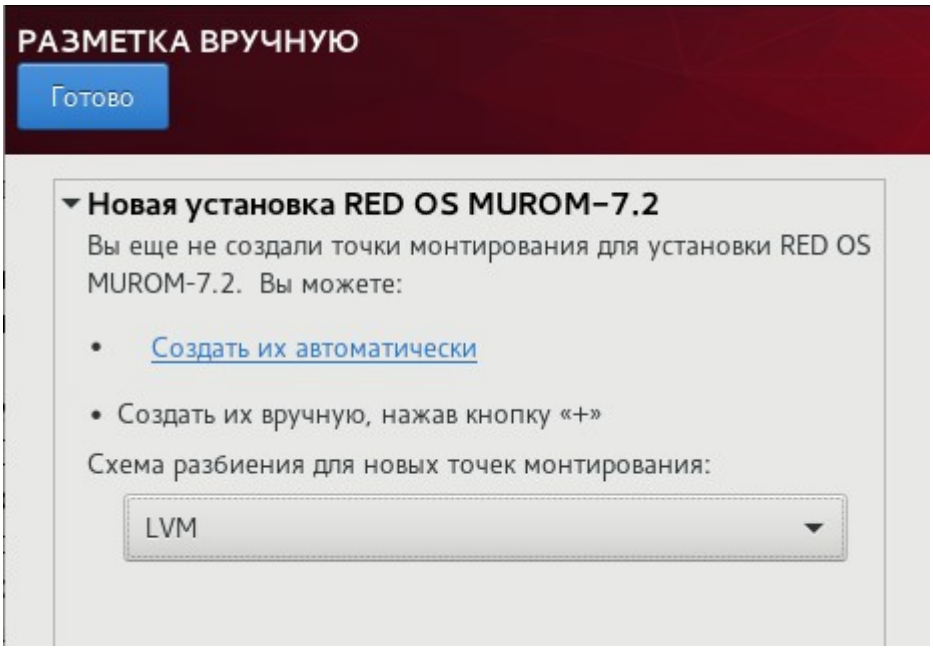


Рисунок 3 — Выбор схемы разбиения диска

Схемы разметки дисков

Стандартный раздел - позволяет создать обычные дисковые разделы, простые в управлении.

Btrfs (можно произносить как «*Butter FS*», «*Better FS*» или «*B-Tree FS*») - файловая система, которая может работать с большим числом файлов, файлами и томами гораздо большего размера, по сравнению с **ext2**, **ext3** и **ext4**, имеется поддержка снапшотов, сжатие и подтома.

Минусы: хотя **btrfs** активно развивается и перешла уже в статус стабильной, у пользователей все еще встречаются проблемы, приводящие к потере данных, а также высокая подверженность фрагментации.

LVM - позволяет создавать логические группы и тома хранения данных без непосредственной переразметки жесткого диска. Например, если у вас используется несколько жестких дисков, то при использовании LVM-разметки можно создать несколько или одну логическую группу, а в ней уже логические тома, которые в дальнейшем можно разбить на разделы и отформатировать под необходимую вам файловую систему — **ext4**, **ext3**, **ntfs** и другие.

Возможности:

способность объединять диски в один логический том;

изменение размера файловых систем, легкость при добавлении дисков в то время, когда система активна;

использование RAID (чередование и зеркалирование);

поддержка моментальных снимков;

Недостатки:

сложность управления (предъявляются довольно высокие требования по администрированию файловой системы);

трудный процесс восстановления данных в случае сбоя LVM из-за более сложных структур на диске;

снимки сложны в использовании, медленны и содержат ошибки.

Динамический LVM - перераспределяет свободное пространство между устройствами в зависимости от требований программ и при необходимости пул пространства может наращиваться динамически.

Пример схемы LVM-разметки 4-х физических дисков, из которых, в свою очередь, создаются два физических тома, а из них одна большая логическая группа объемом, равным объему всех физ. дисков. В дальнейшем из группы VG создаются логические тома с точками монтирования см. Рисунок 4.

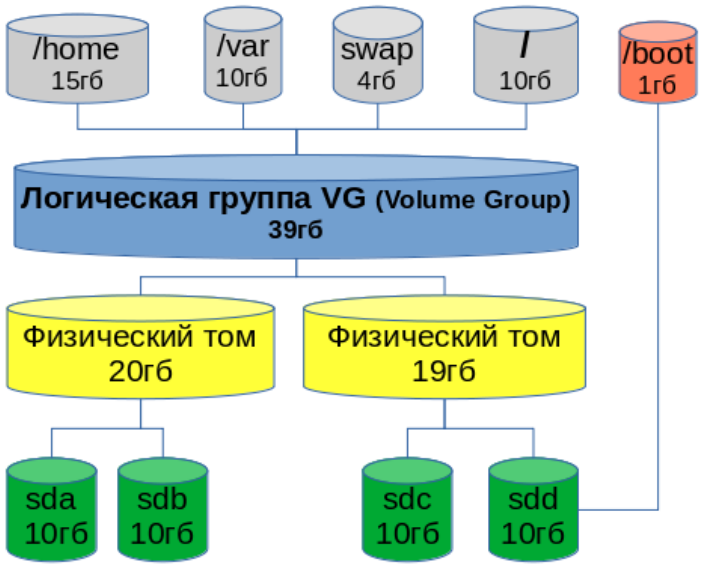


Рисунок 4 - Пример LVM-разметки дисков

Создание разделов, описание интерфейса разметки

Выберем схему со стандартной разметкой диска, которая является более простой, как в плане понимания, так и в сопровождении по сравнению с LVM.

Для обычных пользователей РЕД ОС рекомендуем создавать следующие разделы:

swap (раздел для подкачки ОЗУ);

/boot (загрузочный раздел);

/home (данные пользователей);

/ (корневой раздел).

Выбираем схему разбиения «**Стандартный раздел**»,

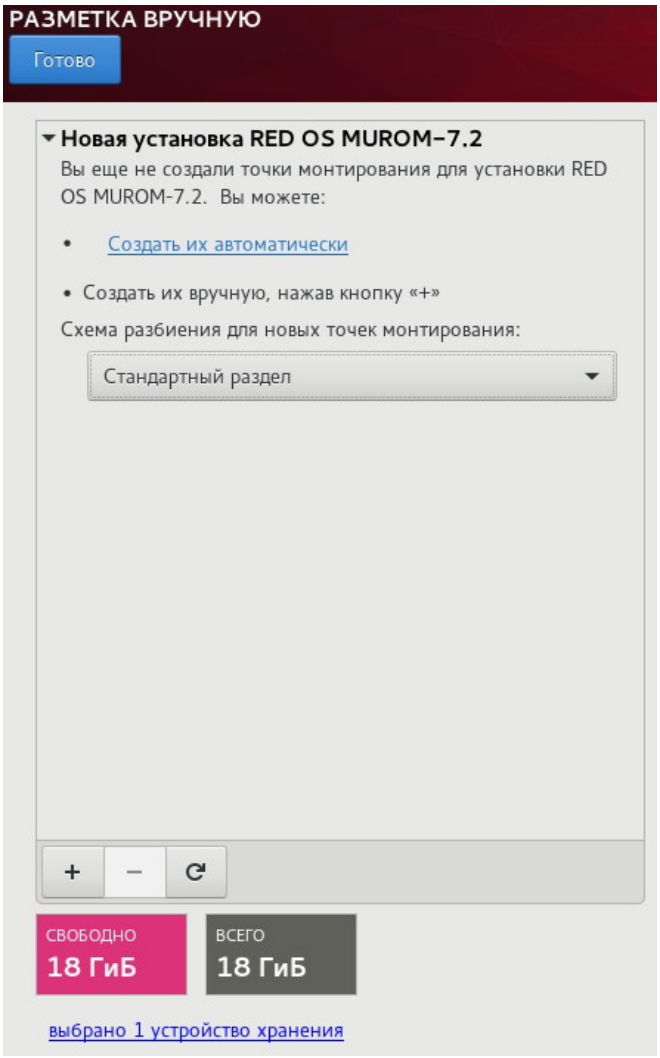


Рисунок 5 - Схема разбиения

далее можно создать все разделы вручную — кнопка плюс (**+**), кнопка минус (**-**) удаляет выбранный раздел.

Рисунок 6 - Добавление разделов

Но если нажать на кнопку «**Создать их автоматически**», то в результате будут созданы все необходимые разделы для установки ОС: **boot**, **swap**, **/** и нам не придется их создавать, а останется выделить только раздел **/home** для домашнего

каталога.

Как создать и зачем выделять home?

Как видим, корневой раздел / самый большой, поэтому возьмем от него часть для домашнего раздела **/home**. Уменьшаем размер, указав в окне «**Требуемый размер**» размер раздела, указываем меньше, чем сейчас есть, нажимаем кнопку «**Применить**». Заметим, при разметке диска нужно учитывать, что и для корневого раздела нужно оставить свободное место, минимально допустимый размер для корня / - не менее 10 ГБ.

Далее нажмите на плюс (+), выберите точку монтирования **/home**, размер оставляем пустым (но можно и указать необходимый). Таким образом, под **/home** выделится все остальное свободное место, которое мы забрали у корневого **/**. Тип файловой системы оставим по умолчанию — **ext4**, как более технологически продвинутую, но также в окне инсталлятора представлены и другие файловые системы.

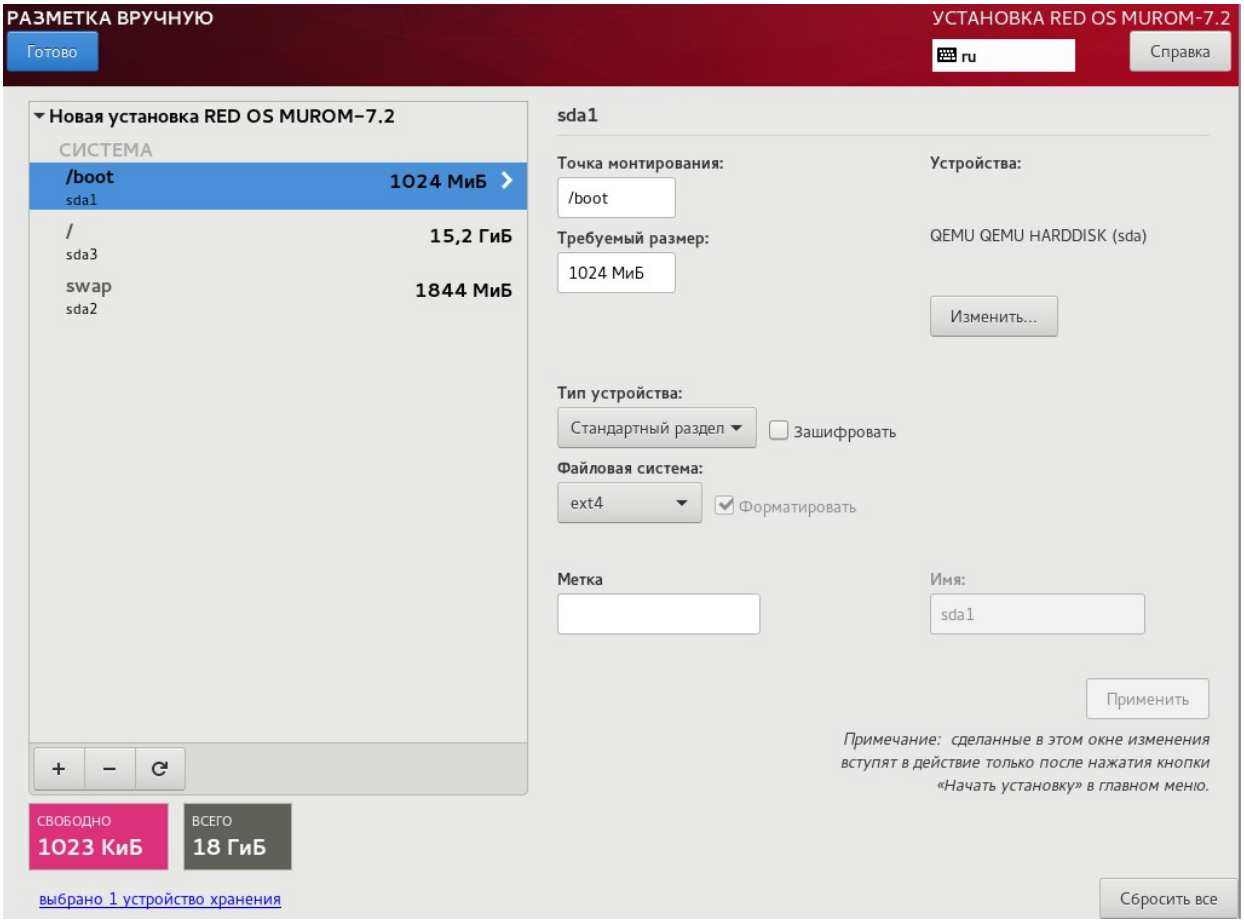


Рисунок 7.1 — Стандартная разметка диска

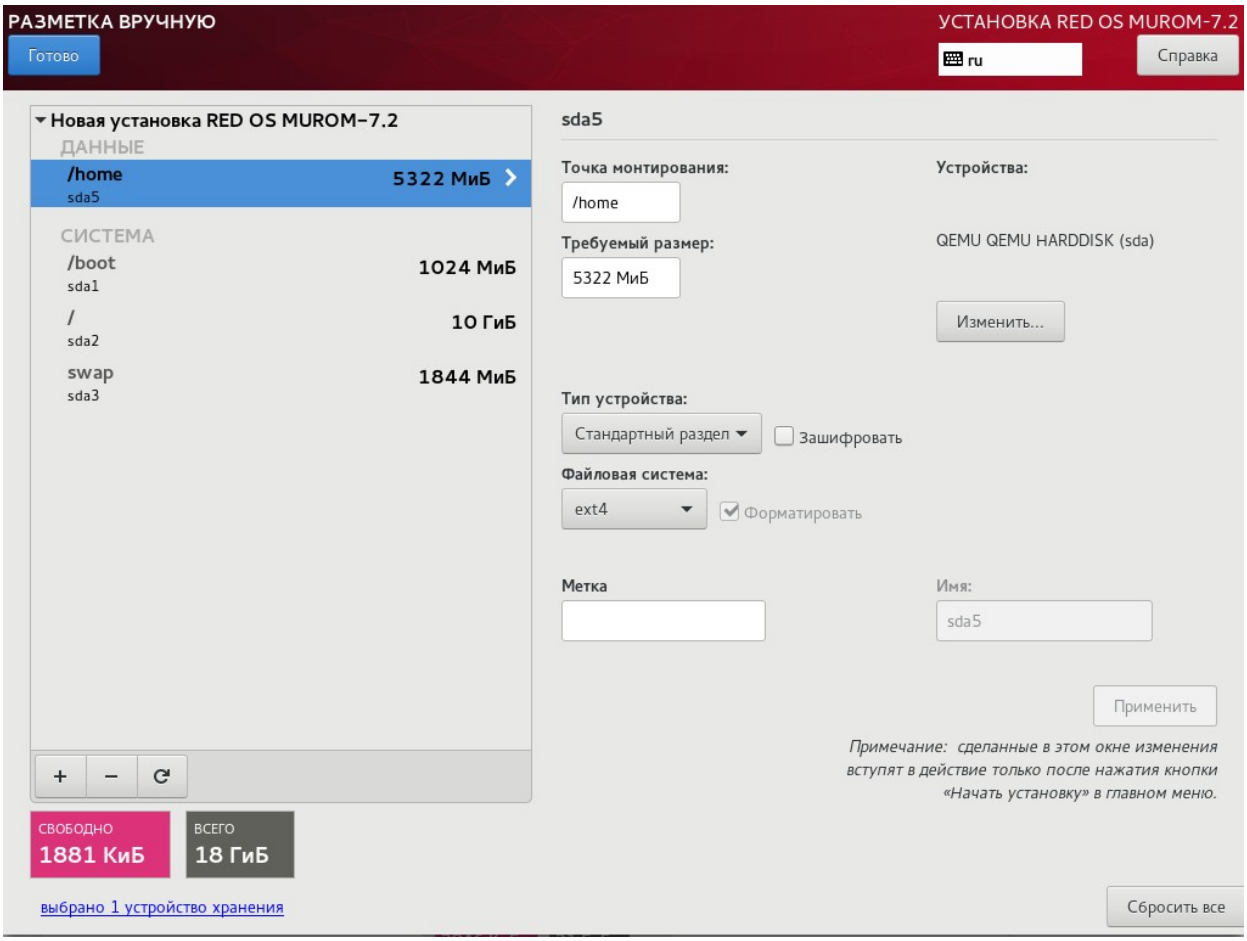


Рисунок 7.2 — Стандартная разметка диска

Другие параметры диалога разметки диска:

Точка монтирования: указывается директория для подключения будущего раздела;

Требуемый размер: указывается размер раздела;

Устройство: выбирается физическое устройство, на котором будет располагаться раздел;

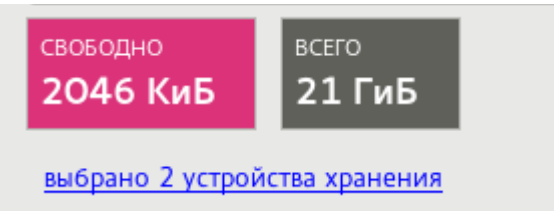
Тип устройства: стандартный раздел, LVM, динамический LVM, RAID, btrfs;

Файловая система: выбирается файловая система для раздела;

Метка: назначается имя разделу под понятным вам именем, не обязательный параметр;

Имя: имя раздела;

В нижнем левом углу показывается оставшееся свободное и израсходованное место под разметку.



Разделы swap, boot, home

Рассмотрим для чего предназначен каждый из созданных разделов.

Раздел swap используется для поддержки виртуальной памяти. Данные попадают в раздел подкачки, когда системе не хватает оперативной памяти для их обработки.

Размер раздела для **swap** - минимум половина от ОЗУ, это если пользователь не собирается использовать режим глубокого сна (гибернации). При гибернации вся оперативная память перемещается в **swap**, в этом случае его размер должен быть не меньше размера оперативной памяти!

Раздел /boot содержит ядро операционной системы и файлы начальной загрузки **grub2** 1024 МБ (1 ГБ) будет достаточно для данного раздела.
Раздел **boot** лучше создать вне группы томов, а создавать его как отдельный стандартный раздел.

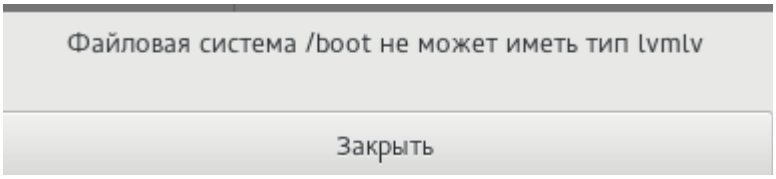
Ряд ограничений, по которым boot выделяется в отдельный стандартный раздел:

раздел **boot** не может располагаться на шифрованных разделах;

раздел **boot** не может находится на разделах, файловая система которых не поддерживается загрузчиком **grub2**;

загрузчик быстрее обнаружит свои файлы, если они находятся на небольшом разделе в начале диска;

файловая система раздела /boot не может иметь тип LVM (для случая, когда используется LVM-разметка диска).



Раздел /home используется для хранения профилей пользователя, настроек рабочего окружения и документов. Как мы уже говорили, выделение отдельного раздела для каталога **/home** позволяет отделить пространство хранения пользовательских данных от системных. Пользовательские данные, размещенные на отдельном разделе, не пострадают при переустановке системы, а также надежнее защищены от ошибок пользователя во время переноса данных. При переполнении домашнего каталога видео, музыкой и прочим, это не скажется на работе системы в

целом, как если бы **home** был на одном разделе с корневой файловой системой, в результате чего закончилось бы все дисковое пространство.

В корневом разделе / будут находиться все системные файлы и к нему будут монтироваться (подключаться) все выше созданные разделы. Минимальный рекомендуемый размер раздела - 10 ГБ.

Пример с размещением home на отдельном диске

Если в компьютере имеется несколько жестких дисков, то для надежности и, в дальнейшем, простоты администрирования можно, например, точку монтирования /home разместить на **отдельном диске**, тем самым изолировав ее от основной файловой системы. Давайте разместим каталог пользователя (**home**) на диске sdb, размер которого в нашем примере 1 Гб. Это легко сделать, выбрав для точки монтирования **/home** нужный диск. Раздел «**Устройство**» - «**Изменить**». Для сохранения настроек нажмите «**Применить**».

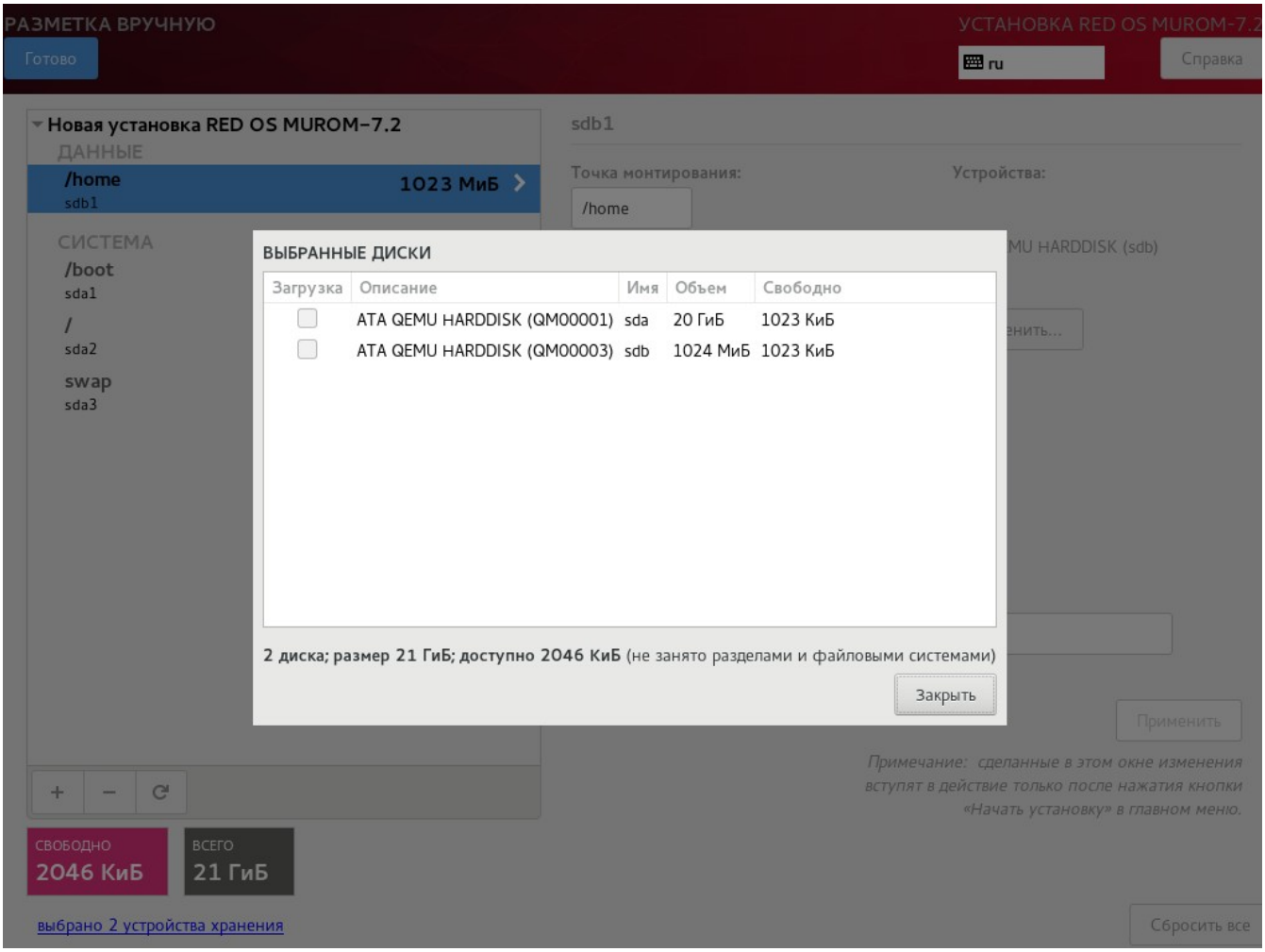
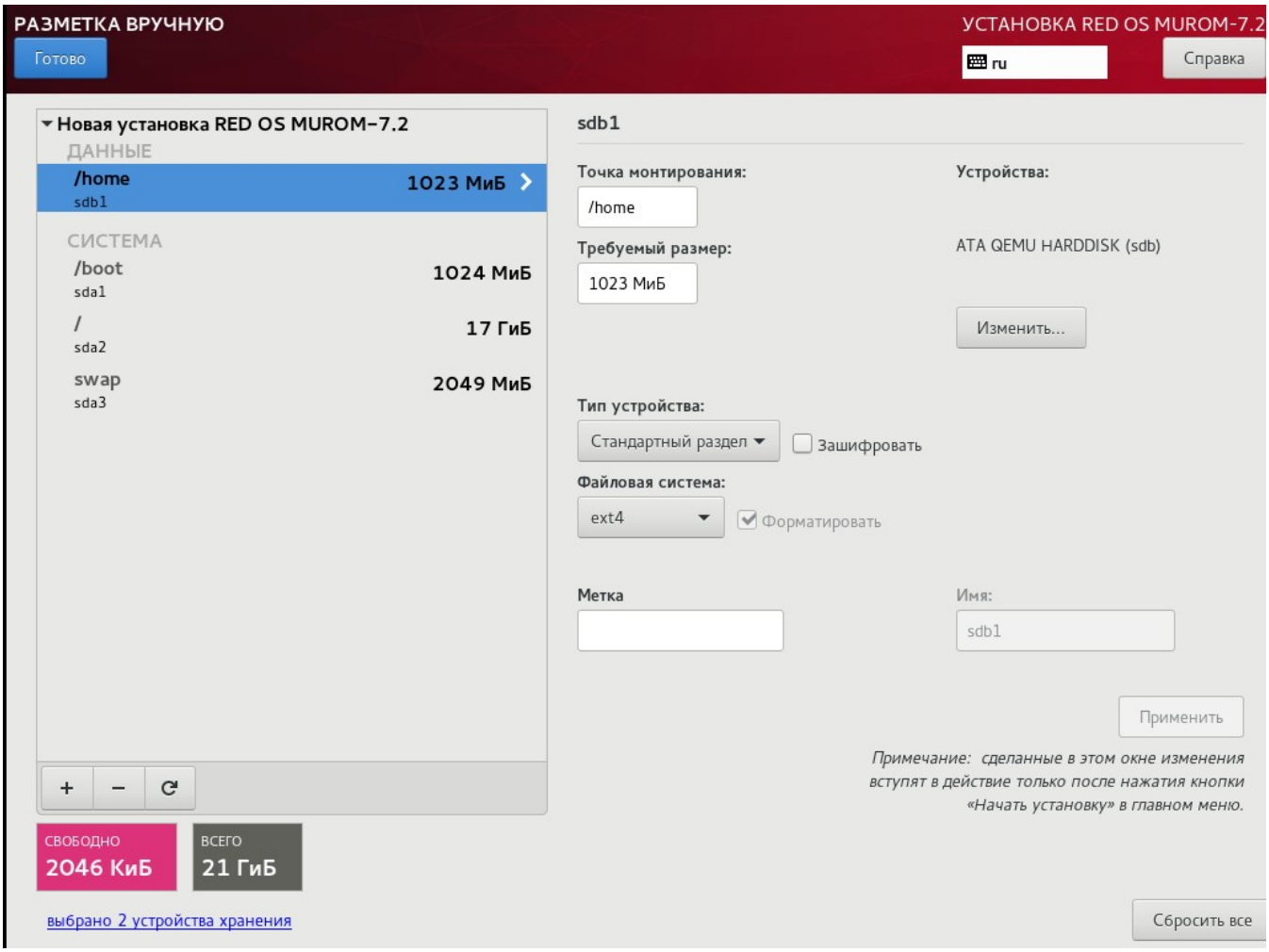


Рисунок 8 — Выбранные диски

Таким образом, под **/home** (домашний каталог профилей пользователей) был выделен отдельный физический диск sdb размером 1 Гб. Остальные разделы boot

(sda1), swap (sda3) и корень (sda2) мы разместили на другом физическом диске sda размером 20 Гб.

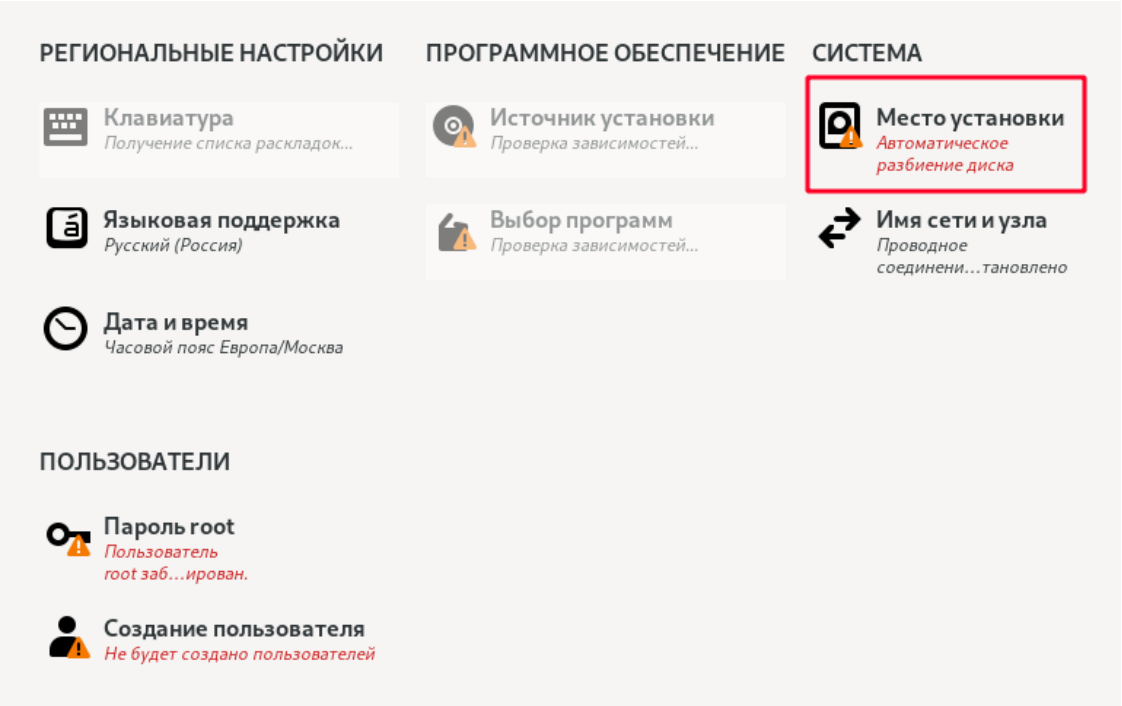


Нажимаем на кнопку «**Готово**» и приступаем к дальнейшим шагам установки РЕД ОС.

Сохранение раздела /home

Далее рассмотрим порядок настройки диска для сохранения (без изменения или удаления) домашнего раздела (**/home**) при переустановке РЕД ОС.

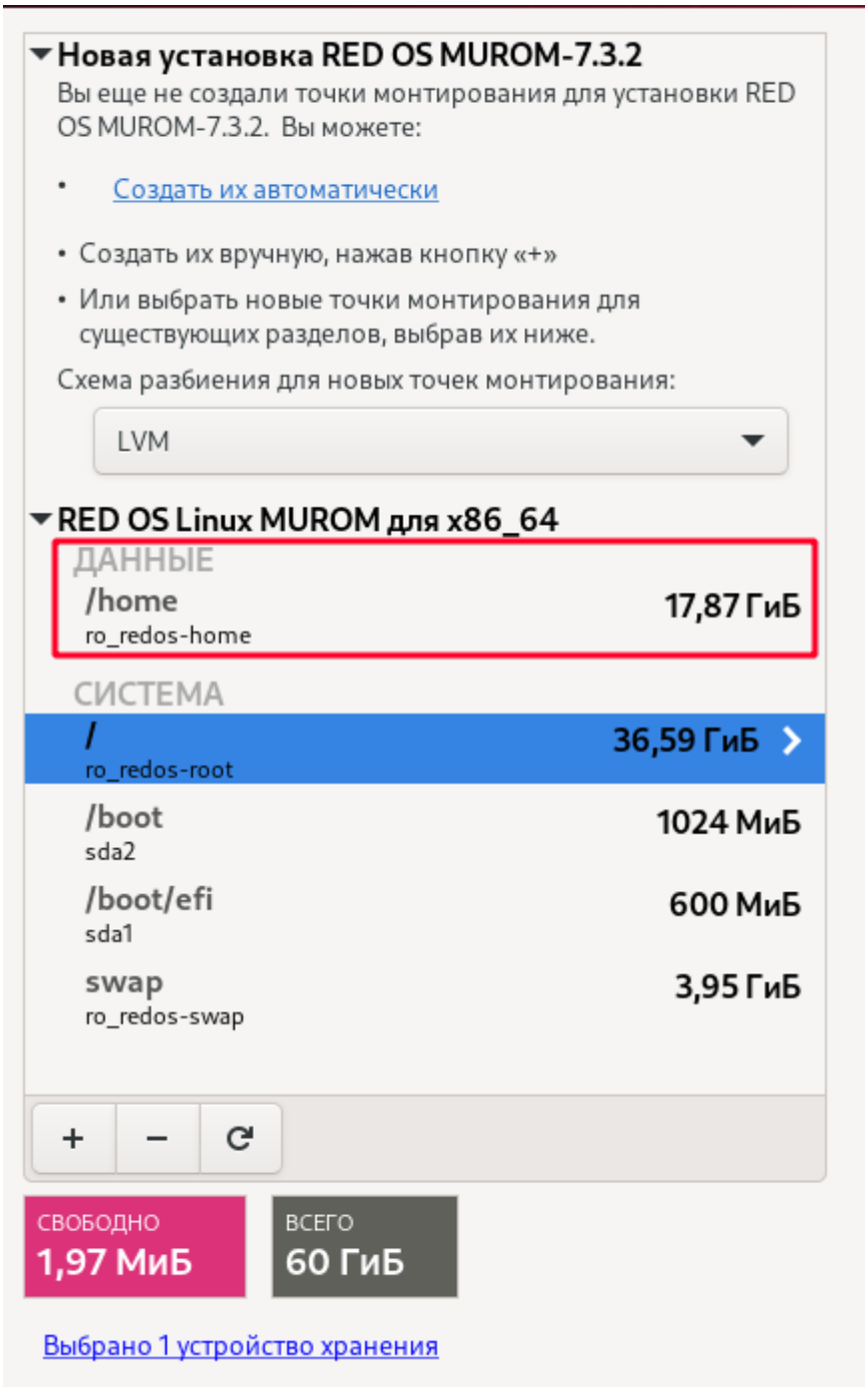
Запустите новую установку РЕД ОС, в мастере установки ОС перейдите в пункт «**Место установки**». Выберите конфигурацию для разметки диска «**По-своему**».



Убедитесь, что на диске присутствует отдельный раздел **/home**.

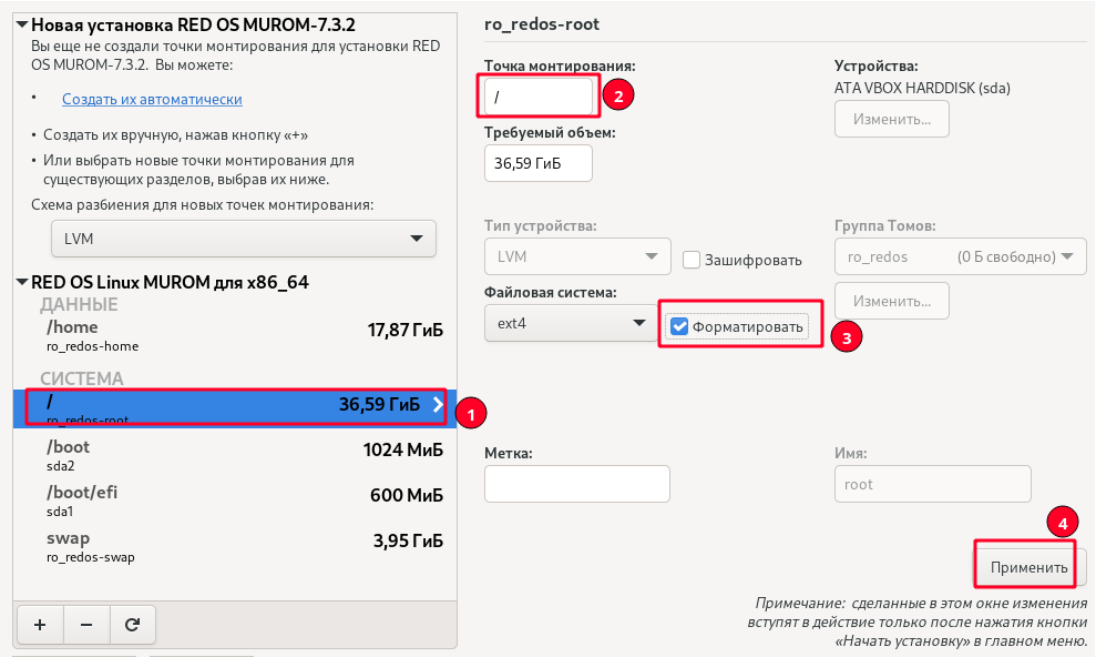
Примечание.

Данный раздел может отсутствовать, если ранее ОС установили с использованием автоматической разметки на жесткий диск, размер которого был менее 50 ГБ, в этом варианте домашний каталог (**/home**) размещается в корневом разделе (**/**), а не на отдельном. В результате при переустановке ОС корневой раздел будет неизбежно отформатирован вместе с домашним каталогом пользователя. В этом случае, перед установкой необходимо сделать резервную копию **/home** и после установки ОС восстановить данные.



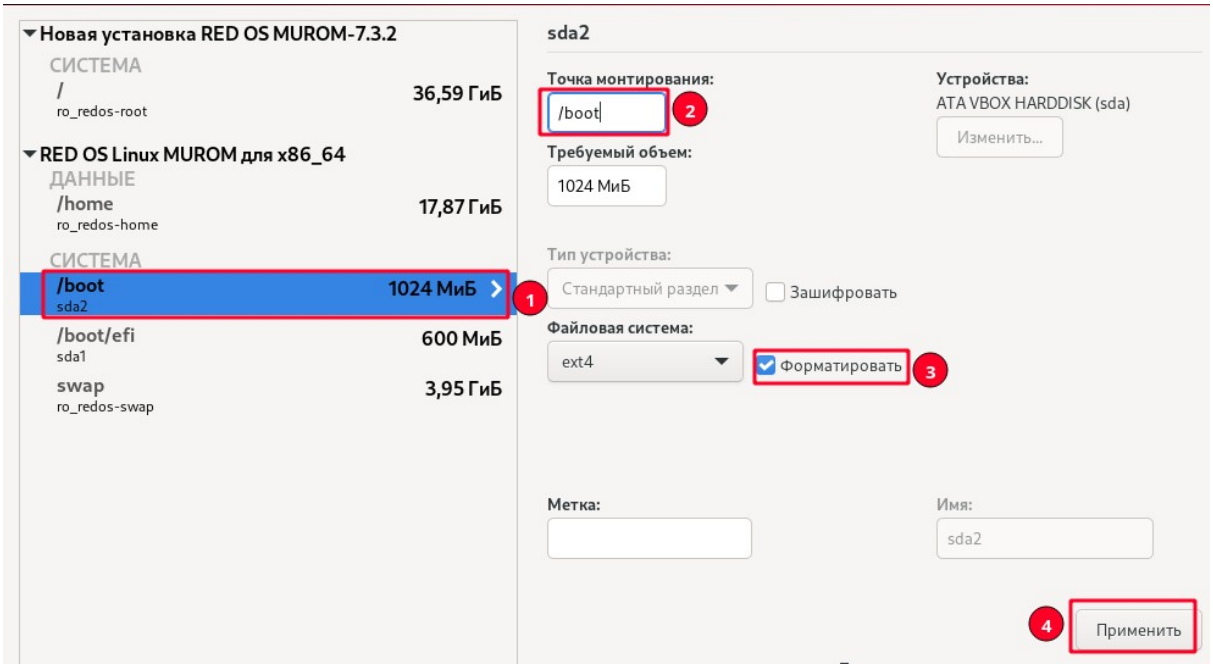
Именно этот раздел (**/home**), содержащий профили пользователей с их данными, **не будет отформатирован**. Остальные разделы необходимо настроить.

Для корневого раздела укажите точку монтирования «**/**», файловую систему **ext4**, а также установите флаг «**Форматировать**», далее нажмите кнопку «**Применить**».

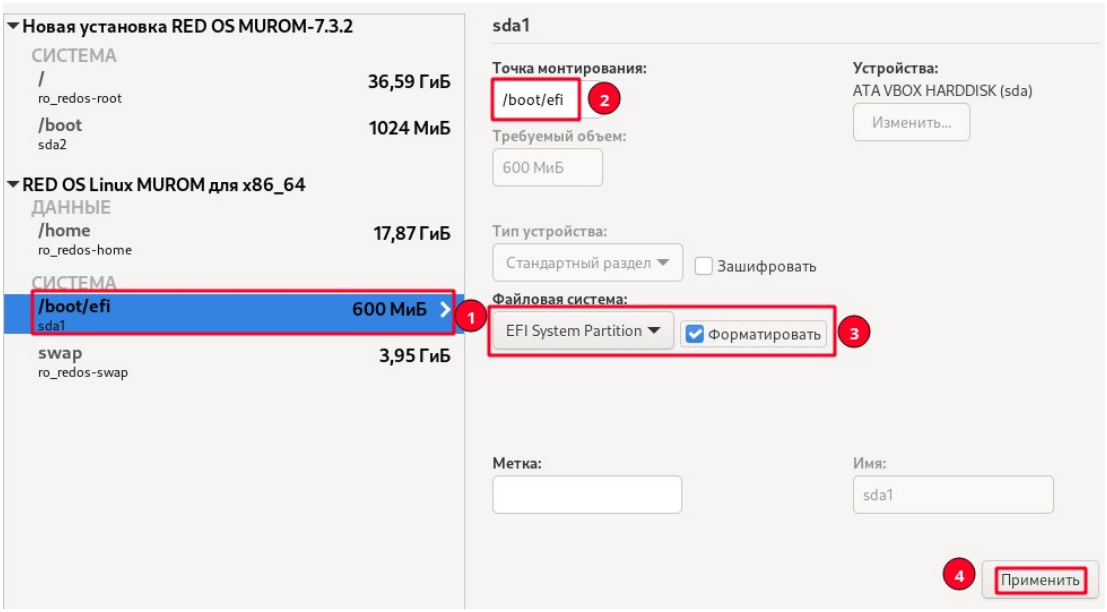


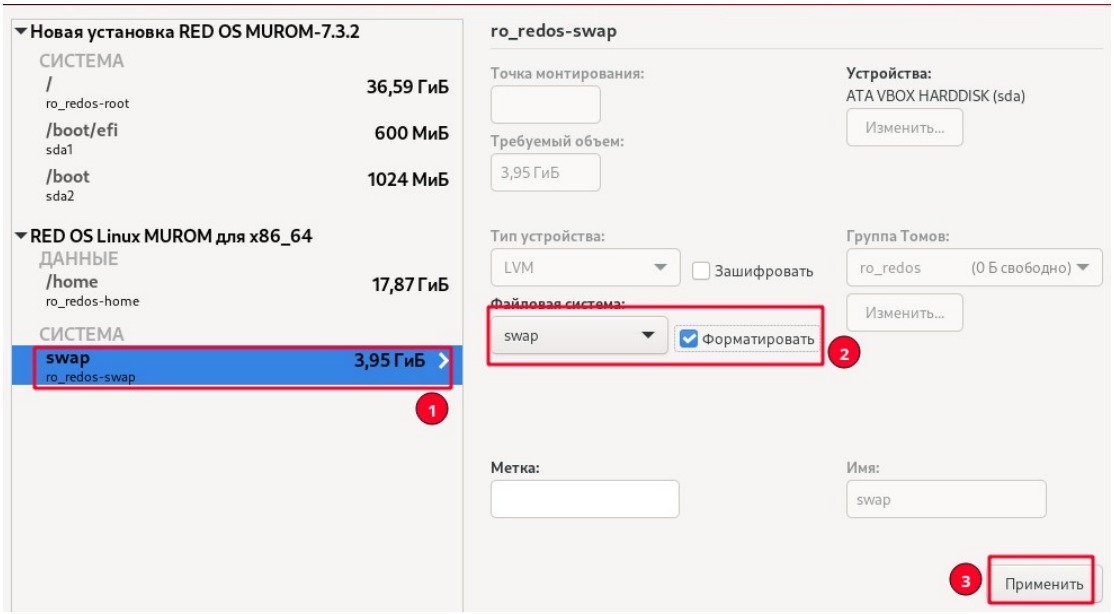
Результат изменения появится в разделе «**Новая установка RED OS MUROM**».

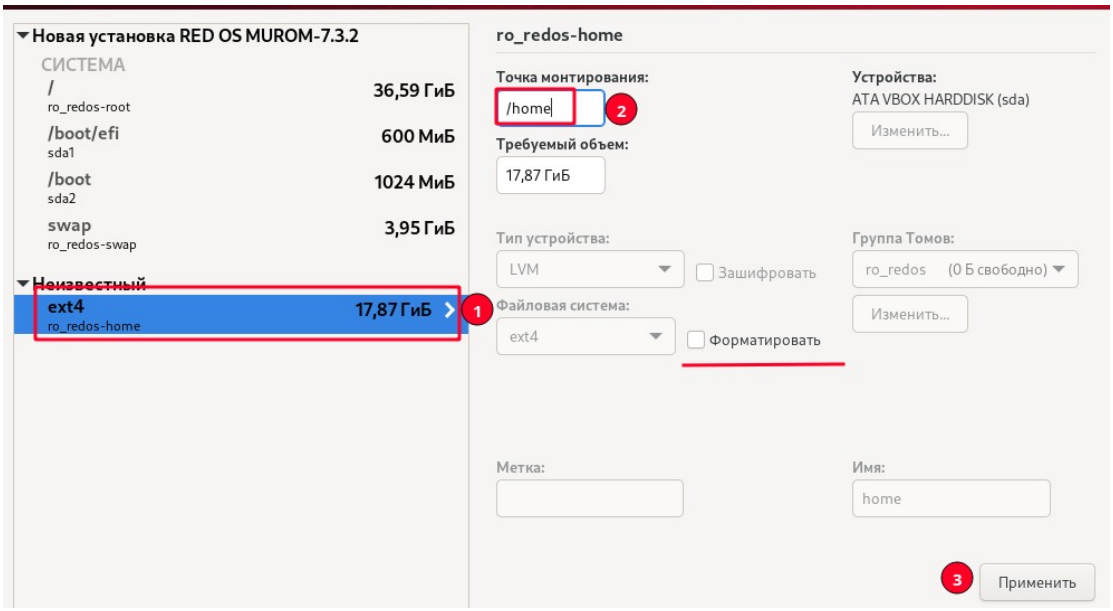
Следующим шагом выберите раздел **/boot** и присвойте ему точку монтирования **/boot**, файловую систему **ext4** и установите флаг «**Форматировать**», нажмите «**Применить**».



Аналогичные действия выполните для раздела **/boot/efi** и **swap** так, как это показано на нижеприведенных рисунках.

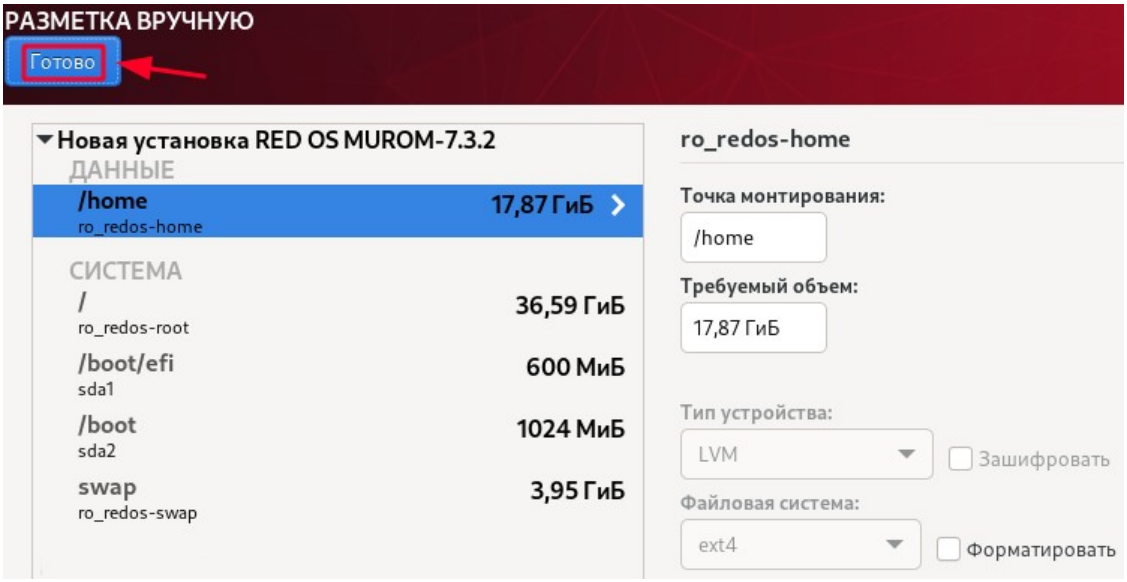

Новая установка RED OS MUROM-7.3.2
СИСТЕМА
/
ro_redos-root
36,59 ГиБ
/boot
sda2
1024 МиБ
RED OS Linux MUROM для x86_64
ДАННЫЕ
/home
ro_redos-home
17,87 ГиБ
СИСТЕМА
/boot/efi
sda1
600 МиБ
swap
ro_redos-swap
3,95 ГиБ
sda1
Точка монтирования:
/boot/efi
Требуемый объем:
600 МиБ
Устройства:
ATA VBOX HARDDISK (sda)
Изменить...
Тип устройства:
Стандартный раздел
Зашифровать
Файловая система:
EFI System Partition
Форматировать
Метка:
Имя:
sda1
Применить
1
2
3
4


Новая установка RED OS MUROM-7.3.2
СИСТЕМА
/
ro_redos-root
36,59 ГиБ
/boot/efi
sda1
600 МиБ
/boot
sda2
1024 МиБ
RED OS Linux MUROM для x86_64
ДАННЫЕ
/home
ro_redos-home
17,87 ГиБ
СИСТЕМА
swap
ro_redos-swap
3,95 ГиБ
ro_redos-swap
Точка монтирования:
Требуемый объем:
3,95 ГиБ
Устройства:
ATA VBOX HARDDISK (sda)
Изменить...
Тип устройства:
LVM
Зашифровать
Группа Томов:
ro_redos
(0 Б свободно)
Изменить...
Файловая система:
swap
Форматировать
Метка:
Имя:
swap
Применить
1
2
3

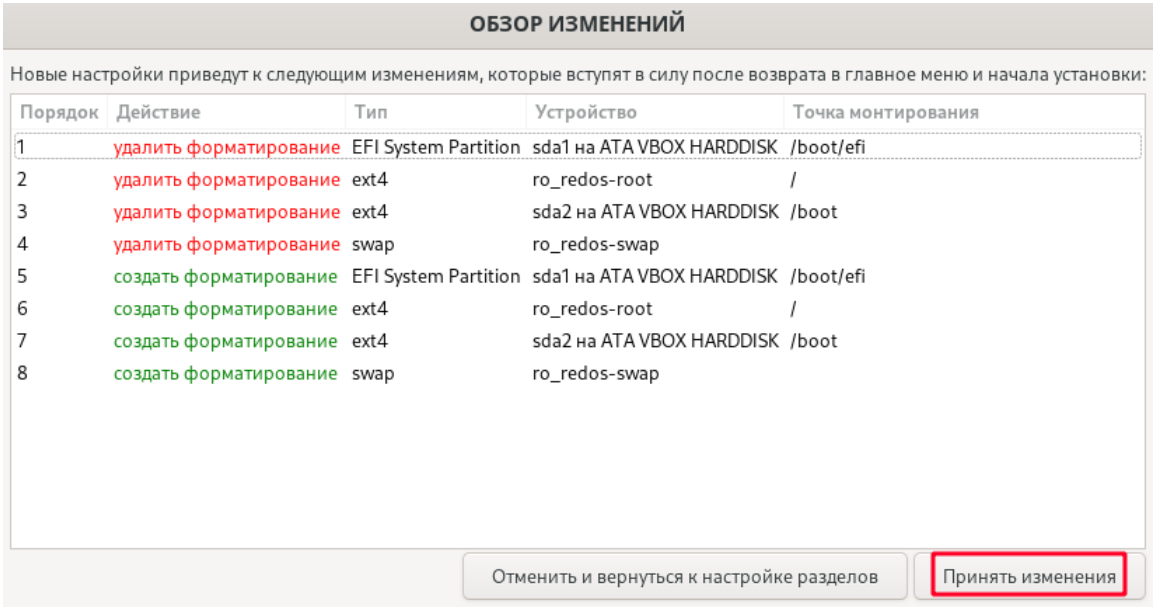


Готовая конфигурация разметки диска показана на рисунке ниже.

Обратите внимание, что для раздела **/home** не был установлен признак форматирования. Но для корневого раздела (/), а также для **/boot/efi**, **/boot** и **swap** признак «**Форматировать**» был присвоен, т.е. все данные перед установкой на этих разделах будет удалены. Нажмите на кнопку «**Готово**» для продолжения.



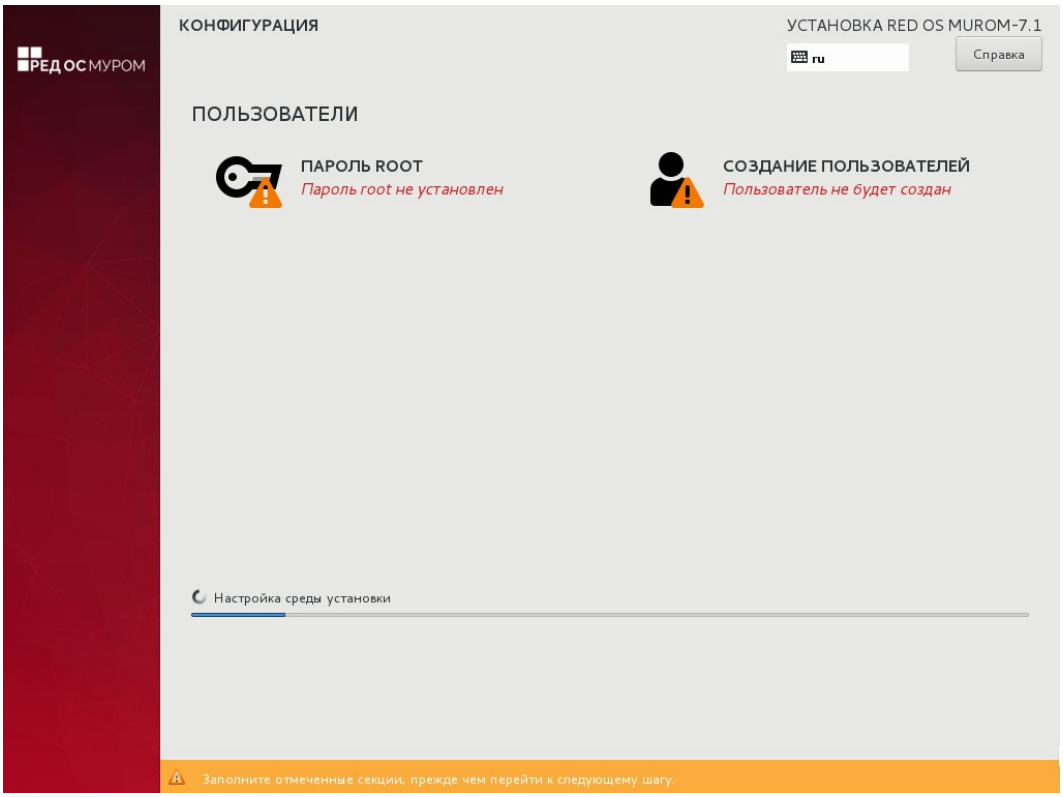
В результате появится итоговая таблица изменений, которые будут выполнены перед установкой ОС.



Примените изменения и продолжите настройку других параметров для установки РЕД ОС.

Задание пароля администратора системы

После того как предварительная настройка системы завершена, начинается непосредственная установка системы. Параллельно с установкой необходимо настроить пароль администратора **root** и, при необходимости, создать учетные записи пользователей.

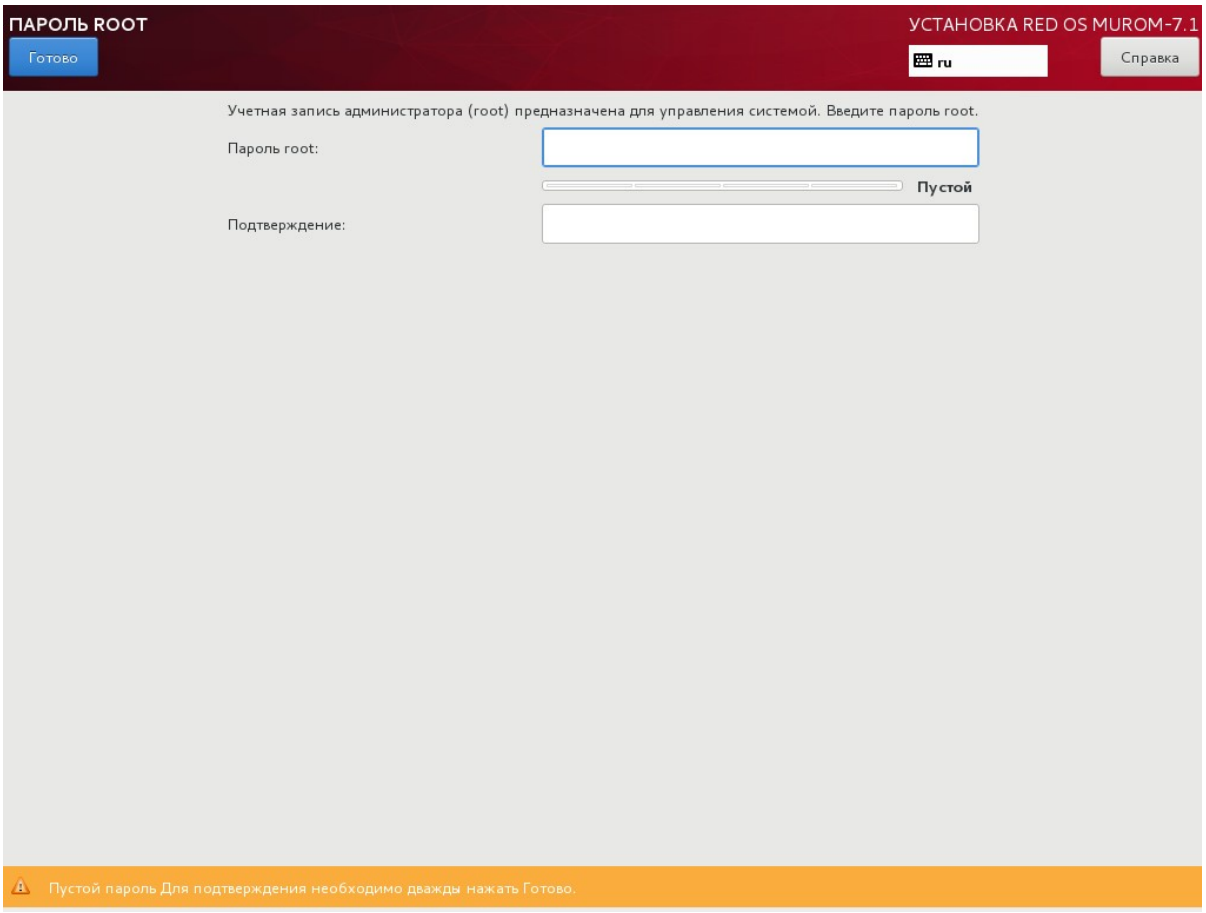


РЕД ОС - это многопользовательская операционная система. На практике это

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

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

Необходимо запомнить пароль **root** - его нужно будет вводить, чтобы получить право изменять настройки системы с помощью стандартных средств настройки РЕД ОС.



Ввод пароля защищен, при наборе пароля вместо символов на экране отображаются специальные символы. Чтобы избежать опечатки при вводе пароля, его предлагается ввести дважды. К введенному паролю в режиме реального времени применяется политика сложности пароля, т.е. производится его проверка, и при слишком простом пароле или совпадении пароля с парольной последовательностью из словаря паролей системой будет предложено произвести смену пароля администратора РЕД ОС.

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

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

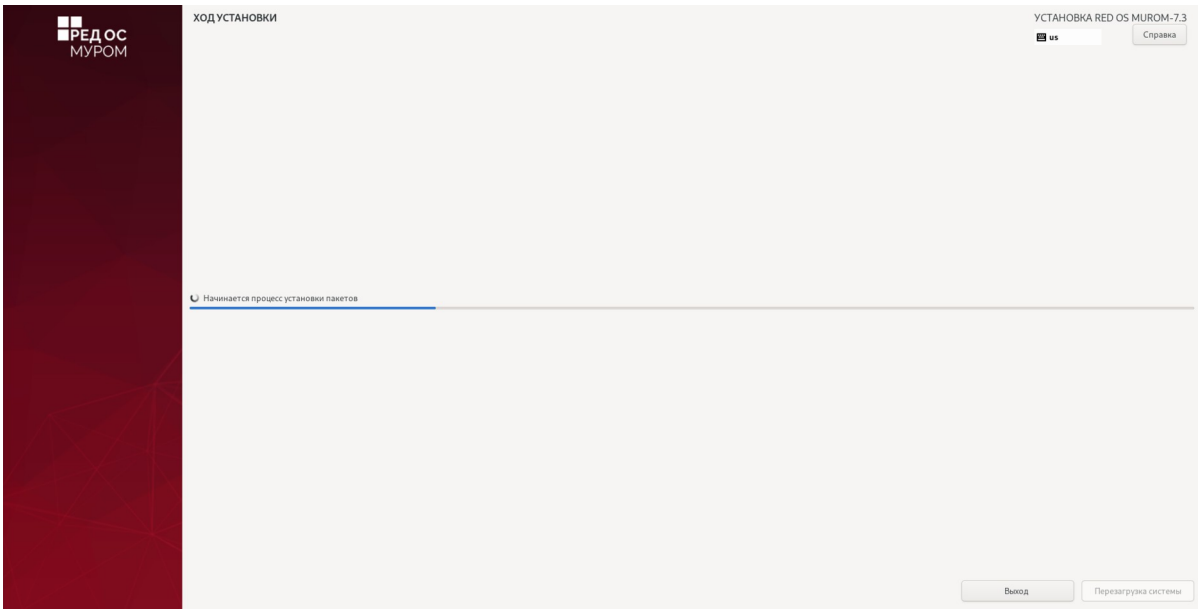
При добавлении пользователя предлагается ввести имя учётной записи (login name) пользователя. Имя учётной записи всегда представляет собой одно слово, состоящее только из строчных латинских букв (заглавные запрещены), цифр и символа подчёркивания «_» (причём цифра и символ «_» не могут стоять в начале слова). Чтобы исключить опечатки, пароль пользователя вводится дважды. Так же, как при выборе пароля администратора РЕД ОС (**root**), действуют требования по сложности пароля.

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

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

Установка системы

Этап установки представляет собой установку набора программ, необходимых для работы системы.



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

Начиная с этого шага, программа установки работает с файлами только что установленной базовой системы. Все последующие изменения можно будет совершить после завершения установки посредством редактирования соответствующих конфигурационных файлов или при помощи модулей управления, включённых в дистрибутив.

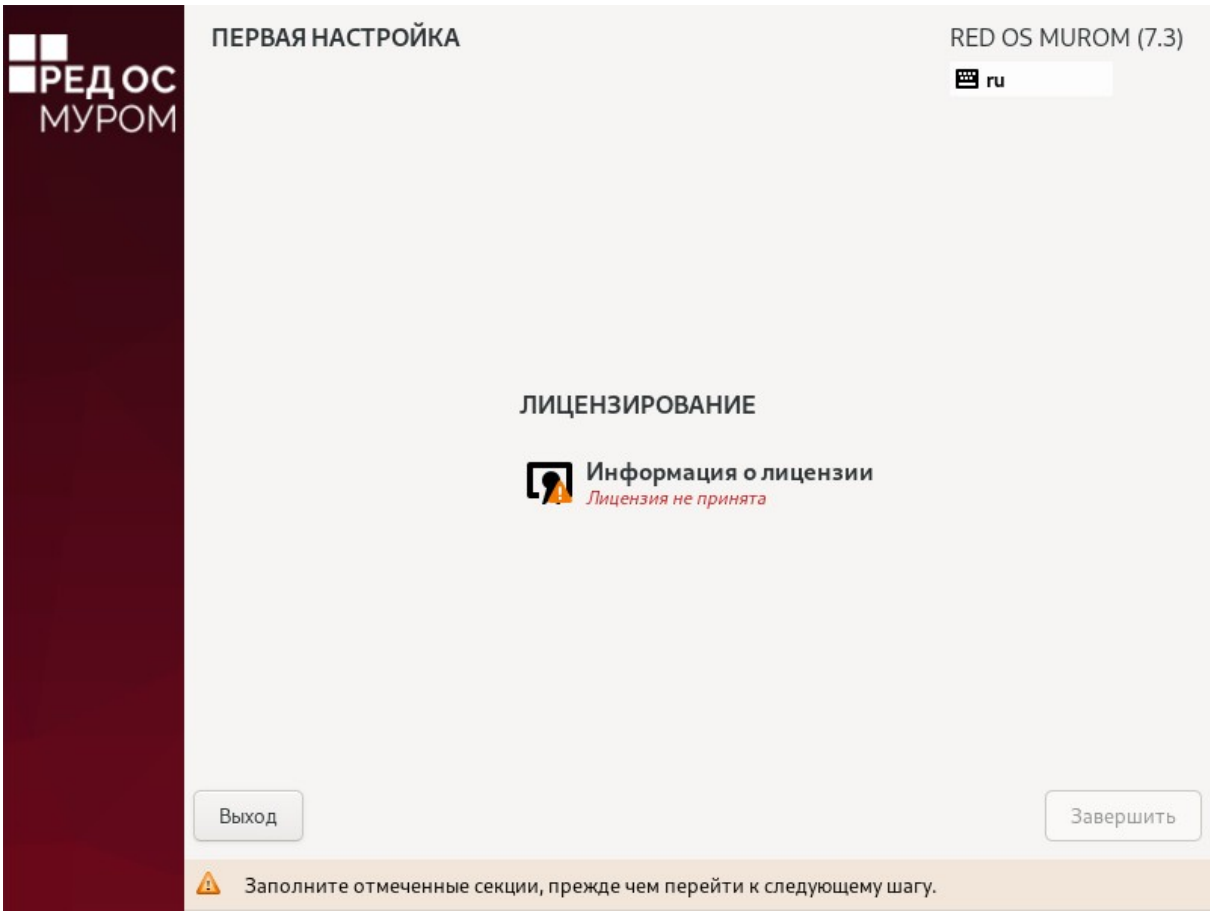
Загрузчик Linux — программа, которая позволяет загружать Linux и другие операционные системы. При автоматическом разбиении разделов накопителя на жестких дисках средства вычислительной техники загрузчик автоматически устанавливается в начальный раздел диска.

Если же планируется использовать и другие операционные системы, уже установленные на этом компьютере, тогда имеет значение, на каком жестком диске или разделе будет расположен загрузчик. В большинстве случаев программа установки правильно подберет расположение загрузчика.

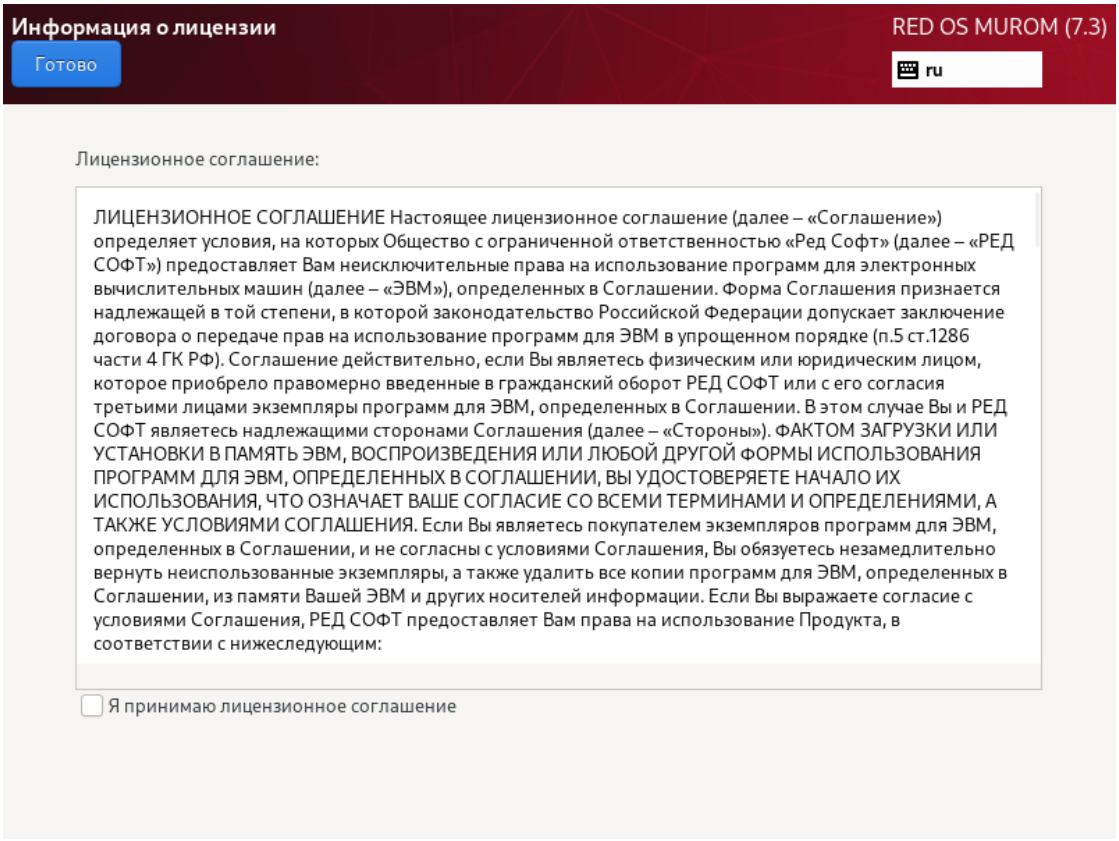
После выполнения копирования файлов РЕД ОС и установки загрузчика пользователю предлагается произвести перезагрузку РЕД ОС кнопкой «**Перезагрузить**».

Соглашение пользователя и Лицензионный договор

После успешной перезагрузки и перед продолжением установки следует внимательно прочитать условия пользовательского соглашения.



В лицензии говорится о правах распространения и гарантиях производителя.



При приобретении дистрибутива данное лицензионное соглашение прилагается в печатном виде к копии дистрибутива на вкладыше комплекта дистрибутива. Лицензия относится ко всему дистрибутиву РЕД ОС.

Завершение установки

После сохранения настроек и перезапуска системы пользователю предоставляется экран приветствия и приглашения к авторизации. На этом установка и настройка РЕД ОС завершена, и РЕД ОС полностью готова к использованию.

Действия после установки ОС

После установки РЕД ОС не все функции безопасности включены по умолчанию. Администратор должен определить функциональную роль установленной копии ОС, перечень решаемых задач, и, в соответствии с действующими политиками безопасности организации, произвести активацию необходимых служб, сервисов и приложений. А именно:

сконфигурировать перечень отслеживаемых событий безопасности;

настроить действия в случае обнаружения критически важных событий безопасности;

настроить действия для предотвращения потери данных аудита;

настроить защищенную передачу данных аудита;

создать необходимое количество пользователей, групп, ролей и связать пользователей с группами и ролями, исходя из необходимых им полномочий;

установить права доступа на создаваемые вновь каталоги и файлы;

определить правила установки программ и правила запуска компонентов программного обеспечения;

определить и настроить используемые механизмы идентификации и аутентификации, сроки действия учетных записей и паролей, требования, предъявляемые к аутентификационным данным;

создать необходимые резервные копии данных и системных настроек и проводить периодическое их тестирование;

настроить доверенные сервера точного времени;

проверить целостность важных данных и системных файлов;

организовать отказоустойчивый кластер при необходимости обеспечения повышенной отказоустойчивости;

определить квоты и приоритеты, выделяемые пользователям;

установить необходимые ограничения на сеансы пользователей;

установить продолжительность бездействия пользовательского сеанса, после которого сеанс будет заблокирован или завершен.

Модуль 3. Основы работы в командной оболочке РЕД ОС .

GNU/Linux является многопользовательской операционной системой, в которой одновременно может работать несколько различных пользователей.

Для каждого пользователя, имеющего право на работу с данной системой, системным администратором создается так называемая *учетная запись (account)*.

Примечание: Говорят, что администратор регистрирует пользователей в системе.

Имя пользователя идентифицирует пользователя в системе, а пароль предназначен для исключения несанкционированного входа в сеанс под именем этого пользователя.

Характерное приглашение на ввод имени пользователя для входа в сеанс при использовании текстовой оболочки выглядит так:

GNU/Linux on TTY2 login:

После приглашения `login:` следует ввести имя пользователя, зарегистрированного в системе. Ввод следует завершить нажатием на клавишу Enter.

После этого на экран будет выведено приглашение ввести пароль:

password:

Ввод пароля не сопровождается отображением вводимых символов из соображений безопасности.

Если при вводе пароля была допущена ошибка, то можно воспользоваться клавишей Back Space (забой) для исправления неверно введенного символа.

Если при вводе пароля все-же будет допущена ошибка, то на экране будет отображено сообщение `Login incorrect`

Пример:

GNU/Linux on TTY2

login:student password:

login incorrect

Примечание: В этом примере пользователь `student` попытался войти в сеанс, однако ввел неверный пароль. Поэтому система выдала сообщение `login incorrect`, и вход в сеанс осуществлен не был.

Система Linux может быть настроена так, что после нескольких неудачных попыток входа терминал может быть заблокирован.

Для выхода из сеанса необходимо набрать команду `exit` . Можно также использовать команду `logout` .

Удобно также использовать сочетание клавиш `C^D`, однако оно может быть заблокировано с помощью специальной настройки оболочки `bash` (блокировка достигается командой `set -o ignoreeof`).

После выхода из сеанса на экране вновь появится приглашение ввести имя пользователя

login: .

Как вводить команды в оболочке?

Оболочка предоставляет интерфейс командной строки, в котором управление операционной системой и запуск программ осуществляется с помощью команд в текстовом виде.

Команды вводятся с учетом регистра символов.

Ввод команд осуществляется с клавиатуры. Команда запускается на исполнение нажатием на клавишу Enter. Однако вместо этого можно пользоваться сочетаниями клавиш `C^J` или `C^M`.

Для удобства пользователей на экран выводится приглашение для ввода команд. Оно называется приглашением командной строки.

Пример: Типичный его вид следующий:

[user@host etc]$

Примечание: В этом примере в приглашении командной строки выводится имя пользователя user, имя хоста host и имя текущего каталога etc.

Вид приглашения командной строки легко изменить с помощью переменной окружения

`PS1`, однако следует придерживаться следующего общепринятого правила:

Приглашение должно заканчиваться символом доллара `$` или реже знаком больше > если это сеанс простого пользователя.

Если в сеанс зашел суперпользователь, то приглашение командной строки заканчивается символом решетки `#`.

Пример: Ниже приведен пример типичного приглашения строки сеанса суперпользователя:

```
[root@host root]#
```

Если команда введена неверно, то система выводит соответствующее сообщение об ошибке.

Пример: если команда `who` введена ошибочно, то на экране будет выведено:

```
$ hwo
bash: hwo: command not found
```

Примечание: Оболочка сообщает, что команда `hwo` не найдена. Это сообщение выведено вследствие ошибочного ввода команды `who`.

Очистить экран можно с помощью команды `clear`.

Что такое оболочка?

Командная оболочка (shell) – это программа, взаимодействующая с пользователем с помощью интерфейса командной строки и позволяющая пользователю запускать прикладные программы и выполнять различные команды операционной системы.

Оболочка интерпретирует введенные пользователем команды, и преобразует их в инструкции операционной системы.

Показывая пользователю, что оболочка готова интерпретировать команды, она выводит специальное приглашение командной строки, заканчивающееся обычно символом доллара `$` в сеансе обычного пользователя.

В сеансе суперпользователя оболочка обычно в качестве приглашения использует символ решетки #, предупреждая о возможности нарушения работоспособности всей системы вследствие ошибочных действий.

В GNU/Linux может быть использовано множество различных оболочек, однако стандартом de facto является оболочка Bourne Again Shell – `bash`.

Оболочка запускается при входе пользователя в сеанс.

Какая конкретно оболочка будет запущена определяется учетной записью пользователя. Определить, какая оболочка установлена в учетной записи пользователя, можно путем вывода переменной `SHELL`:

Пример:

```
$ echo $SHELL
/bin/bash
```

Примечание: Переменная окружения `SHELL` содержит в себе полное имя исполняемого файла оболочки пользователя, используемой при входе в сеанс этого пользователя. В данном случае – это Bash. Символ доллара `$` перед переменной окружения используется для извлечения ее значения.

Команды пользователя представляют собой строки, вводимые с клавиатуры.

После того, как команда введена и нажата клавиша `Enter`, команда интерпретируется оболочкой и, при удачной интерпретации, исполняется.

Если команда введена синтаксически неверно, то выдается сообщение об ошибке. Помимо отдельных команд, вводимых последовательно с клавиатуры, в командной оболочке можно также использовать файлы сценариев.

Сценарии позволяют выполнять достаточно сложные задачи с использованием условных переходов, циклов и подпрограмм.

Наиболее распространенные оболочки в GNU/Linux.

В GNU/Linux можно использовать множество различных оболочек, а также можно написать свою собственную оболочку, если существующие варианты не удовлетворяют имеющихся требований.

Наиболее распространены четыре вида оболочек:

- Bash shell (`bash`) – используется по умолчанию;
- Public domain Korn shell (`pdksh` или `ksh`);
- Enhanced C shell (`tcsh`);
- Z Shell (`zsh`).

Примечание: В скобках указаны команды для запуска этих оболочек (имена исполняемых файлов оболочек).

Различные оболочки обладают различным набором функций и даже различными встроенными командами.

Встроенные команды оболочки – это команды, которые реализованы внутри нее, а не в виде внешних программ.

При запуске скриптов необходимо убеждаться, что скрипт предназначен для выполнения в данной оболочке.

Bash shell является лидером по популярности и большинство пользователей GNU/Linux используют именно эту оболочку.

Bash устанавливается по умолчанию для пользователей GNU/Linux и именно она обычно загружается после входа в сеанс.

Оболочка Bash позволяет настраивать вид приглашения, однако чаще всего, обычные пользователи видят приглашение, заканчивающееся символом `$`, а для суперпользователя (root) приглашение заканчивается символом `#`.

Примечание: Это сделано для лишнего напоминания суперпользователю о требуемой при его работе осторожности.

В системе может быть установлено множество оболочек. Для получения их списка можно воспользоваться командой `chsh -l` , если она установлена в системе и настройки безопасности позволяют ей воспользоваться.

Можно также просто просмотреть содержимое файла `/etc/shells`, содержащего в себе список установленных в системе оболочек.

Пример: Ниже приведено типичное содержимое этого файла:

```
# cat /etc/shells
/bin/sh
/bin/bash
/bin/csh
/bin/tcsh
/bin/ksh
/bin/zsh
```

Примечание: В этом примере с помощью команды `cat` получено содержимое файла `/etc/shells`, содержащего список оболочек в системе.

Примечание: Следует еще раз отметить, что в GNU/Linux, вовсе не обязательно, чтобы оболочка, указанная в файле `/etc/shells` была на самом деле установлена в системе. Этот файл более необходим для программ, обеспечивающих удаленный доступ к системе. Например, большинство программ – серверов FTP запрещают входить в сеанс пользователям, оболочка по умолчанию которых не указана в данном файле.

Для временной загрузки другой оболочки необходимо просто набрать имя соответствующего исполняемого файла.

Пример: для запуска оболочки Enhanced C shell выполняем команду:

```
$ tcsh
$ ps
PID TTY     TIME CMD
2349 pts/0   00:00:00 bash
10295 pts/0 00:00:00 tcsh
10319 pts/0 00:00:00 ps
```

Примечание: Эта команда запустит оболочку Enhanced C shell. Далее с помощью команды `ps`*демонстрируется, что оболочка* `tcsh` *была запущена из* `bash`*.*

Временно загружать оболочку иногда требуется для того, чтобы выполнить сценарий командной строки, предназначенный для этой оболочки.

Для выхода из временно загруженной оболочки достаточно набрать команду `exit` . 14.Если системная политика это позволяет, то пользователь может изменить для себя оболочку, которая будет запускаться при его входе в сеанс. Это можно сделать с помощью команды `chsh –s <shell>` , где `<shell>` - имя исполняемого файла оболочки

Пример:

```
$ chsh –s /bin/csh
```

Примечание: Здесь установлена оболочка, которая будет загружаться по умолчанию, C shell, исполняемый файл которой `/bin/csh`*. При этом, всякий раз, когда пользователь будет входить в сеанс, будет загружена именно эта оболочка.*

Структура командной строки.

Пользователь вводит команды с клавиатуры, и они отображаются после приглашения командной строки.

Обычно на экране оболочкой отображается курсор, показывающий позицию вывода следующего символа, вводимого с клавиатуры.

Для того, чтобы команда была правильно интерпретирована оболочкой и команда правильно обработала переданные ей аргументы, следует придерживаться соглашений о структуре командной строки. В общем виде командная строка состоит из следующих трех частей:

Имя команды – соответствует имени исполняемого файла системной команды или же встроенной команды оболочки.

Опции – дополнительные инструкции, сообщающие команде детали действий, которые она должна выполнить.

Аргументы – объекты, над которыми команда должна произвести заданные действия.

Примечание: То есть, образно говоря, команда сообщает операционной системе что надо сделать, опции уточняют как эти действия должны быть произведены, а аргументы – это то, над чем эти действия будут произведены.

Команды, вводимые в оболочке представляют собой строки, причем команда, опции и аргументы должны быть отделены друг от друга пробелами или табуляцией.

Во многих командах в качестве аргументов используются имена файлов. Обобщающее название таких команд - “файловые команды”.

Существует три основных формата командной строки, поддерживаемых GNU/Linux. Их основное отличие – стиль указания опций.

В формате UNIX98 (иначе - POSIX формат) опции указывают в виде одиночных букв, перед которыми ставится символ – (тире): команда -опции аргументы

Формат UNIX98 краток и удобен для команд с большим набором опций, так как опции чаще всего можно указывать друг за другом

Пример:

```
$ ls -dl /etc/default
```

Примечание: В этом примере команда `ls`, которая обычно выводит содержимое каталога, указанного в качестве аргумента, ведет себя иначе, так как используются опции `-d` и `-l`. Опция `-d` заставляет команду `ls` выводить информацию о самом каталоге, а не о файлах в нем. Опция `-l` сообщает команде, что вывод должен быть осуществлен в подробном формате.

В BSD формате тире перед опциями отсутствует команда опции аргументы

В формате BSD также можно указывать несколько опций подряд

Пример:

```
$ ps aux
```

Примечание: Команда `ps` выводит список процессов в системе. Три используемые опции – `a`, `u`, `x` модифицируют поведение команды так, что она отображает список всех процессов в системе, указывая пользователей, от имени которых запущены эти процессы.

Третий используемый в Linux формат командной строки – длинная нотация GNU. В этом формате опция записывается целым словом, перед которым надо указать двойное тире

команда --опция1 --опция2 аргументы

Удобство этого формата состоит в интуитивной ясности опций, поскольку они записываются целыми словами.

Команды GNU поддерживают специальную опцию `--help`,

обеспечивающую возможность получения краткой справки по команде

Пример:

$ gzip --help

Примечание: Команда `gzip` позволяет сжимать файлы. Однако, в данном случае она просто выводит информацию о себе, так как установлена опция --`help`.

Некоторые команды используют опции, не относящиеся ни к одному из перечисленных форматов

Примечание: Так встречаются опции вида `+5` и подобные. Опции, используемые в системе XFree86 (свободно распространяемая реализация графической системы X Window), указываются после знака тире

- , но являются "длинными", хотя и не в стиле GNU. Например:

$ xterm –display :0.0

Эта команда запускает графический эмулятор командной строки – программу xterm. Легко заметить, что в качестве опции используется целое слово, однако опция отмечена лишь одним тире - .

Многие команды позволяют использовать различные форматы.

Пример: Команды, показанные ниже, делают одно и то же – выводят информацию о подкаталоге `mydir` текущего каталога:

$ls –d mydir

$ls --directory mydir

Примечание: Обе команды сделают одно и то же, поскольку у команды `ls`, опции `-d` и `--directory` эквивалентны и устанавливают вывод информации о самом каталоге, вместо вывода информации о его содержимом.

После ввода команды необходимо нажать клавишу `Enter`, после чего интерпретатор командной строки производит синтаксический анализ, разбирая командную строку на части: имя команды, список опций и список аргументов.

Если разборка строки закончилась удачей, производится попытка исполнить команду.

Встроенные и системные команды.

Все команды GNU/Linux делятся на два больших класса: встроенные и системные.

Встроенные команды интерпретируются и выполняются самой оболочкой.

Системные команды представляют собой исполняемые файлы, находящиеся

в специальных каталогах.

Встроенные команды представляют собой процедуры оболочки и, следовательно, выполняются быстрее системных команд.

В силу того, что они привязаны к конкретной оболочке, необходимо понимать, что одинаковые команды в разных оболочках могут вести себя по-разному.

Примечание: например, команда `set` в Bash и C shell используется совершенно по-разному.

Основная масса встроенных команд, все – таки, одинакова в различных оболочках.

Командная оболочка `bash`, например, предоставляет пользователю такие встроенные команды, как `cd`, `alias`, `bg`, `kill`, `pwd` и `echo`.

Исполняемые файлы системных команд обычно находятся в одном из каталогов, указанных ниже:

/bin

/sbin

/usr/bin

/usr/sbin

/usr/local/bin

/usr/local/sbin

Пользователь может написать свои собственные системные команды и использовать их.

Обычно для размещения таких команд используется домашний каталог пользователя или его подкаталог `bin`.

Если в системе имеется одновременно и встроенная и системная версия какой-либо команды, то если команда вызвана без указания пути к ней, то выполняется встроенная команда.

Если при вызове команды указан путь, то выполняется системная команда.

Ввод, редактирование и исполнение команд.

При работе в командной строке можно использовать привычные клавиши управления курсором и клавиши редактирования.

Однако, во-первых, не все виды клавиатур обеспечивают такие клавиши, как `Home`, а, во- вторых, во многих случаях привычные клавиши управления курсором

не работают.

Ниже приведена таблица клавиатурных сочетаний, которые могут быть использованы при работе в командной строке.

Клавиши	Действие
Ctrl-B	Курсор влево
Ctrl-F	Курсор вправо
Alt-B	Курсор на слово влево
Alt-F	Курсор на слово вправо
Ctrl-A	Курсор в начало строки
Ctrl-E	Курсор в конец строки
Ctrl-H	Удаление символа перед курсором
Ctrl-D	Удаление символа в позиции курсора
Alt-D	Удаление слова
Ctrl-U	Удаление части строки слева от курсора
Ctrl-K	Удаление части строки справа от курсора
Ctrl-M или Ctrl-J	Ввод
Ctrl-V	Отмена специального значения символа
Ctrl-L	Очистка экрана
Alt-T	Перемена мест аргументов
Alt-L	Перевод слова в нижний регистр
Alt-U	Перевод слова в верхний регистр
Ctrl-C	Остановка выполнения задания
Ctrl-Z	Приостановка выполнения задания

Иногда бывает необходимо ввести в командную строку символический код клавиатурного сочетания вместо выполнения команды, связанной с этим сочетанием. В этом случае перед вводом клавиатурного сочетания, имеющего специальное значение, необходимо ввести **`Ctrl-V`**

Пример:

```
$ echo "1    2        3"
```

1 2 3

Примечание: Обратите внимание на то, что в предыдущем примере в качестве аргумента команды `echo` использована строка, в которой имеются символы табуляции. Для ввода такой строки необходимо перед каждым символом табуляции нажать сочетание `Ctrl-V`, так как в противном случае Bash ошибочно воспринимает символ табуляции как специальный символ.

Нажатие **`Ctrl-C`** приводит к передаче процессу сигнала **`INT`**, что эквивалентно команде

```
kill -2 .
```

Однако не все задания могут быть остановлены так.

Нажатие **`Ctrl-Z`** приводит к приостановке активного задания, которое затем может быть переведено в фоновый режим командой `bg`, либо это задание может быть уничтожено командой `kill`.

Оболочка Bash предоставляет специальную возможность исправить ошибку, допущенную при вводе команды. Исправление достигается изменением цепочки символов в выполненной ранее команде и выполнением новой измененной команды. Для этого надо набрать: `^заменяемое^замена` и нажать `Enter` при этом будет выполнена команда с замененной подстрокой

Пример:

```
$ ls /dmb
ls: /dmb: No such file or directory
$ ^dmb^tmp ls /tmp
0103740143   0793455464   1645210352   AcromIO5pU
```

Примечание: Команде `ls` в этом примере был указан в качестве аргумента несуществующий каталог, поэтому было получено сообщение об ошибке. Далее, пользуясь механизмом замены символов, подстрока `dmb` была заменена на `tmp`, команда была исполнена автоматически и на экран было выведено содержимое каталога `/tmp`.

В случае если необходимо ввести длинную команду, которая не помещается в одну строку, необходимо воспользоваться символом обратной косой черты `\` и продолжить ввод на следующей строке

Пример:

```
$ find . \
```

-name "*prim*" \

-user 501 \

-type f \

-ls

1352836 20 -rw-r--r-- 1 user user 16408 Окт 14 1999 ./Documents/DocBook/html/primaryie.html

1353135 16 -rw-r--r-- 1 user user 16093 Окт 14 1999 ./Documents/DocBook/html/primary.html

Примечание: Команда `find . -name "*prim*" -user 501 -type f -ls` , выполненная здесь, довольно длинная. Она ищет все обычные файлы в текущем каталоге, в имени которых встречается строка `prim`, принадлежащие пользователю с UID=501, и печатает найденные имена файлов в формате, подобном `ls -l`. Так как команда длинная, то ее удобно ввести по строкам, разделяя каждую строку, входящую в команду, с помощью символа обратной косой черты. Знаки больше, показанные в листинге, являются вторичным приглашением командной строки. Они могут быть установлены с помощью переменной окружения `PS2`.

Можно вводить несколько команд в одной строке, разделяя их символом точка с запятой

;

Пример:

$ cd /opt; ls -l; cd; pwd итого 8

drwxr-xr-x 7 root root 4096 Июл 23 08:21 drweb

drwxr-xr-x 7 oracle oinstall 4096 Июл 23 07:20 oracle

/home/user

Примечание: В командной строке, приведенной выше, объединено сразу четыре команды, разделенные точкой с запятой.

Если команды отделены друг от друга с помощью двух амперсандов `&&`, то вторая команда будет выполнена только в случае успешного выполнения первой. То есть, в случае, когда первая команда вернула нулевой код возврата.

Пример:

$ ls nofile && cat nofile

ls: nofile: No such file or directory

$ ls /etc/motd && cat /etc/motd

/etc/motd

It's time to work!

Примечание: Первая команда `ls nofile` возвратила ненулевой код возврата, так как такого файла не оказалось, поэтому вторая команда не была выполнена. Во втором случае первая команда закончилась удачно, поэтому была выполнена вторая команда, отобразившая на экране содержимое файла `/etc/motd`.

При необходимости выполнять вторую команду только в случае неудачи первой следует использовать две вертикальные черты `||`

Пример:

```
$ ls nofile || echo "Net EGO"
ls: nofile: No such file or directory Net EGO
```

Примечание: Данный пример демонстрирует, что вторая команда в цепочке `echo "Net EGO"` была выполнена, так как первая команда `ls nofile` такой файл не обнаружила и закончилась с ошибкой.

История команд.

Оболочка Bash предоставляет пользователю возможность выполнять ранее введенные команды.

Строки введенных пользователями команд сохраняются в файлах `~/.bash_history` (в переменной окружения `HISTFILE` можно указать другой файл).

Пример:

```
$ echo $HISTFILE
/home/user1/.bash_history
```

Количество команд, запоминаемых в файле истории, устанавливается с помощью переменной `HISTFILESIZE` . По умолчанию переменной `HISTFILESIZE`, присваивается значение `1000` .

Пример:

```
$ echo $HISTFILESIZE 9999
```

Содержимое файла истории можно вывести с помощью команды `history`.

Пример:

```
$ history
```

685 echo $HISTFILE

686 echo $HISTFILESIZE

687 history

Примечание: В этом примере показаны лишь последние команды из файла истории. Перед каждой командой, запомненной в этом файле, выводится ее номер, с помощью которого эту команду можно вызвать заново.

Наиболее простой способ для повтора команды по ее номеру ввести знак восклицания и номер команды.

Пример:

$!685

echo $HISTFILE

/home/user1/.bash_history

Примечание: Пример, приведенный выше, демонстрирует, как из файла истории была вызвана и выполнена команда с номером 685.

Последнюю выполненную команду можно выполнить снова, если ввести в командной строке два знака восклицания `!!`.

Исполнить заново, недавно исполненную команду, можно введя в командной строке после знака восклицания первые символы ее командной строки.

Пример:

Если необходимо вновь выполнить команду `ls /tmp`, выполненную недавно, достаточно ввести в командной строке

!l

При этом история команд будет просмотрена с конца до тех пор, пока не будет найдена команда с подходящими первыми символами в ее строке. Как только такая команда будет найдена, она будет исполнена. В противном случае будет выдано сообщение об ошибке.

$ ls /opt drweb oracle

$ pwd

/home/user1

```
$ cd /tmp
$ !l
ls /opt
drweb    oracle
```

Примечание: Как видно из этого примера, пользователь ввел команду `ls /opt`, а затем еще несколько команд. После исполнения этих команд пользователю вновь понадобилось исполнить команду `ls /opt`, выполненную недавно. Пользователь добился этого, введя в командной строке `!l`.

Можно вызвать команду из истории, указав строку символов, содержащуюся в команде. Для этого следует ввести эту строку после знака восклицания и знака вопроса `!?`.

Пример:

```
$ !?cho
echo $HISTFILE
/home/user1/.bash_history
```

Примечание: Здесь из файла истории была вызвана последняя выполненная команда, содержащая подстроку cho.

Недостатком описанных выше способов вызова команд из истории является отсутствие возможности их интерактивно отредактировать. Команды, подходящие для заданных шаблонов, извлекаются из файла истории и выполняются немедленно.

Имеется способ вызова недавно выполненной команды из истории в текстовый редактор, последующего ее редактирования в этом редакторе и исполнения отредактированной команды с помощью команды `fc`.

Аргумент команды `fc` – строка, с которой начинается искомая в файле истории команда. Найденная команда отображается в редакторе, используемом в системе по умолчанию. После редактирования команда исполняется. Для того, чтобы отказаться от выполнения найденной команды следует просто удалить в редакторе всю ее командную строку.

Команда `fc -l` выводит список из последних выполненных команд, подобный списку, выводимому командой `history`. Отличие в том, что `fc -l` выводит не всю историю, а только последние команды.

С опцией `-s` команда `fc` работает в не интерактивном режиме, подобно тому, как работает команда знак восклицания.

Вызов команды `fc -s <строка>` приводит к поиску последней команды в

истории, начинающейся с заданной строки, и исполнению ее.

Опция `s` обозначает режим замены.

Пример:

```
$ ls -ld /tmp
drwxrwxrwt 30 root root 4096 Окт 2 22:01 /tmp
$ fc -s tmp=opt ls
ls -ld /opt
drwxr-xr-x 4 root root 4096 Июл 23 08:21 /opt
```

Примечание: Обратите внимание на конструкцию `tmp=opt`, она позволяет команде `fc` до исполнения найденной по заданной строке `ls` команде сначала выполнить подстановку в командной строке `opt` вместо `tmp`.

Наиболее часто для работы с историей команд используют обычные клавиши управления курсором – вверх (или `Ctrl-P`) для получения из истории предыдущей команды и вниз – (или `Ctrl-N`) для следующей команды в истории.

Найденные команды не исполняются сразу, а позволяют отредактировать командную строку и затем выполнить команду.

Автоматическое дополнение в командной строке.

Bash предоставляет удобный механизм дополнения имен файлов и команд по первым символам их имен.

Bash пытается продолжить введенные символы как имя команды или имя файла после нажатия на клавишу табуляции.

Если оболочка не может продолжить имя файла, то выводится звуковой сигнал. Это может происходить по двум причинам:

Файла или команды с таким именем не существует.

Имеется несколько вариантов продолжения строки.

Во втором случае при повторном нажатии на клавишу табуляции Bash выводит список возможных подстановок, ориентируясь на который, пользователь может ввести еще несколько символов командной строки и снова нажать на клавишу табуляции.

Механизм продолжения в Bash действует не только для имен файлов и команд. Если строка начинается с одного из символов `$`, или `~`, или `@`, то Bash попытается дополнить строку как:

имя переменной оболочки ($);

имя пользователя (~);

имя хоста (@).

Помощь и документация. Сообщения о неверном синтаксисе и встроенная в команды подсказка.

Большинство команд выводит в ответ на попытки ввести их в неверном синтаксисе сообщение об ошибке и подсказку о том, как их следует использовать.

Пример:

```
$ pwd -h
bash: pwd: -h: invalid option pwd: usage: pwd [-PL]
```

Примечание: Команда `pwd`, выводящая путь к текущему каталогу, была введена здесь с использованием опции, которая не соответствует ее синтаксису. Поэтому было получено сообщение об ошибке и на экран была выведена краткая подсказка об использовании команды.

Команды GNU позволяют получить подсказку, пользуясь опцией `--help` .

Пример:

```
$wc --help
Usage: wc[OPTION]…[FILE]…
Print line, word, and byte counts for each FILE …
```

Примечание: В этом примере продемонстрировано, что использование опции `--help` приводит к выводу базовой помощи по команде. Во всяком случае, это верно для команд GNU, а их – подавляющее большинство.

Встроенная помощь оболочки Bash.

Оболочка Bash предоставляет по своим встроенным командам собственную помощь. Список всех встроенных команд Bash можно увидеть с помощью команды `help`.

Если этой команде передать в качестве аргумента любую встроенную команду Bash, то по использованию этой команды будет выведена подсказка.

Пример:

```
$ help pwd
```

pwd: pwd [-PL]

Print the current working directory. With the -P option, pwd prints the physical directory, without any symbolic links; the -L option makes pwd follow symbolic links.

Страницы помощи man.

Система `man` (от слова manual – руководство) имеется в любой UNIX или UNIX-подобной системе. Это основное средство получения подробной информации о командах, структуре файлов конфигурации, системным вызовам и прочему.

Система `man` не рассчитана на обучение пользователей, но она предоставляет подробное описание команд.

Для получения подробной информации о команде, системном файле и т.п., необходимо вызвать команду `man` с аргументом – именем команды или иного требуемого объекта:

Пример:

```
$ man ls
```

Примечание: Эта команда выведет страницы помощи по системной команде `ls`.

Все страницы помощи `man` разделены на секции, приведенные в таблице ниже.

Секция	Информация
1	Описание команды пользователя.
2	Описание системных вызовов ядра.
3	Описание библиотек.
4	Информация о файлах устройств и иных специальных файлах.
5	Форматы конфигурационных файлов.
6	Помощь по играм.
7	Макросы, описание кодировок, различная информация для программиста.
8	Команды системного администрирования.
9	Процедуры и функции ядра.

Примечание: Часто используются секции с необычными именами, например, `n` или `1x`, соответственно для команд языка TCL и для пользовательских команд с графическим интерфейсом.

Для указания команде `man` требуемой секции ее необходимо указать `man` в качестве первого аргумента.

Примечание: указание секции используется в тех случаях, если существует несколько man-страниц с одинаковыми именами.

Пример:

```
$ man 3 zlib
```

Примечание: Эта команда выведет информацию о библиотеке компрессии `zlib`. Информация находится в третьей секции страниц `man`.

Команда `man` находит среди всех страниц помощи нужную, форматирует ее и передает ее программе постраничного просмотра, используемому в системе по умолчанию.

Для постраничного просмотра обычно используется утилита `less`.

Ниже приведены некоторые внутренние команды утилиты `less`:

Команда	Действие
Ctrl-N стрелка вниз	Следующая строка.
Ctrl-P стрелка вверх	Предыдущая строка.
Ctrl-V PgDown	Страница вниз.
Alt-V PgUp	Страница вверх.
<Space>	Следующая страница.
/строка	Поиск подстроки вниз.
?строка	Поиск подстроки вверх.
n	Найти следующее вхождение.
q	Выход из man.

Если необходимо получить все возможные страницы по данной теме, то необходимо использовать опцию `-a` команды `man`.

Примечание: При этом `man` будет выводить все найденные страницы последовательно, то есть при выходе из страницы будет отображаться следующая, если таковая имеется.

Для получения помощи о команде или файле не зная его или ее точного названия используется опция `-k` команды `man` или же полностью эквивалентная

команда apropos.

Каждая страница man начинается с раздела NAME, являющимся обязательным и содержащим описание объекта поиска.

Команда man -k производит поиск строки, заданной ей в качестве аргумента, по всем имеющимся страницам, просматривая нет ли в разделе NAME такой строки.

Пример:

```
$ man -k clock
CLOG_csync          (4)      - synchronize clocks for adjusting times
in merge
adjtimex            (2)      - tune kernel clock
alarm               (2)      - set an alarm clock for delivery of a
signal
clock               (3)      - Determine processor time
clock               (n)      - Obtain and manipulate time
clockdiff           (8)      - measure clock difference between hosts
hwclock             (8)      - query and set the hardware clock (RTC)
```

Тот же самый результат будет получен при выполнении команды apropos clock. Эта команда также просмотрит все страницы, отыскивая заданную подстроку.

Если необходимо в разделе NAME страниц помощи отыскивать не подстроку, а ее точное название, то следует использовать команду man -f или же whatis.

Пример:

```
$ man -f clock
clock               (3)      - Determine processor time
clock               (n)      - Obtain and manipulate time
```

Примечание: Легко заметить, что в данном случае были получены только те страницы, имя которых точно совпадает с заданным критерием поиска. То есть, команда apropos отыскивает строку, а команда

whatis – слово.

Файлы страниц man.

Страницы man в коммерческих UNIX системах размечены с помощью специального языка nroff.

Утилита nroff не является публично доступной и свободной для использования. В силу этого в GNU была создана собственная утилита groff, реализующая функции nroff.

Пример: Ниже приведен пример фрагмента страницы man для команды apropos.

```
.TH apropos 1 "Jan 15, 1991"
.LO 1
.SH NAME
apropos \- search the whatis database for strings
.SH SYNOPSIS
.BI apropos keyword ...
.SH DESCRIPTION
apropos searches a set of database files containing short descriptions
of system commands for keywords and displays the result on the standard output.
.SH "SEE ALSO"
whatis(1), man(1).
```

Примечание: Здесь легко заметить управляющие команды этого языка, начинающиеся с точки. Кстати, здесь с помощью конструкции .SH отмечаются ненумерованные заголовки разделов man.

Страницы помощи man хранятся в отдельных файлах, имена которых состоят из имени раздела и суффикса – секции man.

Пример: Файл rm.1 описывает команду rm, которая относится к первой секции системы

man.

Чаще всего страницы man хранятся в сжатом с помощью утилит gzip или bzip2 виде, поэтому к расширению имени файла добавляется, соответственно, gz или bz2.

Стандартное место хранения страниц man – каталог /usr/share/man.

Примечание: Однако, местоположение, использовавшееся ранее – каталог /usr/man до сих пор используется достаточно часто, хотя это не отвечает стандарту файловой системы FHS (см. man hier).

В каталоге /usr/share/man имеются подкаталоги с именами man1, man2 ... man9. Это – каталоги, в которых хранятся страницы соответствующих разделов.

Пример: В каталоге /usr/share/man/man8/ хранятся страницы помощи по восьмой секции man – командам системного администрирования.

Если система локализована и в ней установлены национальные страницы помощи, то в каталоге /usr/share/man создается специальный подкаталог с локализованными страницами.

Русские страницы man установлены в каталоге /usr/share/man/ru, в котором аналогично обычным страницам man находятся подкаталоги, соответствующие секциям локализованных страниц.

Пример:

```
$ ls /usr/share/man
man1  man2  man3  man4  man5  man6  man7  man8  man9  mann  ru
$ ls /usr/share/man/ru
man1  man2  man3  man4  man5  man6  man7  man8  man9  mann
```

Примечание: Каталог /usr/share/man/ru содержит подкаталоги с локализованными версиями страниц man.

Приложения и дополнительные команды, установленные в системе, обычно размещают свои собственные страницы помощи в каталоге /usr/share/local/man.

Во многих случаях стандартного списка секций помощи явно не достаточно, поэтому используют специальные секции, предназначенные для помощи по таким программам, как графическая подсистема X Window.

Примечание: Для этой подсистемы выделен специальная секция – 1x, хотя стандарт файловой системы предписывает использовать цифровые обозначения. Файлам страниц помощи для команд, относящихся к X Windows, принято устанавливать суффикс вида 1x, например, xterm.1x .

Документация в формате man может быть установлена и в нестандартных местах файловой системы. Переменная окружения MANPATH позволяет устанавливать пути доступа к различным каталогам, содержащим страницы man.

Для установки и переменной MANPATH можно использовать специальную команду manpath.

Примечание: Однако в локализованных версиях GNU/Linux эта переменная может работать иначе и ее установка не приведет к изменению работы man, так как при поиске страниц на поведение man влияет переменная LANG.

Файл конфигурации /etc/man.config позволяет настраивать параметры системы man. Для разработки собственных страниц man можно использовать

любой текстовый редактор, однако файл страницы помощи должен следовать формату `groff`.

Для этого в файле должны быть выделены специальные поля, приведенные в таблице ниже, и он должен быть размечен с помощью макросов `groff`.

Поле	Назначение
NAME	Предназначено для указания информации, которая будет
	использована при поиске по ключевому слову.
SYNOPSIS	Указывает краткий список опций, заключенный в квадратные скобки.
DESCRIPTION	Детально описывает функции субъекта помощи (например, программы), здесь же приводится информация о формате команды, аргументах и так далее.
OPTIONS	Содержит список всех используемых опций и их действие.
FILE	Указывает, какие файлы используются программой.
AUTHOR	Имя автора с указанием адреса электронной почты.
SEE-ALSO	Предназначено для указания других файлов, команд и т.п., которые связаны с данной страницей помощи.
COPYRIGHT	Права собственности, политика распространения и т.п.

Файлы, относящиеся к программам, использующимся только в данной системе и не связанные с системой установки программного обеспечения (`rpm`, `dselect` и пр.), следует устанавливать в каталог `/usr/local/share/man` или `/usr/local/man`.

Система TexInfo.

Система TexInfo была разработана как свободная альтернатива `man` в рамках проекта GNU. Страницы `man` по многим командам и утилитам GNU предлагают обратиться к системе TexInfo, как к более полному источнику информации.

TexInfo представляет собой гипертекстовую документацию в специальном формате, организованную иерархически.

Точки разветвления называются узлами (node).

В системе TexInfo имеется возможность переходить между соседними узлами, родительскими и дочерними узлами, а также по специальным гипертекстовым ссылкам.

Для получения базовой помощи по системе TexInfo достаточно выполнить

команду `info` без аргументов.

Примечание: Команда `info`, вызванная без аргументов, выводит наивысшую в иерархии страницу документации, отображающую верхний уровень разделов информационной системы.

Если указать конкретный объект, по которому требуется получить помощь, в качестве аргумента команды `info`, то будет выведена соответствующая страница TexInfo, в случае отсутствия таковой по этому объекту будет выведена страница `man`.

Примечание: Поэтому, в любом случае, команда `info` обеспечивает не меньшие ресурсы помощи, чем система `man`.

Пример:

```
$ info ls
```

Примечание: Эта команда отобразит информацию, имеющуюся в системе TexInfo, которая относится к команде `ls`.

Файлы документации info обычно находятся в каталоге либо `/usr/share/info`, либо /usr/info

Находясь в системе `info`, можно пользоваться следующими командами:

Команда	Действие
n	Следующий узел.
p	Предыдущий узел.
u	Переход к родительскому узлу.
l	Предыдущая страница.
Ctrl-N Курсор вниз	Следующая строка.
Ctrl-P Курсор вверх	Предыдущая строка.
Ctrl-F	Курсор вправо.
Ctrl-B	Курсор влево.
Ctrl-E End	Курсор в конец строки.
Ctrl-A Home	Курсор в начало строки.
Alt-F	Слово вправо.
Alt-B	Слово назад.
Ctrl-V PgDown	Страница вниз.
Alt-V PgUp	Страница вверх.
Alt-<	В начало страницы.
Alt->	В конец страницы.

S строка	Поиск строки на странице.
q	Выход из info.

Документация, поставляющаяся с программными пакетами.

В каталоге `/usr/share/doc` находятся подкаталоги с именами вида

приложение-версия.

Пример: `anjuta-0.1.9` – каталог, соответствующий среде разработки anjuta, версии 0.1.9.

В этих каталогах размещается дополнительная документация от разработчиков программного обеспечения. Это могут быть подробные руководства по использованию программ или же просто файлы `README`.

Ранее эта документация размещалась в каталоге `/usr/doc`.

Примечание: Насколько подробная и исчерпывающая документация хранится в этих каталогах зависит от разработчика. Часто документация бывает очень подробной и по умолчанию в данном дистрибутиве может не устанавливаться. В таком случае обычно имеется отдельный пакет, содержащий документацию для данного программного продукта.

Информация, доступная в этих каталогах, может быть ограничена следующими файлами:

`README` – краткая информация о продукте, последние изменения, не вошедшие в документацию, предупреждения и, возможно, инструкции по установке.

`INSTALL` – инструкции по установке.

`ToDo` - что необходимо изменить или доделать в продукте в ближайшее время.

`Changelog` – история изменений и разработки продукта.

`License` – текст лицензионного соглашения.

В последнее время документация по продуктам часто поставляется в формате HTML.

Источники информации в Internet.

Основным ресурсом для получения помощи по GNU/Linux является сайт The Linux Documentation Project www.tldp.org.

Он содержит следующую информацию:

HOWTO – подробная рецептурная информация по разнообразным аспектам использования GNU/Linux.

Примечание: Чаще всего HOWTO (как сделать) – это тематические описания различных программ,

методов установки, настройки и работы с ними, созданные авторами – добровольцами на основе собственного опыта работы с описываемыми им программами и системами. Профессиональный уровень и достоверность многих документов не всегда достаточны, однако в основном HOWTO – это главный способ получения пошаговых инструкций для настройки Linux систем, полезный как для начинающего пользователя, так и для опытного профессионала.

FAQ – часто задаваемые вопросы (Frequently Asked Questions). Представляют собой подборки вопросов, задаваемых пользователями программ, на которые чаще всего отвечают разработчики этих программ.

Примечание: Если вы в процессе настройки Linux системы натолкнулись на некоторую проблему, то, вполне вероятно, что кто-либо также наталкивался на нее и, может быть, проблема уже решена. Именно такого рода информация может быть найдена в FAQ.

Ссылки на руководства (Tutorals) по GNU/Linux.

Можно обратиться за дополнительной информацией по следующим адресам:

www.redhat.com – сайт компании Red Hat.

redos.red-soft.ru – сайт компании РЕД СОФТ – разработчика РЕД ОС. На сайте есть база знаний.

www.sf.net – база данных Source Forge, содержащая гигантский объем программного обеспечения и документации к ней.

www.ibm.com – на сайте IBM имеется большое количество статей и документации по GNU/Linux.

www.linuxshop.com – журнал по тематике GNU/Linux.

www.gnu.org – сайт проекта FSF GNU.

www.linuxshop.ru – русский сайт, содержащий качественно написанные статьи по Linux.

www.opennet.ru – содержит много ссылок и русскоязычной документации

Имеется несколько удобных сочетаний клавиш, предназначенных для различных способов продолжения строки команды:

`Alt-?` - выводит список возможных продолжений строки.

`Ctrl-x /` - выводит список возможных имен файлов.

`Ctrl-x ~` - выводит список возможных имен пользователей.

`Ctrl-x @` - выводит список возможных имен хостов.

`Alt-*` - вставляет в командную строку сразу все возможные варианты продолжения.

`Alt-/` - осуществляет продолжение строки, как имени файла.

`Alt-~` - осуществляет продолжение строки, как имени пользователя.

`Alt-$` - осуществляет продолжение строки, как имени переменной.

`Alt-@` - осуществляет продолжение строки, как имени хоста.

При вводе имени команды механизм дополнения позволяет быстро находить программу, которая находится в каталогах, перечисленных в переменной `PATH`.

Модуль 4. Файловая система, иерархия каталогов и работа с дисками в РЕД ОС.

Система файлов и каталогов.

Логически файловая структура в GNU/Linux организована в виде единой иерархии, напоминающей перевернутое дерево с корнем наверху.

Древовидная структура организована с помощью каталогов, содержащих файлы и подкаталоги.

Каждый каталог может иметь множество подкаталогов, но у каждого подкаталога имеется только один родительский каталог.

На каких физических носителях не находились бы файлы, в GNU/Linux они всегда находятся на одной из ветвей единой древовидной файловой структуры.

Вершиной файловой структуры является корневой каталог (root directory).

Имя корневого каталога: /.

У корневого каталога нет родительского каталога, вернее он сам является для себя родительским.

Пример: Команда, приведенная ниже, выведет содержимое корневого каталога.

```
$ ls /
bin boot dev   etc   home   lib   mnt   opt   proc   root   sbin   swap tmp   usr   var
```

Примечание: Команда `ls` выводит содержимое каталога, имя которого указано в качестве аргумента. Здесь аргумент – имя корневого каталога /.

Файлы в GNU/Linux являются основополагающими объектами, поскольку вся работа с данными, устройствами компьютера, процессами и прочим обеспечивается посредством файлов.

Файл состоит из трех компонентов:

1. Имя.
2. Inode или метаданные. В метаданных хранятся свойства файла такие как права доступа, размер, указатели на блоки данных и пр.
3. Данные (блоки данных). Не обязательный эламент. Например пустой файл не имеет блоков данных.

Обычные файлы (plain files, ordinary files или regular files) обеспечивают хранение данных в компьютере. Они представляют собой именованный блок данных, находящихся в устройстве хранения. Ядро ОС не интерпретирует содержание файлов данного типа.

Древовидная структура образуется за счет использования каталогов, которые могут содержать файлы и другие каталоги.

Каталоги являются *особым типом файлов*, предназначенным для поддержки иерархической структуры файловой системы.

Содержимое файла каталога - это перечень имен файлов, содержащихся в каталоге.

В корневом каталоге не принято хранить пользовательские файлы. Чаще всего в корневом каталоге содержатся исключительно подкаталоги.

Файлы пользователей принято хранить в их домашних каталогах, системные файлы хранятся в специальных каталогах и так далее.

Примечание: Таким образом, для доступа к файлу необходимо пройти один или несколько каталогов.

Последовательность имен каталогов, которые необходимо пройти от корневого каталога для доступа к файлу, называется путем (*path*).

Для разделения имен вложенных каталогов применяется символ `/`.

Пример: Путь к файлу `/etc/hostname` начинается, естественно, с корневого каталога, далее путь проходит в каталог `/etc`, в котором и находится указанный файл.

Имя файла может содержать любые символы кроме символов `/` и `\0` (null), а длина имени файла не должна превышать 255 символов.

Следует давать файлам осмысленные имена и избегать излишнего использования метасимволов в именах файлов.

Заглавные и строчные буквы различаются (case sensitive).

Пример: Имена файлов `TheFile` и `thefile` относятся к различным файлам.

В различных каталогах могут находиться различные файлы с одинаковыми именами.

Для однозначной идентификации файла необходимо применять полное или абсолютное имя файла. Оно состоит из пути (path) к нему в дереве каталогов и собственно имени файла.

Пути бывают двух типов:

1. Абсолютные – те, которые начинаются с символа косая черта `/` - корневого каталога. Путь доступа в таких именах начинается от корневого каталога.
2. Относительные - не начинаются с косой черты и, следовательно, показывают путь доступа к файлам относительно текущего каталога.

Имя файла может содержать точки `.` . В GNU/Linux, в отличие, например, от MS DOS, никакого особого значения точки в именах файлов не имеют.

Часть имени файла, находящуюся после точки, называется суффиксом (расширением) имени файла.

Суффиксы сообщают пользователю о том, какого типа (характера) информация хранится в файле.

Пример: Файл `myarch.gz` является архивом, сжатым утилитой `gzip`. Имя файла может содержать более одного расширения: `tarball.tar.gz`.

Файлы, у которых точка является первым символом в имени, являются скрытыми и командой `ls` не выводятся. Или, что более точно, файловые команды их пропускают, если имена этих файлов не указанны явно.

Список всех файлов (в том числе и скрытых) можно получить, пользуясь командой `ls` с опцией `-a` (`--all`) или опцией `-A`:

Пример:

```
$ > .hidden
$ ls
$ ls -a
.  ..  .hidden
$ ls -A
.hidden
```

Примечание: В этом примере команда `ls` не обнаружила никаких файлов в текущем каталоге. Тем не менее, этот каталог содержит файл `.hidden`, являющийся скрытым, так как его имя начинается с точки. Этот файл был создан в каталоге с помощью команды `> .hidden`. Для вывода списка всех файлов используется команда `ls -a`, в том числе и скрытых, в текущем каталоге. Эта команда выводит имя файла `.hidden`. Помимо него выведены еще два имени файлов – `.` (точка), то есть имя текущего каталога, и `..`, являющееся именем родительского каталога. Команда `ls -A` не отобразила имена текущего и родительского каталога, а скрытый файл был ей показан.

Устройство файловой системы.

Файловая система построена из трех основных компонент:

суперблок (superblock);

массив индексных дескрипторов (inode list);

блоки хранения данных.

<table>
<tr><th colspan="6">Магнитный диск</th></tr>
<tr><td rowspan="2">Суперблок</td><td rowspan="2">Массив индексных дескрипторов</td><td colspan="4">Данные</td></tr>
<tr><td>Блок1</td><td>Блок2</td><td>Блок3</td><td>…</td><td>БлокN</td></tr>
</table>

Суперблок содержит основную информацию, необходимую для монтирования и работы файловой системы:

- тип файловой системы;

- размер и количество блоков в файловой системе;
- количество индексных дескрипторов;
- время последнего монтирования;
- информация о том была ли размонтирована файловая система;
- счетчик числа монтирований;
- список свободных блоков и индексных дескрипторов, и т.п.

В случае если суперблок файловой системы испорчен, то монтирование файловой системы без его восстановления невозможно.

Современные файловые системы специально сохраняют в заранее известных блоках диска копии суперблока, обеспечивая возможность их использования при восстановлении структуры файловой системы.

В индексных дескрипторах хранятся метаданные (атрибуты) файлов:

- Владелец и группа пользователей файла;
- Права доступа к файлу;
- Тип файла;
- Количество имен у файла (link count);
- Дата доступа к файлу (файл открыт на чтение);
- Дата модификации (файл открыт на запись);
- Дата изменения метаданных;
- Количество блоков, занятое файлом;
- Указатели на блоки данных файла.

Каталоги предоставляют собой особый тип файлов, содержащие таблицу, в которой содержатся имена файлов, находящихся в данном каталоге, и соответствующие им номера индексных дескрипторов (inode).

Записи, хранящиеся в каталоге, связывают имена файлов с их индексными дескрипторами, а те, в свою очередь, предоставляют информацию о местонахождении блоков данных файла.

Одному и тому же индексному дескриптору может соответствовать несколько имен файлов.

Если у одного и того же файла имеется несколько имен, то говорят, что между этими именами существует жесткая связь (hard link).

Количество разных имен файл фиксируется счетчиком имен файла (link counter) в его индексном дескрипторе.

Примечание: То есть, разные имена файлов указывают на одни метаданные, и, следовательно, на одни и те же блоки данных. Все имена файла совершенно эквивалентны и изменение содержимого этих файлов будет совершенно синхронным. Нет никакой возможности определить, какое имя у файла было исходно, а какое появилось потом.

У каждого каталога имеется как минимум два имени (то есть счетчик имен всегда не меньше двух):

Обычное имя каталога, находящееся в родительском каталоге (например, /home/user1).

Имя точка (.) - имя текущего каталога.

Примечание: При появлении в любом каталоге подкаталога, количество имен у этого каталога увеличивается на единицу, так как в дочернем каталоге всегда содержится имя две точки (..) - имя родительского каталога.

Количество имен у файла можно определить с помощью команды ls -l, а получить номера индексных дескрипторов можно, используя ls -i.

Пример:

$ ls -ldi /etc

6 drwxr-xr-x 87 root root 6064 Дек 14 00:13 /etc

Примечание: Из полученного листинга видно, что номер inode *каталога* /etc *равен* 6*, а количество имен у этого каталога равняется* 87*, следовательно, количество подкаталогов в нем равняется* 85*.*

$ ls -l /etc | grep -c '^d'

85

Примечание: Эта команда подтверждает, что приведенное выше правило подсчета имен каталогов верно.

Подробную информацию об индексном дескрипторе файла можно получить, используя команду stat.

Пример:

$ stat /etc

File: `/etc'

Size: 6064 Blocks: 12 IO Block: 4096 Directory

Device: 305h/773d Inode: 6 Links: 87

Access: (0755/drwxr-xr-x) Uid: (0/ root) Gid: (0/ root)

Access: 2003-06-17 23:19:59.000000000 +0600

Modify: 2003-12-14 00:13:23.000000000 +0500

Change: 2003-12-14 00:13:23.000000000 +0500

Монтирование файловых систем.

В GNU/Linux все файловые системы, доступные для работы с ними, должны быть “подцеплены” к логической структуре файлов и каталогов.

Процесс “подцепления” файловой системы, существующей на дисковом или ином блочном устройстве, в общее дерево файлов и каталогов называется монтированием.

Каталог, в который произошло “подцепление” отформатированного устройства, называется точкой монтирования.

За исключением файловых систем, для которых установлены специальные настройки в, например, файле /etc/fstab, монтирование файловых систем производится суперпользователем.

Стандарт FHS предписывает, что точки монтирования временных файловых систем должны находиться в каталоге /mnt.

Временные файловые системы для сменных носителей должны в соответствии с FHS находится в каталоге /media

Примечание: каталог */media/cdrom* может быть точкой монтирования для CD-ROM, а */media/usbflash* – для флоппи диска.

Команда mount монтирует файловую систему указанного с помощью опции -t типа (по умолчанию ext2) в каталог - точку монтирования.

Первый аргумент команды mount - файл блочного устройства, на котором находится монтируемая файловая система.

Второй аргумент - точка монтирования этой файловой системы.

Пример: для монтирования USB диска с файловой системой EXT2 следует выполнить команду:

```
# mount /dev/sda1 /mnt/usbflash
```

Если команда mount вызвана без аргументов, то она показывает список смонтированных файловых систем, то есть имена файлов устройств и соответствующих им точек монтирования.

Пример:

```
# mount | grep ^/dev
/dev/vda2 on / type ext4 (rw,relatime,seclabel)
/dev/vda1 on /boot type ext4 (rw,relatime,seclabel)
/dev/sda1 on /media/usbflash type ext3 (rw,relatime,seclabel)
```

Последние версии mount умеют сами определять тип подключаемой ФС, если нет, то при монтировании, например, CD ROM необходимо указать тип файловой системы iso9660 :

Пример:

mount -t iso9660 /dev/cdrom /media/cdrom

Монтирование файловой системы подменяет индексный дескриптор каталога - точки монтирования. До монтирования индексный дескриптор соответствует каталогу, находящемуся в файловой системе, к которой монтируется устройство. После монтирования - индексный дескриптор каталога - точки монтирования принадлежит уже смонтированной файловой системе.

Пример:

ls -ldi /media/usbflash/

529793 drwxr-xr-x. 2 root root 4096 Feb 10 12:13 /media/usbflash/

mount /dev/sda1 /media/usbflash

ls -ldi /media/usbflash/

2 drwxr-xr-x. 3 root root 4096 Feb 10 08:47 /media/usbflash/

Примечание: Из этого примера заметно, что до монтирования временной файловой системы в каталог /media/usbdisk, *индексный дескриптор каталога был* 529793 , *а после монтирования* 2. *У всех корневых каталогов файловых систем на диске индексный дескриптор имеет номер* 2. *У виртуальных файловых систем корневой каталог имеет номер inode 1.*

Пример:

ls -ldi /boot /

2 dr-xr-xr-x. 18 root root 4096 Nov 9 2019 /

2 dr-xr-xr-x. 6 root root 4096 Feb 9 10:39 /boot

mount | grep ^/dev

/dev/vda2 on / type ext4 (rw,relatime,seclabel)

/dev/vda1 on /boot type ext4 (rw,relatime,seclabel)

/dev/sda1 on /media/usbflash type ext3 (rw,relatime,seclabel)

ls -ldi /proc

1 dr-xr-xr-x. 91 root root 0 Feb 9 10:50 /proc

Примечание: В этом примере демонстрируется то, что для каждой файловой системы inode корневого каталога, то есть точки монтирования - 2.

Как и операцию монтирования, размонтировать файловые системы (без специальных настроек в /etc/fstab) имеет право только суперпользователь.

Для размонтирования файловой системы применяется команда umount, которой в качестве аргумента должен быть задан единственный аргумент - либо точка монтирования, либо файл устройства.

Пример:

ls /media/usbflash/

lost+found somedocs.txt

[root@lin00 ~]# umount /media/usbflash/

[root@lin00 ~]# ls /media/usbflash/

Примечание: В этом примере показана работа команды umount*. После ее выполнения в каталоге - точке монтирования больше нет доступа к файлам на временной файловой системе.*

Хорошим правилом является следующее: не следует хранить какие-либо файлы в каталогах, являющихся точками монтирования временных файловых систем.

Работа с разделом подкачки.

Раздел или файл подкачки необходимы при работе GNU/Linux для обеспечения временного перемещения страниц памяти из ОЗУ в этот раздел или файл.

Такое перемещение происходит при недостатке физической памяти.

Процесс обмена страницами памяти между ОЗУ и разделом подкачки называется swapping, а раздел подкачки называется swap – разделом.

Получить информацию об использовании раздела подкачки можно с помощью команды swapon -s.

Пример:

$ /sbin/swapon -s

Filename Type Size Used Priority

/dev/hda5 partition 248968 0 -1

Примечание: Информация, полученная от команды swapon -s*, демонстрирует следующее: имеется раздел подкачки в первом логическом разделе Primary Master IDE диска. Размер раздела – 248 Мб, из них использовано в настоящий момент – 0.*

Столбец приоритет отображает порядок использования swap разделов. Сначала будут использованы разделы подкачки с большим номером приоритета, затем – с меньшим.

При необходимости можно добавить в систему дополнительные области подкачки. Они могут быть размещены как в разделах дисков, так и в обычных файлах.

Создавать, подключать и отключать разделы подкачки имеет право суперпользователь.

Примечание: Если в системе не хватает ОЗУ для выполнения каких-либо приложений, то создание раздела подкачки может быть единственно возможным методом решения этой проблемы.

Если в системе имеется несколько жестких дисков, то рекомендуется для оптимизации производительности системы разместить разделы подкачки на нескольких дисках.

Для создания файла подкачки необходимо создать файл, заполненный нулями.

Пример: Приведенная ниже команда создает 128 Мб файл, заполненный нулями:

```
$ dd if=/dev/zero of=swap.file bs=1k count=131072
131072+0 входных записей
131072+0 выходных записей
$ ls -l swap.file
-rw-r--r-- 1 user1 user1 134217728 Дек 19 17:43 swap.file
```

Команда mkswap создает в файле или разделе (указанном при помощи файла устройства) область подкачки, специальным образом размечая ее.

Пример:

```
$ /sbin/mkswap swap.file
Setting up swapspace version 1, size = 134213 kB
```

Примечание: Эта команда создала в файле swap.file *область подкачки.*

Создать область подкачки, в разделе можно лишь тогда, когда тип раздела установлен Linux Swap (82 тип в команде fdisk).

Для создания раздела подкачки в разделе выполняется та же команда mkswap

Пример:

```
# mkswap -c /dev/sda2
```

Примечание: В этом примере создается раздел подкачки на втором первичном разделе первого SCSI диска. При этом используется опция -c*, которая перед созданием области подкачки проверяет поверхность диска на наличие плохих блоков.*

Подключить созданный раздел или файл подкачки можно с помощью команды swapon , а отключить – с помощью команды swapoff.

Пример:

```
# swapon ~user1/swap.file

# swapon -s
Filename Type Size Used Priority
/dev/sda5 partition 248968 0 -1
/home/user1/swap.file file 131064 0 -2

# swapoff ~user1/swap.file

# swapon -s
Filename Type Size Used Priority
/dev/sda5 partition 248968 0 -1
```

Примечание: В этом примере был подключен дополнительный файл подкачки с помощью команды swapon*, а затем этот файл был отключен командой* swapoff.

Файл информации о файловых системах /etc/fstab.

Конфигурационный файл /etc/fstab (filesystem table) содержит информацию о файловых системах, которые нужно смонтировать при загрузке или по требованию пользователя.

Информация в файле /etc/fstab представлена в виде таблицы:

Пример:

```
# cat /etc/fstab

#
# /etc/fstab
# Created by anaconda on Sat Nov 9 04:18:20 2019
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
UUID=1a5655e7-613b-4733-ba6b-e598633402fb / ext4 defaults 1 1
UUID=897234d2-2307-406f-990d-a9620bcb4d1f /boot ext4 defaults 1 2
UUID=5a4f53ca-c983-4def-8535-e2541d8419bb swap swap defaults 0 0
```

Строки, начинающиеся с символа # являются комментариями.

Таблица, содержащаяся в файле /etc/fstab , состоит из колонок полей, назначение которых следующее:

Первое поле (fs_spec) содержит указатель на монтируемое устройство. Здесь может быть имя файла устройства, UUID файловой системы, UUD GPT, метка GPT или метка тома в файловой системе.

Второе (fs_file) указывает каталог - точку монтирования

Третье (fs_vfstype) содержит тип файловой системы на данном носителе.

Четвертое (fs_mntops) содержит опции команды mount , которые должны быть использованы при монтировании данной файловой системы. Для разделов swap это поле должно содержать sw.

Пятое (fs_freq) указывает надо ли для данной файловой системы производить автоматическое резервное копирование (backup) командой dump . Если в этом поле находится 1, то резервное копирование производится, если 0 – нет.

Шестое (fs_passno) предназначено для порядка проверки целостности файловых систем при загрузке операционной системы. Для корневой файловой системы в этом поле должно быть установлено значение 1. Для других файловых систем, которые необходимо проверять при загрузке, следует указать 2. Если проверка не требуется, то в этом поле ставится 0 .

Ниже приведена таблица, содержащая часто используемые опции команды mount, указываемые в поле fs_mntops файла /etc/fstab .

Опция	**Назначение**
defaults	Установки: rw,suid,dev,exec,auto,nouser,asynch.
asynch	Асинхронный режим ввода/вывода.
auto	Монтировать во время загрузки.
noauto	Не монтировать во время загрузки.
exec	Разрешение выполнения файлов с машинным кодом.
noexec	Запрет выполнения файлов с машинным кодом.
suid	Бит SUID устанавливать можно.
nosuid	Запрещена установка битов SUID.
user	Обычный пользователь может монтировать устройство.
nouser	Монтировать разрешено только суперпользователю.
ro	Режим только для чтения.
rw	Разрешено как чтение, так и запись.

Наличие записи в файле /etc/fstab означает, что данная файловая система может быть смонтирована без указания обоих аргументов командной строки mount – файла устройства и каталога – точки монтирования. Для монтирования файловой системы, указанной в /etc/fstab , достаточно указать либо точку монтирования, либо файл устройства.

В ОС построенных на systemd для монтирования разделов на основе /etc/fstab генерируются специальные юниты типа mount.

 # systemctl list-units -t mount

UNIT LOAD ACTIVE SUB DESCRIPTION
-.mount loaded active mounted Root Mount
boot.mount loaded active mounted /boot
dev-hugepages.mount loaded active mounted Huge Pages File System
dev-mqueue.mount loaded active mounted POSIX Message Queue File System
run-user-0.mount loaded active mounted /run/user/0
sys-kernel-config.mount loaded active mounted Kernel Configuration File System
sys-kernel-debug.mount loaded active mounted Kernel Debug File System

LOAD = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB = The low-level unit activation state, values depend on unit type.

7 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.

 # systemctl status [-].mount
● -.mount - Root Mount
Loaded: loaded (/etc/fstab; generated)
Active: active (mounted) since Tue 2021-02-09 10:50:18 +05; 1 day 6h ago
Where: /
What: /dev/vda2
Docs: man:fstab(5)
man:systemd-fstab-generator(8)

Юнит для монтирования можно создать самостоятельно или написать специальные опции монтирования в файл /etc/fstab , которые сформируют юнит монтирования с нужными вам параметрами. (см. man 5 systemd.mount).

Использование /etc/fstab считается предпочтительным.

Пример: создадим юнит, который описывает новую точку монтирования:

 # mkfs.ext4 /dev/sda1
mke2fs 1.44.6 (5-Mar-2019)
/dev/sda1 contains a ext3 file system
last mounted on /media/usbflash on Wed Feb 10 12:16:19 2021
Proceed anyway? (y,N) y
Creating filesystem with 1310720 4k blocks and 327680 inodes
Filesystem UUID: **a13a401f-cdf5-48e3-a277-21409ad76de0**
Superblock backups stored on blocks:
...

mkdir **/storage1**

```
# vi /etc/systemd/system/storage1.mount
# cat /etc/systemd/system/storage1.mount
[Unit]
Description=Persistent Mount Point for directory /storage1

[Mount]
What=/dev/disk/by-uuid/a13a401f-cdf5-48e3-a277-21409ad76de0
Where=/storage1
Type=ext4
Options=defaults

[Install]
WantedBy=sysinit.target

# systemctl start storage1.mount

# systemctl status storage1.mount
● storage1.mount - Persistent Mount Point for directory /storage1
Loaded: loaded (/etc/systemd/system/storage1.mount; disabled; vendor preset:>
Active: active (mounted) since Wed 2021-02-10 17:49:04 +05; 6s ago
Where: /storage1
What: /dev/sda1
Tasks: 0 (limit: 12472)
Memory: 76.0K
CGroup: /system.slice/storage1.mount

Feb 10 17:49:04 lin00 systemd[1]: Mounting Persistent Mount Point for
directory>
Feb 10 17:49:04 lin00 systemd[1]: Mounted Persistent Mount Point for directory

# systemctl enable storage1.mount
Created symlink /etc/systemd/system/sysinit.target.wants/storage1.mount →
/etc/systemd/system/storage1.mount.

# systemctl status storage1.mount
● storage1.mount - Persistent Mount Point for directory /storage1
Loaded: loaded (/etc/systemd/system/storage1.mount; enabled; vendor preset: >
Active: active (mounted) since Wed 2021-02-10 17:49:04 +05; 2min 17s ago
...
```

Мониторинг дисковых ресурсов.

Команда lsblk выдает информацию о дисках, разделах и точках монтирования.

Пример:

```
# lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda 8:0 0 20G 0 disk
├─sda1 8:1 0 5G 0 part /storage1
└─sda2 8:2 0 2G 0 part /storage2
sr0 11:0 1 1024M 0 rom
vda 253:0 0 20G 0 disk
├─vda1 253:1 0 1G 0 part /boot
├─vda2 253:2 0 17G 0 part /
└─vda3 253:3 0 2G 0 part [SWAP]

# lsblk -f
NAME FSTYPE LABEL UUID MOUNTPOINT
sda
├─sda1 ext4 a13a401f-cdf5-48e3-a277-21409ad76de0 /storage1
└─sda2 xfs e5b66aa7-c101-4bcd-bb63-2d10d2e4aa44 /storage2
sr0
vda
├─vda1 ext4 BOOT 897234d2-2307-406f-990d-a9620bcb4d1f /boot
├─vda2 ext4 ROOT 1a5655e7-613b-4733-ba6b-e598633402fb /
└─vda3 swap SWAP1 5a4f53ca-c983-4def-8535-e2541d8419bb [SWAP]
```

Команда findmnt показывает точки монтирования в удобном для восприятия виде.

Пример:

```
# findmnt
TARGET SOURCE FSTYPE OPTIONS
/ /dev/vda2 ext4 rw,relatime,seclabel
├─/sys sysfs sysfs rw,nosuid,nodev,noexec,
│ ├─/sys/kernel/security securityfs securit rw,nosuid,nodev,noexec,
│ ├─/sys/fs/cgroup tmpfs tmpfs ro,nosuid,nodev,noexec,
│ │ ├─/sys/fs/cgroup/systemd cgroup cgroup rw,nosuid,nodev,noexec,
│ │ ├─/sys/fs/cgroup/perf_event cgroup cgroup rw,nosuid,nodev,noexec,
│ │ ├─/sys/fs/cgroup/pids cgroup cgroup rw,nosuid,nodev,noexec,
│ │ ├─/sys/fs/cgroup/freezer cgroup cgroup rw,nosuid,nodev,noexec,
│ │ ├─/sys/fs/cgroup/cpuset cgroup cgroup rw,nosuid,nodev,noexec,
│ │ ├─/sys/fs/cgroup/net_cls,net_prio cgroup cgroup rw,nosuid,nodev,noexec,
│ │ ├─/sys/fs/cgroup/cpu,cpuacct cgroup cgroup rw,nosuid,nodev,noexec,
│ │ ├─/sys/fs/cgroup/rdma cgroup cgroup rw,nosuid,nodev,noexec,
```

```
│   │   ├─/sys/fs/cgroup/devices cgroup cgroup rw,nosuid,nodev,noexec,
│   │   ├─/sys/fs/cgroup/memory cgroup cgroup rw,nosuid,nodev,noexec,
│   │   ├─/sys/fs/cgroup/blkio cgroup cgroup rw,nosuid,nodev,noexec,
│   │   └─/sys/fs/cgroup/hugetlb cgroup cgroup rw,nosuid,nodev,noexec,
│   ├─/sys/fs/pstore pstore pstore rw,nosuid,nodev,noexec,
│   ├─/sys/fs/bpf bpf bpf rw,nosuid,nodev,noexec,
│   ├─/sys/fs/selinux selinuxfs selinux rw,relatime
│   ├─/sys/kernel/debug debugfs debugfs rw,relatime,seclabel
│   └─/sys/kernel/config configfs configf rw,relatime
├─/proc proc proc rw,nosuid,nodev,noexec,
│ └─/proc/sys/fs/binfmt_misc systemd-1 autofs rw,relatime,fd=35,pgrp=
├─/dev devtmpfs devtmpf rw,nosuid,seclabel,size
│   ├─/dev/shm tmpfs tmpfs rw,nosuid,nodev,seclabe
│   ├─/dev/pts devpts devpts rw,nosuid,noexec,relati
│   ├─/dev/mqueue mqueue mqueue rw,relatime,seclabel
│   └─/dev/hugepages hugetlbfs hugetlb rw,relatime,seclabel,pa
├─/run tmpfs tmpfs rw,nosuid,nodev,seclabe
│ └─/run/user/0 tmpfs tmpfs rw,nosuid,nodev,relatim
├─/boot /dev/vda1 ext4 rw,relatime,seclabel
├─/storage1 /dev/sda1 ext4 rw,relatime,seclabel
└─/storage2 /dev/sda2 xfs rw,relatime,seclabel,at
```

Команда blkid находит и печатает атрибуты блочных устройств.

Пример:

```
# blkid
/dev/vda1: LABEL="BOOT" UUID="897234d2-2307-406f-990d-a9620bcb4d1f" TYPE="ext4" PARTUUID="706c87eb-01"
/dev/vda2: LABEL="ROOT" UUID="1a5655e7-613b-4733-ba6b-e598633402fb" TYPE="ext4" PARTUUID="706c87eb-02"
/dev/vda3: LABEL="SWAP1" UUID="5a4f53ca-c983-4def-8535-e2541d8419bb" TYPE="swap" PARTUUID="706c87eb-03"
/dev/sda1: UUID="a13a401f-cdf5-48e3-a277-21409ad76de0" TYPE="ext4" PARTUUID="87ce5033-6121-0b42-b875-05ff23508aad"
/dev/sda2: UUID="e5b66aa7-c101-4bcd-bb63-2d10d2e4aa44" TYPE="xfs" PARTUUID="fea45c49-5143-a34b-8b92-9e78bc409109"

# blkid -i /dev/sda1
/dev/sda1: MINIMUM_IO_SIZE="512" PHYSICAL_SECTOR_SIZE="512" LOGICAL_SECTOR_SIZE="512"

# blkid -t TYPE="ext4"
```

/dev/vda1: LABEL="BOOT" UUID="897234d2-2307-406f-990d-a9620bcb4d1f" TYPE="ext4" PARTUUID="706c87eb-01"
/dev/vda2: LABEL="ROOT" UUID="1a5655e7-613b-4733-ba6b-e598633402fb" TYPE="ext4" PARTUUID="706c87eb-02"
/dev/sda1: UUID="a13a401f-cdf5-48e3-a277-21409ad76de0" TYPE="ext4" PARTUUID="87ce5033-6121-0b42-b875-05ff23508aad"

Команда df устройство выводит количество свободного места в блоках на специфицированном устройстве, а если оно не указано, то на всех смонтированных файловых системах.

Удобно использовать опцию -h, для отображения информации в понятных для пользователя единицах (human readable format):

Пример:

```
$ df -h
Filesystem Size Used Avail Use% Mounted on
devtmpfs 975M 0 975M 0% /dev
tmpfs 989M 0 989M 0% /dev/shm
tmpfs 989M 17M 973M 2% /run
tmpfs 989M 0 989M 0% /sys/fs/cgroup
/dev/vda2 17G 3.2G 13G 21% /
/dev/vda1 976M 163M 747M 18% /boot
tmpfs 198M 0 198M 0% /run/user/0
/dev/sda1 4.9G 20M 4.6G 1% /storage1
/dev/sda2 2.0G 47M 2.0G 3% /storage2
```

Для получения информации о наличии свободных индексных дескрипторов необходимо вызвать команду df -i :

Пример:

```
$ df -i
Filesystem Inodes IUsed IFree IUse% Mounted on
devtmpfs 249450 376 249074 1% /dev
tmpfs 253090 1 253089 1% /dev/shm
tmpfs 253090 513 252577 1% /run
tmpfs 253090 17 253073 1% /sys/fs/cgroup
/dev/vda2 1114112 138925 975187 13% /
/dev/vda1 65536 320 65216 1% /boot
tmpfs 253090 5 253085 1% /run/user/0
/dev/sda1 327680 11 327669 1% /storage1
/dev/sda2 1048576 3 1048573 1% /storage2
```

Примечание: Обозначения K и M присутствующие в листинге выше следует понимать, как тысячи и миллионы индексных дескрипторов.

Для того, чтобы узнать сколько пространства занимают файлы в каталоге следует использовать команду du, отображающую количество блоков, занимаемое каждым каталогом и каждым его подкаталогом.

Можно использовать du -h для отображения информации в удобных единицах:

Пример:

```
$ du -h /etc/rc.d
161K /etc/rc.d/init.d
1.0K /etc/rc.d/rc0.d
1.0K /etc/rc.d/rc1.d
1.0K /etc/rc.d/rc2.d
1.0K /etc/rc.d/rc3.d
1.0K /etc/rc.d/rc4.d
1.0K /etc/rc.d/rc5.d
1.0K /etc/rc.d/rc6.d
18K /etc/rc.d/scripts
207K /etc/rc.d
```

Также можно отобразить лишь суммарную информацию о каталоге, без вывода подробностей о подкаталогах. Для этого используется du -s

Пример:

```
$ du -sh ~
467M /home/user1
```

Примечание: В этом примере получена информация о суммарном пространстве, используемом домашним каталогом пользователя.

Получение списков файлов и каталогов.

Текущим каталогом называют каталог, в котором будет производиться работа файловых команд без указания пути (абсолютного или относительного) к файлам.

Если в качестве аргумента файловой команде указано имя файла без пути к нему, то действие команды будет применено к файлу в текущем каталоге.

Команда `pwd` выводит полное имя текущего каталога

Пример:

```
$ pwd
/home/anna
```

Примечание: В этом примере пользователь `anna` вывела имя текущего каталога с помощью команды

pwd. Имя текущего каталога: /home/anna – это (вероятнее всего, но совсем не обязательно) домашний каталог пользователя anna.

Для вывода списка файлов и подкаталогов текущего каталога можно использовать команду `ls`.

Пример:

$ ls

archive.gz referat.txt text.txt

Примечание: Команда `ls` вывела имена файлов, содержащихся в текущем каталоге: `archive.gz`, `referat.txt` и `text.txt`.

Для вывода содержимого каталога, не являющегося текущим, необходимо указать его имя в качестве аргумента:

Пример:

#ls /home/anna

archive.gz referat.txt text.txt

Примечание: В этом примере суперпользователь вывел содержимое домашнего каталога пользователя anna, указав его полное имя. То, что команду выполнил суперпользователь указывает решетка # в качестве символа приглашения. При обычных настройках системы безопасности только суперпользователь имеет право просматривать содержимое чужих домашних каталогов.

К текущему каталогу можно обращаться по имени `.` (точка). Команда `ls` действует аналогично команде `ls ./` или `ls . .`

Для надежности при указании относительных путей необходимо ставить `./` в начале пути.

Пример: `./myfile` – файл, находящийся в текущем каталоге.

Имя `..` соответствует родительскому каталогу текущего каталога.

Пример: для каталога `/home/anna` родительским каталогом является каталог `/home`. Таким образом, если текущим каталогом является `/home/anna`, то команда `ls ..` выведет содержимое каталога `/home`.

Обычно при регистрации в системе нового пользователя ему назначается его домашний каталог, в котором он может хранить личные файлы.

При входе пользователей в систему текущими обычно становятся их домашние каталоги.

Имена домашних каталогов чаще всего совпадают с именами пользователей – владельцев этих каталогов.

Стандартное место для размещения домашних каталогов пользователей - каталог `/home`.

Пример: домашний каталог пользователя `anna` - `/home/anna`.

При пользовании оболочкой Bash существует короткий путь для указания имени домашнего каталога: имя ~ указывает на домашний каталог пользователя, вошедшего в систему, а ~имя_пользователя – на домашний каталог указанного пользователя.

Пример: команда, приведенная ниже, выведет содержимое домашнего каталога пользователя anna:

```
# ls ~anna
archive.gz referat.txt text.txt
```

Примечание: В этом примере суперпользователь воспользовался более удобным путем обращения к домашнему каталогу пользователя anna вместо указания его полного имени /home/anna.

При входе в сеанс имя домашнего каталога пользователя сохраняется в переменной окружения HOME.

Примечание: Поэтому команда ls $HOME, выполненная каким-либо пользователем, выведет содержимое его домашнего каталога.

Для получения подробных данных о выводимых командой ls файлах необходимо воспользоваться опцией -l

Пример:

```
$ ls -l /proc/sys итого 0
dr-xr-xr-x  2 root  root  0 Окт  7 06:57 abi
dr-xr-xr-x  2 root  root  0 Окт  7 06:57 debug
dr-xr-xr-x  4 root  root  0 Окт  7 06:57 dev
dr-xr-xr-x  4 root  root  0 Окт  7 06:57 fs
dr-xr-xr-x  3 root  root  0 Окт  7 06:57 kernel
dr-xr-xr-x  9 root  root  0 Окт  7 06:57 net
dr-xr-xr-x  2 root  root  0 Окт  7 06:57 proc
dr-xr-xr-x  2 root  root  0 Окт  7 06:57 vm
```

Примечание: В первом столбце выводится тип файла, далее права доступа к файлу, количество имен файла (жестких связей), владелец файла, первичная группа владельца, размер файла, дата изменения и имя файла. Права владения и права доступа будут рассмотрены ниже.

В первом столбце листинга команды ls -l выводятся типы файлов, поддерживаемых ядром ОС:

1. - - обычные файлы.
2. d – каталоги.
3. l – символические ссылки (содержат указатели на другие файлы).
4. b – блочные устройства (специальные файлы, предназначенные для обращения к устройствам, информация на которые записывается и

считывается оттуда блоками, например, флоппи диск).

5. `c` - символьные устройства (специальные файлы, предназначенные для ввода – вывода с неформатированных устройства, таких, как терминал или мышь, обмен с которыми производится посимвольно).
6. `p` – именованный канал (PIPE или FIFO, они являются одним из вариантов организации межпроцессного взаимодействия).
7. `s` – сокеты (socket, предназначенные для организации сетевого межпроцессного взаимодействия).

Другая часто используемая опция команды `ls` – это опция `-F`. При использовании этой опции после имен каталогов выводится `/`, после имен исполняемых файлов - `*`, после символьных ссылок - `@` .

Пример:

```
$ ls –F ~
Desktop/ intro.txt scr1.sh*
```

Примечание: Команда `ls` с опцией `F` вывела содержимое домашнего каталога пользователя с использованием символов подсказки. Здесь `Desktop` – каталог, так как после его имени выводится знак `/`. Файл `intro.txt` – обычный файл. А скрипт `scr1.sh` является исполняемым файлом, так как после его имени выведен символ `*`.

Для получения информации собственно о каталогах, а не о файлах, содержащихся в них необходимо воспользоваться опцией `-d` команды `ls` .

Чаще всего опция `-d` команды `ls` используется для вывода информации о каталоге в подробном формате, то есть совместно с опцией `-l`:

Пример:

```
$ ls -ld /etc
drwxr-xr-x  87 root root 6064 Окт 7 06:16 /etc
```

Примечание: В этом примере получена подробная информация о каталоге `/etc`. Если бы опция `-d` отсутствовала, то была бы получена информация не о каталоге, а о файлах, содержащихся в нем.

Перемещение по дереву каталогов.

Команда `cd` предназначена для смены текущего каталога.

Если она вызвана без аргументов, то текущим становится домашний каталог пользователя.

Пример:

```
$ pwd
```

/home/anna/books/poetry/Lermontov

$ cd

$ pwd

/home/anna

Примечание: В первой строке примера была дана команда вывести имя текущего каталога: `/home/anna/books/poetry/Lermontov`. Далее была выполнена команда `cd` без аргументов. Выполненная затем команда `pwd` вывела имя текущего каталога: `/home/anna`. То есть команда `cd` сделала текущим домашний каталог пользователя.

Для перехода в каталог `dir_name` необходимо вызвать команду `cd` с аргументом

dir_name

Пример:

$ pwd

/home/anna

$ cd /etc

$ pwd

/etc

Примечание: В данном примере команда `cd /etc` сделала текущим каталог `/etc`.

Создание и удаление файлов и каталогов.

Команда `touch file` создает пустой файл с именем `file`

Пример:

$ ls

article1.txt article2.txt

$ touch empty

$ ls

article1.txt article2.txt empty

Примечание: В этом примере создан пустой файл `empty`, имя которого выведено последней командой `ls` среди других имен файлов, находившихся в текущем каталоге до его создания. Преимуществом команды `touch` является возможность создания сразу многих файлов, если их имена указать в качестве аргументов.

Если в качестве аргумента команды `touch` указан файл, который уже существует, то у этого файла в результате выполнения команды `touch` будет изменена дата модификации.

Пример:

$ ls

$ touch f1

$ ls -l итого 0

-rw-r--r-- 1 tania tania 0 Окт 7 08:46 f1

$ touch f1 f2 f3

$ ls -l

итого 0

-rw-r--r-- 1 tania tania 0 Окт 7 08:47 f1

-rw-r--r-- 1 tania tania 0 Окт 7 08:47 f2

-rw-r--r-- 1 tania tania 0 Окт 7 08:47 f3

Примечание: Файл `f1` был создан командой `touch`. Затем эта же команда была вызвана с аргументами `f1 f2 f3`, в результате чего файлы `f2` и `f3` были созданы, а у файла `f1` была изменена дата модификации. Содержимое файла `f1` после этого не изменилось.

Более распространенный способ создать пустой файл заключается в использовании перенаправления "пустого вывода" в файл с помощью команды `> newfile`.

Для удаления файла необходимо воспользоваться командой `rm`

Пример:

$ ls

f1 f2 f3

$ rm -f f1 f2

$ rm -i f3

rm: удалить пустой обычный файл `f3'? y

$ ls

$

Примечание: Исходно в текущем каталоге находились три файла `f1`, `f2` и `f3`. Первые два из них были удалены командой `rm -f`. Эта команда удаляет файлы без каких-либо дополнительных запросов, так как используется опция `-f`. Наоборот, если необходимо выводить запрос на удаление каждого файла, указанного в качестве аргумента команды `rm`, требуется использовать опцию `-i`.

Команда `rm`, вызванная без опций не задает никаких вопросов по поводу необходимости удаления файлов (по умолчанию действует опция `-f`)

При использовании шаблонов подстановки в качестве аргументов команды `rm`

настоятельно рекомендуется предварительно проверить шаблон командой `ls`.

Пример:

```
$ touch file{1,2,3}
$ ls
file1   file2   file3
$ ls f*[12] file1   file2
$ rm f*[12]
$
```

Примечание: В этом примере продемонстрирована предварительная проверка шаблона с помощью `ls` перед вызовом `rm`.

Также в целях безопасности настоятельно рекомендуется использовать псевдоним для команды `rm`, активизирующий ее интерактивный режим (опция `-i`) работы по умолчанию.

Пример:

```
$ alias rm='rm -i'
```

Примечание: Приведенный в примере псевдоним следует разместить в файле профиля пользователя `.bash_profile`, либо в `.bashrc`. Использование его включит интерактивный режим работы команды `rm` по умолчанию. Важно понимать, что затруднительно, а часто просто невозможно, восстановить удаленный файл.

Если необходимо рекурсивно удалить каталог со всем его содержимым, необходимо использовать команду `rm` с опцией `-r`

Пример:

```
$ ls -R d1 d1:
direc1   file3

d1/direc1:
Remember
$ rm -rf d1
$ ls -R d1
ls: d1: No such file or directory
```

Примечание: Каталог `d1` содержал исходно файлы и подкаталоги. После его удаления командой `rm -rf` он исчез со всем его содержимым.

Командой `rm -rf` следует пользоваться с особой осторожностью, так как установленная опция `-f` запрещает выводить предупреждающие сообщения об удалении файлов.

Примечание: Рекомендуется перед использованием такой команды переходить в каталог, в котором находится подкаталог, подлежащий удалению, и в качестве аргумента указывать команде `rm -rf` относительное имя каталога, подлежащего удалению. Этого правила следует придерживаться обязательно при работе в сеансе суперпользователя, так как он легко может повредить и даже полностью удалить всю файловую систему в результате неосторожного вызова этой команды.

Команда `mkdir dir` создает каталог с именем `dir`:

Пример:

```
$ mkdir dir1
$ cd dir1
$ ls
$ mkdir -p mydir/mydir/mydir
$ ls -R mydir mydir:
mydir

mydir/mydir:
mydir

mydir/mydir/mydir:
```

Примечание: Приведенный пример демонстрирует как с помощью команды `mkdir` был создан каталог `dir1`. Далее показано, что пользователь сделал вновь созданный каталог текущим с помощью команды `cd`. А затем, используя ключ `-p` команды `mkdir` , пользователь создал целую ветвь вложенных каталогов `mydir/mydir/mydir`. То есть ключ `-p` позволяет указывать для создания целый путь.

Команда `rmdir` позволяет удалять каталоги, если они пустые.

Пример:

```
$ mkdir emptyDir
$ ls -FR
.:
emptyDir/   mydir/

./emptyDir:

./mydir:
```

mydir/

./mydir/mydir:
mydir/

./mydir/mydir/mydir:

$ rmdir *
rmdir: `mydir': Directory not empty

Примечание: В этом примере был создан еще один каталог `emptyDir`. Затем пользователь попытался удалить оба каталога, однако, удален был лишь каталог пустой каталог `emptyDir`. Каталог `mydir` был оставлен, поскольку он не пустой.

Команда `rmdir -p` способна удалить ветвь пустых каталогов. Если в пути некоторый каталог окажется не пуст, то будут удалены все пустые каталоги в пути до первого не пустого каталога.

Пример:
$ > mydir/f1
$ rmdir -p mydir/mydir/mydir rmdir: `mydir': Directory not empty
$ ls -FR
.:
mydir/

./mydir:
f1

Примечание: В данном случае два последних в пути каталога `mydir` были удалены, так как они пустые. Первый в пути `mydir` каталог содержит файл, созданный командой `> mydir/f1`, поэтому он не был удален.

Команда `rm -rf mydir` удалит каталог со всем его содержимым.

Копирование, перемещение и переименование файлов.

Команда `cp` применяется для копирования:

1. команда cp srcFile tagFile копируетsrcFile в tagFile;

2. команда cp file1 file2 fileN dir копирует указанные файлы file1 file2 fileN в каталог dir;
3. команда `cp -r dir1 dir2` копирует каталог `dir1` в каталог `dir2` рекурсивно, делая в каталоге `dir2` подкаталог `dir1` с копиями всех файлов, содержащихся в нем.

Пример:

```
$ ls -FR
.:
mydir/

./mydir:
f1
$ cp mydir/f{1,2}
$ ls -FR
.:
mydir/

./mydir:
f1   f2
```

Примечание: С помощью команды `cp mydir/f{1,2}` создается копия файла `f1` с именем `f2` в том же каталоге `mydir`, где находится исходный файл. Команда `cp mydir/f{1,2}` эквивалентна команде `cp mydir/f1 mydir/f2` , однако она значительно короче.

Пример: копируем каталог `mydir` со всем содержимым в каталог `/tmp`.

```
$ ls mydir
$ cp -r mydir/ /tmp/
$ ls -R /tmp/mydir
/tmp/mydir:
f1   f2
```

Примечание: Пример демонстрирует, что копия каталога `mydir` является подкаталогом `/tmp`.

Команда `mv` используется для перемещения:

1. команда mv oldName newName переименовывает oldName в newName ;
2. команда `mv file1 file2 fileN dir` перемещает заданные файлы в каталог `dir` ;

3. команда `mv oldName newName` переименовывает каталог `oldName` в `newName` .

Пример: Переместим каталог `/tmp/mydir` в домашний каталог.

```
$ mv /tmp/mydir/ ~
$ ls -R ~/mydir
/home/user1/mydir:
f1   f2
```

Примечание: Команда `mv` работает с каталогами точно так же, как и с файлами, то есть для перемещения каталога не надо указывать дополнительных опций.

Пример: Ниже приведен пример, в котором перемещению подвергаются сразу несколько файлов. Для указания имен этих файлов используются символы подстановки.

```
$ mv ~/mydir/f* .
$ ls -R
.:
f1   f2   mydir

./mydir:
f1   f2
```

Примечание: Здесь два файла `f1` и `f2` из каталога `~/mydir/` перемещены в текущий каталог.

Поиск файлов.

Команда `find` позволяет производить поиск файлов по указанным критериям.

Формат команды:

find место_поиска критерии действия

где `место_поиска` – каталог, начиная с которого будет произведен поиск. Поиск также производится и в подкаталогах указанных в качестве мест поиска каталогов.

Критериями поиска могут быть любые атрибуты файла, например, имя файла, его размер, владелец файла, тип, даты доступа, модификации и прочее.

Действие по умолчанию имена найденных файлов выводятся на экране.

Примечание: Сама команда `find` не обеспечивает поиска файла по его содержимому, но ее можно использовать вместе с утилитой `grep`

Команда `find` по историческим причинам придерживается нестандартного

формата командной строки, в котором после тире указывается длинная опция с дополнительным аргументом.

Наиболее часто используют следующие опции:

1. `-name` – поиск по имени файла или строке в его имени;
2. `-iname` – то же с игнорированием регистра;
3. `-type` – поиск по типу файла;
4. `-size` – для поиска по размеру или диапазону возможных размеров файла;
5. `-perm` – поиск по правам доступа;
6. `-user` и `-group` – по принадлежности файла.

Пример: В текущем каталоге требуется найти все файлы, имя которых начинается со строки

d1.

```
$ find . -name "d1*"
./d1
```

Примечание: Обратите внимание, что если критерий поиска по подстроке в имени файла использует шаблоны подстановки, то такой шаблон должен быть экранирован кавычками. В противном случае до исполнения команды Bash заменит шаблоны на имена файлов, подходящие им, и команда будет работать неверно.

Пример: Можно ужесточить критерий поиска, потребовав от предыдущей команды отыскивать только каталоги с заданным именем:

```
$ find . -name "d1*" -type d -ls
12042    0 drwxr-xr-x 2 aberes aberes     192     Окт     7 21:58 ./d1
```

Примечание: Более того, в последнем примере использована специальная настройка для команды `find` (такой формат работает только с GNU версией этой команды), позволяющая отображать найденные файлы в формате, подобном `ls -l`.

Если критерии поиска необходимо объединить по логическому условию ИЛИ, то необходимо использовать `-o`.

Пример:

```
$ find . -name "d1*" -o -empty
./d1
./d1/*
./d1/s
./d1/f1
./d1/f2
```

./d1/s1

./d1/s2

Примечание: В этом примере продемонстрирован поиск по двум критериям, объединенным условием ИЛИ. Первый критерий – поиск файлов, начинающихся со строки `d1`, а второй – поиск пустых файлов (`-empty`).

Для поиска файлов определенного типа необходимо использовать критерий `-type тип` , где `тип` один из:

1. `b` – файл блочного устройства;
2. `c` – файл символьного устройства;
3. `d` – каталог;
4. `f` – обычный файл;
5. `p` – именованный канал;
6. `s` – сокет;
7. `l` – символьная ссылка.

Имеется возможность исполнять команды с найденными `find` файлами. Для этого необходимо использовать модификатор `-exec`

Пример:

```
$find ~ -empty -exec rm –f {} \;
```

Примечание: В этом примере производится поиск и удаление всех пустых файлов, начиная с домашнего каталога.

Смысл конструкции `команда {} \;` состоит в следующем: фигурные скобки будут замены именами найденных файлов, которые и будут аргументами команды :

```
команда файл1 команда файл2
…
команда файлN
```

Примечание: Символ обратной косой черты `\` здесь применен для экранирования символа точки с запятой от возможной неправильной интерпретации Bash. Символ точка с запятой показывает окончание командной строки.

Поиск файлов по подстроке в имени в базе данных.

В GNU/Linux используется специальная индексированная база данных `slocate`, в которую помещаются все имена всех файлов в системе.

Индексирование файлов производится на регулярной основе автоматически обычно в ночное время с помощью команды `updatedb`.

Эта база данных позволяет быстро производить поиск файла по подстроке в его имени. Данная система не воспринимает шаблоны поиска, а только строки.

Пример: Отыщем все имена файлов, содержащих строку `spice` :

```
$ locate spice
/usr/share/jed/lib/spicemod.sl
/usr/share/jed/lib/spicemod.slc
/usr/share/vim/syntax/spice.vim
```

Примечание: Команда `locate` вывела найденные имена файлов.

Определение типа файлов.

Для определения характера содержимого (типа) файла используется команда `file`, пытающаяся установить тип файла, используя так называемые магические числа (magic numbers)

Пример:

```
$ file tab_d tab_d: ASCII text
```

Примечание: Файл `tab_d`, как определила команда `file`, является текстовым файлом, содержащим текст в формате ASCII.

Перед выводом на экран незнакомого вам файла рекомендуется узнать его тип с помощью команды `file`. В противном случае на экран может быть выведен бинарный файл и настройки терминала могут быть испорчены.

Перенаправление потоков ввода-вывода.

Если программа требует чтения или записи в файл, то этот файл должен быть сначала открыт, т.е. создан поток (stream) данных.

Процедура открытия файла создает структуру в ядре, называемую файловым дескриптором.

Все файловые дескрипторы пронумерованы.

С любым пользовательским процессом при его создании связываются три файловых дескриптора (потока):

стандартный поток ввода (stdin), файловый дескриптор которого – 0;

стандартный поток вывода (stdout), файловый дескриптор – 1;

стандартный поток вывода ошибок (stderr), файловый дескриптор – 2.

Примечание: Поток ввода открыт на чтение, а потоки вывода и ошибок – на запись. Обычно по умолчанию стандартный поток ввода связан с клавиатурой, а стандартные потоки вывода и ошибок связаны с дисплеем.

Оболочка Bash позволяет перенаправить стандартные потоки в файлы, для чего используются следующие операторы:

< имя_файла или 0< имя_файла – перенаправление стандартного потока ввода;

> имя_файла или 1> имя_файла – перенаправление стандартного потока вывода;

2> имя_файла – перенаправление стандартного потока вывода ошибок

Если файла, в который производится запись не существует, то такой файл создается

Пример: с помощью такой команды можно отправить электронное письмо с текстом, содержащемся в файле letter :

$ mail -s 'Pismo' user1 < letter

Примечание: Эта команда направит электронное письмо пользователю user1*. Тема письма (Subject), указана после опции* -s*, а текст письма передан через стандартный поток ввода из файла* letter*.*

Пример: Следующая команда использует перенаправление стандартного потока вывода в файл ls.txt :

$ ls > ls.txt

Примечание: В файле ls.txt *в результате работы такой команды окажется список файлов в текущем каталоге, выведенный командой* ls *в стандартный поток вывода.*

Пример: Аналогично в файл можно перенаправить ошибки:

$ ls -ld /etc /ctc 2> ls.err

drwxr-xr-x 87 root root 6064 Окт 15 18:57 /etc

$ cat ls.err

ls: /ctc: No such file or directory

Примечание: В этом примере поток вывода ошибок был перенаправлен в файл ls.err*. В силу того, что в качестве аргумента команды* ls -ld *был задан несуществующий каталог* /ctc*, то команда вывела соответствующее сообщение об ошибке, которое и было перенаправлено в файл* ls.err*. Содержимое файла было выведено с помощью команды* cat*.*

Пример: Поток вывода перенаправлен в файл ls.txt одновременно с перенаправлением потока ошибок в файл ls.err.

```
$ ls -ld /etc /ctc > ls.txt 2> ls.err
```

Чтобы перенаправить оба потока вывода в один и тот же файл следует использовать оператор сцепления потоков &

Оболочка bash позволяет следующие эквивалентные формы записи перенаправления потоков вывода в один и тот же файл:

> имя_файла 2>&1

2> имя_файла 1>&2

>& имя_файла

&> имя_файла

Примеры:

```
$ ls -ld /etc /ctc > ls.txt 2>&1
$ ls -ld /etc /ctc 2> ls.txt >&2
$ ls -ld /etc /ctc &> ls.txt
```

Операции перенаправления потоков вывода и вывода ошибок в файл стирают его содержимое, записывая новое содержимое взамен старого.

Операцию перенаправления можно использовать для стирания содержимого файлов и создания новых пустых файлов.

Пример: для стирания содержимого файла ls.txt можно использовать команду

$ > ls.txt

Оболочка Bash позволяет исключить стирание содержимого файлов при перенаправлении в них потоков вывода или ошибок с помощью установки флага noclobber командой set -o.

Пример:

$ set -o noclobber

Значение всех флагов оболочки можно с помощью команды set -o.

Пример:

$ set -o
allexport off
braceexpand on
emacs on
errexit off
hashall on
histexpand on
history on
ignoreeof off
interactive-comments on
keyword off
monitor on
noclobber on
noexec off
noglob off
nolog off

notify on

nounset off

onecmd off

physical off

posix off

privileged off

verbose off

vi off

xtrace off

Примечание: Листинг, выведенный командой set -o*, демонстрирует, что после выполнения пользователем команды* set -o noclobber*, данная опция была активизирована (состояние* on*). Установка этой опции оболочки предотвращает перезапись существующих файлов с помощью операций вывода.*

Пример: Попробуем очистить существующий файл dir.txt.

$ > dir.txt

bash: dir.txt: cannot overwrite existing file

Примечание: Так как установлена опция noclobber*, то оболочка не позволила переписать (в данном случае стереть) существующий файл.*

Для перезаписи содержимого файла при установленной опции noclobber можно воспользоваться операторами:

>| - для перенаправления потока вывода с гарантированной перезаписью файла;

2>| - для перенаправления потока вывода ошибок с гарантированной перезаписью файла.

Пример:

$ ls -l >| dir.txt

Примечание: Эта команда запишет подробную информацию о содержимом текущего каталога в существующий файл dir.txt*, не обращая внимания на то, что файл уже существует и установлена опция оболочки* noclobber.

Отключить опцию noclobber (как и другие опции по аналогии) можно с помощью команды: set +o noclobber

Если необходимо добавить в существующий файл информацию из потоков вывода, то можно использовать следующие операторы:

>> - для перенаправления потока вывода на добавление к файлу;

2>> - для перенаправления потока вывода ошибок на добавление к файлу

Если файла не существует, то создается новый файл.

Пример: следующая команда добавит в файл dir.txt имя текущего каталога:

$ pwd >> dir.txt

Некоторые команды, работающие с текстовыми файлами, позволяют вместо имени файла указать стандартный поток ввода. Для чего используется символ -

Пример: команда

vi -

позволяет считать редактируемый текст из стандартного потока ввода вместо открытия файла.

Для завершения потока ввода, производимого с клавиатуры, следует набрать сочетание клавиш Ctrl-D, передающее в поток ввода символ конца файла.

Пример:

$ view -

Privet

<Ctrl-D>

Примечание: В этом примере команда view *позволила считать содержимое потока ввода с клавиатуры, окончание которого было обозначено с помощью сочетания* Ctrl-D.

Конструкция *here document* (документ здесь) позволяет вместо символа конца файла (EOF), который не может быть использован внутри файла, воспользоваться любым другим удобным символом.

Пример: следующая конструкция обеспечивает посылку письма пользователю user1, где тело письма передается через поток ввода команды mail с помощью here document:

```
$ mail -s HereDoc user1 << .
This is Here Document here!
.
```

Примечание: Обратите внимание, что в предыдущем примере вместо использования символа окончания потока здесь используется символ точки. Этот символ был задан в качестве ограничителя потока ввода в командной строке << . *. Это обозначает, что текст, вводимый через стандартный поток ввода, должен быть ограничен символом точка. Символ – ограничитель должен находиться в строке, перед которой чтение стандартного потока ввода прекращается. Символ – ограничитель должен быть единственным символом в строке (не считая символа перевода строки).*

Конвейеры и фильтры.

Конвейер (pipe) позволяет направить стандартный поток вывода (stdout) одного процесса в стандартный поток ввода (stdin) другого процесса.

Для организации конвейера необходимо поставить символ конвейера – вертикальную черту | между командами, потоки которых необходимо объединить.

Пример:

```
$ last | view -
```

Примечание: Выводимый командой last *список входивших в сеанс пользователей передан через конвейер команде просмотра* view *(один из вариантов вызова* vi*). Знак тире после команды* view *сообщает, что данные должны быть введены из стандартного потока ввода, в который передает данные команда* last *через свой поток вывода.*

Команда tee позволяет организовать вывод информации, полученной из потока ввода, в файлы, указанные как аргументы, и в стандартный поток вывода одновременно.

Пример: команда ps в примере ниже выведет список процессов в файл ps.txt и в стандартный поток вывода.

```
$ ps | tee ps.txt
PID TTY TIME CMD
1720 pts/0 00:00:00 bash
2438 pts/0 00:00:00 ps
2439 pts/0 00:00:00 tee
$ cat ps.txt
PID TTY TIME CMD
1720 pts/0 00:00:00 bash
2438 pts/0 00:00:00 ps
2439 pts/0 00:00:00 tee
```

Примечание: Содержимое файла ps.txt *совершенно идентично выводу команды* ps*. Обратите внимание на присутствие в списке процессов команды* tee*.*

Примечание: Команда tee *наиболее часто используется для отладки работы сложных конвейеров, состоящих из множества команд. Эту команду удобно устанавливать в месте конвейера, которое вызывает подозрения. Так как в файле, указанном в качестве аргумента* tee*, будет находиться та же информация, что была передана в данном месте конвейера, то по ней легко можно будет определить наличие и суть ошибок.*

Фильтр – это не интерактивная команда, способная принимать поток данных через стандартный поток ввода, обрабатывать его, и выводить обработанные данные в стандартный поток вывода.

Пример: команда wc -l – фильтр, подсчитывающий количество строк в потоке ввода и передающий результат подсчета в поток стандартного вывода. Конвейер, показанный ниже, подсчитывает число процессов, работающих в системе от имени пользователя user1.

```
$ ps -u user1 | wc -l
8
```

Команды, находящиеся в конвейере должны удовлетворять следующим требованиям:

Первая команда конвейера должна уметь выводить информацию в стандартный поток вывода.

Команды, находящиеся внутри конвейера, должны быть фильтрами.

Последняя команда в конвейере должна уметь читать стандартный поток ввода.

Пример: для отправки суперпользователю письма с отсортированным списком пользователей, вошедших в сеанс, можно использовать такой конвейер:

```
$ who | sort | mail -s 'Logged users' root
```

Примечание: В этом примере команды who *и* mail *не являются фильтрами, но поскольку команда* who *осуществляет вывод в стандартный поток вывода, а* mail *читает поток ввода, то их можно использовать, соответственно, в начале и конце конвейера. Команда* sort*, осуществляющая сортировку, является фильтром, то есть она читает из стандартного потока ввода и выводит отсортированный текст в стандартный поток вывода, поэтому она вполне может находиться где-либо в середине конвейера.*

Общепринятые соглашения об именовании файлов.

Несмотря на то, что при именовании файлов в Linux можно использовать практически любые символы кроме косой черты / , рекомендуется использовать алфавитно-цифровые символы, символ подчеркивания, точку и дефис.

В отличие от многих операционных систем, например, MS DOS, Linux не придает особого значения точке в имени файла. Тем не менее, пользоваться точками в именах файлов удобно для указания после нее суффикса имени файла, показывающего тип данных, хранящихся в данном файле.

Ниже перечислены часто используемые общепринятые суффиксы, которые не стоит без особой цели использовать для файлов с другим типом содержимого.

Суффикс	Содержимое файла

.c	Исходный код программы на языке C.
.cpp	Исходный код программы на языке C++.
.h	Заголовочный файл для программы на C или C++ (header).
.o	Объектный код.
.a	Статическая библиотека.
.so	Разделяемая библиотека (shared object).
.sh	Shell скрипт.
.csh	C Shell скрипт.
.pl	Программа на Perl.
.pm	Скомпилированная в байт-код программа на Perl.
.py	Программа на Python.
.rpm	Программный пакет в формате RPM (Red Hat Package Manager).
.src.rpm	Пакет с исходным кодом в формате RPM.
.deb	Программный пакет в формате Debian.
.tar	Архив tar (tape archive).
.cpio	Архив cpio.
.gz	Файл, сжатый gzip.
.bz2	Файл, сжатый bzip2.
.Z	Файл, сжатый compress.

Специальные файлы в Linux.

В GNU/Linux используются следующие типы специальных файлов (в столбце обозначение приводятся символы из первого столбца вывода команды `ls -l`, обозначающие тип файла):

Обозначение	Тип файла	Назначение
-	Обычный файл.	Для хранения данных.

d	Каталог.	Организует древовидную файловую структуру.
l	Символьная ссылка.	Указывает на другой файл.
b	Блочное устройство.	Обеспечивает доступ к блочным устройствам.
c	Символьное устройство.	Обеспечивает доступ к символьным устройствам.
p	Именованный канал.	Передает поток от одного процесса к другому.
s	Сокет.	Для передачи данных по сетевому протоколу.

Специальные файлы, не являющихся обычными обрабатываются ядром операционной системой особым образом.

Примечание: Так, например, операционная система не пытается вывести на экран двоичное содержимое файла каталога, вместо этого при обращении к нему с помощью ls *отображаются имена файлов в нем в понятной человеку форме.*

В файле символьной ссылки находится строка, представляющая собой имя файла, на которое эта ссылка указывает.

Примечание: Когда пользователь обращается к символьной ссылке операционная система вместо того, чтобы показывать ее содержимое, на самом деле обращается к файлу, на который ссылка указывает.

Пример:

```
$ ls -l doc

lrwxrwxrwx  1 prof  prof 14 Май  1  2003 doc -> /usr/share/doc
$ cd doc

$ pwd
/home/prof/doc
```

Примечание: В этом примере с помощью команды `ls -l` *демонстрируется, что файл символьной ссылки* `doc` *указывает на каталог* `/usr/share/doc` *. Несмотря на то, что* `doc` *является не каталогом, а ссылкой на каталог, можно сделать команду* `cd` *и оказаться в каталоге, на который ссылка указывает. При этом создается полная иллюзия, будто бы был осуществлен переход в каталог* `doc`*, что демонстрирует команда* `pwd`*.*

Специальные файлы блочные устройства предназначены для обращения к блочным устройствам, то есть к тем, с которых информация считывается и записывается блоками строго определенной длины.

Примечание: Такие устройства требуют форматирования, и к этой категории относятся, например, магнитные и лазерные диски.

Специальные файлы символьных устройств предназначены для обращения к устройствам, которые принимают и передают информацию посимвольно, а не блоками.

Примечание: Примерами таких устройств являются терминалы, мыши, модемы и т.п.

Специальные файлы блочных и символьных устройств – пустые файлы.

Вместо размера для файлов устройств в индексном дескрипторе хранятся два числа – мажор (major number, старшее число) и минор (minor number, младшее число).

Пример:

```
$ ls -l /dev/fd0 /dev/tty1
brw-rw----    1 prof     floppy    2,   0 Янв 20   2003 /dev/fd0
crw-------    1 root       root    4,   1 Дек 14 23:55 /dev/tty1
```

Примечание: Из листинга, приведенного выше, заметно, что мажор файла блочного устройства (символ `b` *в первом столбце) флоппи диска* `/dev/fd0` *равен* `2`*, а минор -* `0`*. А для символьного файла устройства (символ* `c` *в первом столбце) первого виртуального терминала* `/dev/tty1` *мажор -* `4`*, минор -* `1`*.*

Мажор файла устройства соответствует номеру драйвера устройства в ядре операционной системы

Минор - дополнительный параметр для драйвера, который указывает с каким именно устройством нужно работать драйверу.

Файлы именованных каналов предназначены для передачи потока вывода одного процесса на поток ввода другого процесса.

Примечание: Если один процесс записывает какие-либо данные в именованный канал, то его работа прерывается до тех пор, пока другой процесс не считает требуемые данные из этого именованного канала. Именованные каналы были введены в BSD.

Сокеты являются файлами, подобными именованным каналам, однако являются двунаправленными и могут быть как локальными, так и сетевыми.

Использование жестких связей.

Жесткая связь (hard link) образуется, когда несколько имен файлов указывают на один и тот же индексный дескриптор (inode).

Команда `ls -l` во втором столбце показывает количество имен у файла (link count).

Примечание: Если у файла link count имеет значение, большее единицы, значит у файла имеются жесткие связи.

Пример:

```
$ ls -l f1 link1
-rw-r--r--          2 tania          tania       8 Окт 22 21:04 f1
-rw-r--r--          2 tania          tania       8 Окт 22 21:04 link1
$ ls -i f1 link1
  13598 f1 13598 link1
```

Примечание: Здесь между файлами `f1` *и* `link1` *существует жесткая связь, то есть это два разных имени файла. Команда* `ls -l` *демонстрирует, что количество имен у файла - два (цифра во*

втором столбце вывода `ls -l`*), права доступа и владения, размер файла и дата модификации совершенно одинаковы, поскольку метаданные одни и те же. Команда* `ls -i` *показывает, что номера* `inode` *у этих файлов одинаковы.*

Поскольку метаданные у файлов, между которыми установлена жесткая связь, одинаковы, то и блоки данных тоже одинаковы, следовательно любые изменения, производимые с одним файлом, зеркально отразятся на файле, жестко связанным с ним.

Фактически жесткая связь это один файл с несколькими именами.

Пример:

```
$ echo 'This is a hard link' > f1
$ cat link1

This is a hard link
```

Примечание: В этом примере информация в файл была записана с использованием имени файла `f1`*, а чтение информации было произведено из* `link1`*.*

Удаление одной жесткой связи с файлом уменьшает количество имен (link count) на единицу, однако реально файл удаляется только тогда, когда счетчик количества имен становится равным нулю.

Пример:

```
$ ls -l f1 link1

-rw-rw-rw-          2 tania        tania 20 Дек 15 22:03 f1
-rw-rw-rw-          2 tania        tania 20 Дек 15 22:03 link1


$ rm -f f1


$ ls -l f1 link1

ls: f1: No such file or directory
-rw-rw-rw-          1 tania        tania 20 Дек 15 22:03 link1
```

Примечание: Пример, приведенный выше, демонстрирует, что при удалении одного из имен файла счетчик имен уменьшился на единицу.

Для создания жесткой связи с файлом применяется команда ln. Первый аргумент команды - имя файла, для которого создается новое имя, а второй - имя жесткой связи.

Пример:

```
$ ln file1 newname
$ ls -li file1

  4676 -rw-rw-r--       2 prof        prof    0 Дек 14 20:43 file1
$ ls -li newname

  4676 -rw-rw-r--       2 prof        prof    0 Дек 14 20:43 newname
```

Примечание: В этом примере создано новое имя для файла `file1 -newname`. *Комада* `ls -li` *демонстрирует, что inode у этих файлов одинаковые.*

Жесткие связи могут быть установлены только в пределах одной файловой системы.

Примечание: То есть нельзя установить жесткую связь между файлом на жестком диске и, например, дискете. Это вызвано тем, что у жестко связанных файлов должен быть один и тот же индексный дескриптор, что невозможно если файловые системы разные.

Установить жесткую связь с каталогом с помощью команды `ln` нельзя.

Пример:

```
$ ls -ld LPI
drwxrwx---       10 prof       users    544 Сен  2 10:15 LPI
$ ln LPI l1
ln: `LPI': не допускается создавать жесткие ссылки на каталоги
```

У каждого каталога имеется как минимум два имени - имя каталога, записанное в его родительском каталоге, и имя точка.

У родительского каталога после создания нового подкаталога количество имен увеличивается на единицу, так как в дочернем каталоге имеется имя две точки, являющееся жесткой связью с именем родительского каталога

Пример:

```
$ ls -ldi
3555 drwx------     16 user1      user1  1120 Дек 15 22:15 .

$ mkdir dir1

$ ls -ild dir1
1598 drwxr-x---     2 user1       user1  48 Дек 15 23:03 dir1

$ ls -ial dir1
итого 2
1598 drwxr-x---     2 user1       user1    48 Дек  15   23:03 .
3555 drwx------    17 user1       user1  1144 Дек  15   23:03 ..

$ ls -ldi
 3555 drwx------         17 user1     user1   1144 Дек 15 23:03 .
```

Примечание: В этом примере был создан каталог `dir1` *, inode которого* `1598`*. Имя точка, находящееся в каталоге* `dir1` *- имя текущего каталога, имеет тот же индексный дескриптор* `1598`*. Таким образом, у нового каталога* `dir1` *имеется два имени. Имя родительского каталога - две точки, находящиеся в каталоге* `dir1` *, имеет inode* `3555`*, значение которого совпадает с inode родительского каталога. Сравнив вывод первой и последней команд, становится заметно, что после создания каталога* `dir1` *количество имен у его родительского каталога увеличилось на единицу.*

Команда `cp` имеет специальную опцию `-l` , которая позволяет (в пределах одной файловой системы) вместо копирования создавать жесткие связи с исходными файлами.

Пример:

```
$ ls -li f???
4629 -rw-r--r--  1 user1  user1  0 Дек 11 12:30 f112
4631 -rw-r--r--  1 user1  user1  0 Дек 11 12:30 f113
4632 -rw-r--r--  1 user1  user1  0 Дек 11 12:30 f117
$ cp -l f??? dir1
$ ls -li dir1
итого 0
4629 -rw-r--r--  2 user1  user1  0 Дек 11 12:30 f112
4631 -rw-r--r--  2 user1  user1  0 Дек 11 12:30 f113
4632 -rw-r--r--  2 user1  user1  0 Дек 11 12:30 f117
```

Примечание: В этом примере показано, что при использовании команды `cp` *с опцией* `-l` *для исходных файлов были созданы жесткие связи с такими же именами в каталоге* `dir1`*.*

Использование символьных ссылок.

Символьные ссылки - это специальный тип файлов, представляющий собой указатели на другие файлы.

Символьные ссылки можно создавать с помощью команды `ln -s` , где первый аргумент - имя уже существующего файла, а второй аргумент - либо имя символьной ссылки, либо каталога, где будет образован файл символьной ссылки с таким же именем, что и исходный файл.

Пример:

```
$ ln -s file1 slink1

$ ls -li file1 slink1
4639 -rw-rw-r--  1 user1  user1  0 Дек 11 15:55 file1
1604 lrwxrwxrwx  1 user1  user1  5 Дек 16 01:37 slink1 -> file1

$ echo 123 > file1
```

$ cat slink1 123

Примечание: В этом примере на файл `file1` *создана символьная ссылка* `slink1` *. Метаданные у* `file1` *и* `slink1` *различаются, поскольку это совершенно разные файлы. Тем не менее, к файлу можно обращаться с помощью ссылки.*

Символьные ссылки можно устанавливать на каталоги

Пример:

```
$ ln -s ~/dir1 /tmp

$ ls -ldi dir1 /tmp/dir1
  4635 drwxr-xr-x  2 user1 user1   144 Дек 16 00:26 dir1
163286 lrwxrwxrwx  1 user1 user1    16 Дек 16 01:12 /tmp/dir1 ->
/home/user1/dir1
```

Примечание: В этом примере создается символьная ссылка на каталог `dir1` *в каталоге* `/tmp` *. Заметно, что символьная ссылка имеет совершенно иной тип файла и номер inode, местоположение ее не ограничивается пределами одной файловой системы.*

При создании символьной ссылки в другом каталоге следует указывать полное имя исходного файла. Иначе:

- либо ссылка будет оборванной (или иначе – висящей, broken link), то есть символьная ссылка будет указывать на несуществующий файл;
- либо ссылка будет указывать на существующий файл в целевом каталоге, имя которого (случайно или намеренно) совпадет с исходным файлом, однако, ссылка будет указывать не на исходный файл.

Символьная ссылка может оказаться оборванной в случае, если:

- файл, на который она указывает перемещен, переименован или удален;
- нет достаточных прав доступа на файл, указываемый символьной ссылкой;
- файл находится в файловой системе, которая в настоящий момент не смонтирована.

Пример:

```
$ mv file1 file11
$ cat slink1
cat: slink1: No such file or directory
```

Примечание: После переименования файла `file1` *в* `file11` *ссылка* `slink1` *стала оборванной.*

Символьная ссылка может быть создана на любой специальный файл.

Пример:

```
$ ln -s /dev/fd0 .
$ ls -l fd0
lrwxrwxrwx          1 user1          user1       8 Дек 16 01:13 fd0 -> /dev/fd0
```

Размер файла символьной ссылки равна длине имени файла, на который она указывает.

Команда `cp` обладает специальной опцией `-s` , которая позволяет вместо копирования файлов создавать на них символьные ссылки

Пример:

```
$ cp -s ~/f??? /tmp
$ ls -l /tmp/f???
lrwxrwxrwx 1 user1 user1 16 Дек 16 01:52 /tmp/f112 ->
/home/user1/f112
lrwxrwxrwx 1 user1 user1 16 Дек 16 01:52 /tmp/f113 ->
/home/user1/f113
lrwxrwxrwx 1 user1 user1 16 Дек 16 01:52 /tmp/f117 ->
/home/user1/f117
lrwxrwxrwx 1 user1 user1 16 Дек 16 01:52 /tmp/f122 ->
/home/user1/f122
```

Примечание: В этом примере на группу файлов были созданы символьные ссылки в каталоге `/tmp`.

Если используется команда `cp` , для которой в качестве аргумента указаны файлы символьных ссылок, то будут скопированы не файлы символьных ссылок, а файлы- оригиналы, на которые они указывают.

Если же необходимо скопировать не исходные файлы, а именно символьные ссылки, то следует использовать опцию `-d` команды `cp`

Пример:

```
$ cp -d /tmp/f122 Documents/
$ ls -l Documents/f122
lrwxrwxrwx          1 user1        user1 16 Дек 16 02:00 Documents/f122 ->
/home/user1/f122
```

Модуль 5. Пользователи и дискретные права доступа в РЕД ОС.

Linux – многопользовательская операционная система. Пользователь определяется именем пользователи (username) и обладает личной частью системы, которую он может использовать. Существует специально определенный пользователь с именем root, (суперпользователь), который имеет право осуществлять в системе любые операции. !!! Назначив пользователю права на администрирование ОС (права суперпользователя root), то по факту такой пользователь не становится суперпользователем во всех смыслах данного понятия

К понятию «права доступа к объектам файловой системы» в стандартной парадигме Linux можно отнести владение объектом и режим доступа. Любой объект файловой системы обязательно должен иметь владельца, то есть пользователя, которому доступны все действия над содержимым и метаданными файла, если не имеется каких-либо иных ограничений на выполнение данных операций. При создании файла владельцем автоматически назначается пользователь, который создал файл

Суперпользователь (администратор): root Идентификатор пользователя: 0 Группа root с идентификатором 0. Сеанс суперпользователя — su Выполнение команды от имени root — sudo.

sudo useradd -m -d /home/test test

sudo -u test mkdir /home/test/newdir

Работать от имени пользователя root небезопасно и поэтому не рекомендуется root только для администрирования системы

Хранение учетных записей пользователей.

Аутентификация — системная процедура, позволяющая однозначно определить пользователя.

В GNU/Linux процедура аутентификации пользователя при входе в сеанс может быть проведена разными способами. Вот некоторые из них:

Традиционный способ, опирающийся на база данных учетных записей в файлах

/etc/passwd и /etc/shadow.

Аутентификация с помощью системы Kerberos.

Аутентификация в NIS/NIS+.

Аутентификация в LDAP.

Использование специализированных систем аутентификации (например,

TCB) и т.п.

Не смотря на наличие различных систем аутентификации наиболее простым, а следовательно, и наиболее распространенным способом является традиционный, использующий текстовые файлы учетных записей пользователей.

Файл `/etc/passwd` для аутентификации пользователей используется уже давно, а файл

`/etc/shadow` стал использоваться с появлением системы теневых паролей.

Каждая запись в этих файлах соответствует конкретному пользователю системы.

Поля записей разделены двоеточиями.

Структура записей в файле `/etc/passwd` следующая:

Первое поле – имя пользователя.

Второе поле содержит символ `x` если используется система теневых паролей

`/etc/shadow`. Если эта система не используется, то во втором поле находится зашифрованный пароль пользователя.

Третье поле – UID пользователя.

Четвертое поле – GID пользователя.

Пятое поле содержит необязательную справочную информацию о пользователе, например, его обычное (человеческое) имя.

Шестое поле указывает домашний каталог пользователя.

Седьмое поле соответствует имени исполняемого файла оболочки, запускаемой при входе в сеанс для этого пользователя.

Права доступа, устанавливаемые на файл `/etc/passwd` позволяют читать этот файл всем пользователям. Поэтому при хранении шифрованного пароля во втором поле этого файла представляет реальную угрозу безопасности, так как любой злоумышленник, имеющий доступ к данной системе может воспользоваться программами подбора паролей для взлома системы.

Использование системы теневых паролей существенно снижает эту опасность, так как файл, где хранятся шифрованные пароли - `/etc/shadow` не позволяет его читать никому, кроме суперпользователя. Структура этого файла такова:

Первое поле – имя пользователя.

Второе поле – зашифрованный пароль.

Третье поле – количество дней с 01 января 1970 г., прошедших с момента последней смены пароля.

Четвертое поле – количество дней, которые должны пройти с момента последней смены пароля пользователя, прежде чем он сможет снова поменять пароль.

Пятое поле – срок устаревания пароля в днях с момента его смены. До истечения этого срока пароль обязательно должен быть изменен.

Шестое поле – время в днях до момента устаревания пароля, начиная с которого пользователь будет получать предупреждения о необходимости очередной смены пароля.

Седьмое поле – период времени в днях с момента устаревания пароля, по прошествии которого учетная запись пользователя будет заблокирована по причине устаревания пароля.

Восьмое поле устанавливает срок жизни учетной записи и предназначен для создания временных учетных записей. Оно содержит число дней с 01 января 1970 г., по прошествии которых учетная запись будет заблокирована вне зависимости от состояния пароля пользователя.

Девятое поле зарезервировано и в настоящее время не используется.

Регистрация, удаление и блокирование учетных записей пользователей.

Правами регистрации пользователей в системе обладает суперпользователь.

Для добавления учетной записи пользователя используется команда `useradd`. В качестве аргумента для этой команды должно быть указано имя пользователя.

Пример:

useradd user1 # id user1

uid=504(user1) gid=504(user1) groups=504(user1)

Примечание: В этом примере в системе был зарегистрирован новый пользователь – `user1`. Для него была зарегистрирована его приватная группа пользователей – `user1`. Приватная группа пользователей состоит из единственного пользователя. Она является первичной группой для этого пользователя.

Регистрация пользователя приводит к появлению соответствующих записей в файлах

/etc/passwd и /etc/shadow .

Пример:

grep user1 /etc/passwd user1:x:504:504::/home/user1:/bin/bash

grep user1 /etc/shadow user1:!!:12407::::::

Примечание: Также для этого пользователя был создан его домашний каталог:

ls -a /home/user1

. .bash_logout .bashrc .i18n .mutt .rpmmacros Documents

.. .bash_profile .emacs .lpoptions .pinerc .xsession.d tmp

Создание приватной группы и домашнему каталога для пользователя характерно для Red Hat Linux и подобных дистрибутивов.

В Red Hat создание приватной группы можно запретить, используя опцию -n. При этом для вновь зарегистрированных пользователей в системе будет установлена группа по умолчанию (см. ниже) в качестве первичной группы.

Отсутствие домашнего каталога может препятствовать входу пользователя в систему.

Настройки для команды `useradd` находятся в файле `/etc/default/useradd` и могут быть получены с помощью опции `-D` команды `useradd` .

Пример:

useradd -D GROUP=100

HOME=/home INACTIVE=-1 EXPIRE=

SHELL=/bin/bash SKEL=/etc/skel

Примечание: Выведенная информация командой `useradd -D` говорит о следующем:

1. GROUP=100 - GID для вновь регистрируемых пользователей - 100 (для Red Hat эта настройка игнорируется при создании приватной группы пользователя);
2. HOME=/home - домашние каталоги для пользователей будут создаваться в каталоге /home;
3. INACTIVE =-1 – блокирование учетной записи пользователя при устаревании его пароля не будет;
4. SHELL=/bin/bash – оболочка для вновь регистрируемых пользователей по умолчанию;
5. SKEL=/etc/skel – каталог “скелета”, из которого в домашние каталоги вновь регистрируемых пользователей копируются файлы, необходимые для

каждого пользователя.

6. Каталог /etc/skel обычно содержит файлы профиля для вновь регистрируемых пользователей и другие служебные файлы, которые копируются при регистрации пользователя в его домашний каталог.

Пример:

ls -a /etc/skel

. .bash_logout .bashrc .mutt .xsession.d tmp

.. .bash_profile .lpoptions .rpmmacros Documents

Наиболее часто используемые опции команды `useradd` :

`-s` – указывает исполняемый файл оболочки по умолчанию;

`-d` – путь к домашнему каталогу пользователя;

`-m` – опция, указывающая на необходимость создать домашний каталог;

`-M` – не создавать домашний каталог;

`-k` – путь к альтернативному каталогу скелета;

`-u` – назначить UID пользователю;

`-g` – назначить GID (первичную группу) пользователю;

`-G` – список групп, в которых принимает участие пользователь (разделены запятыми);

`-e` – календарная дата, после которой учетная запись будет заблокирована (срок жизни учетной записи);

10.`-f` – количество дней, которое должно пройти после срока устаревания пароля до блокировки учетной записи.

Пример:

useradd -M -g sinix -s /bin/nologin -e 01-01-2004 sinixuser

id sinixuser

uid=505(sinixuser) gid=107(sinix) groups=107(sinix)

grep sinixuser /etc/shadow sinixuser:!!:12409:::::13326:

Примечание: Приведенная выше команда регистрирует пользователя без создания для него домашнего каталога (опция `-M`) с первичной группой `sinix` (опция `-g`). Для этого пользователя запрещен вход в сеанс, так как в качестве оболочки для этого пользователя указан файл `/bin/nologin` . Учетная запись пользователя будет заблокирована 1 января 2004 г. (опция `-e`).Последняя команда примера демонстрирует запись в файле

теневых паролей для пользователя `sinixuser`. Восьмое поле записи содержит число дней с 1 января 1970 г. до дня, когда учетная запись пользователя будет заблокирована.

Если пользователь не имеет право входить в сеанс, то в качестве оболочки должен быть установлен один из следующих вариантов:

1. `/bin/false` – системная команда, всегда возвращающая код 1 (код ошибочного завершения);
2. `/dev/null` – специальный файл символьного устройства, при попытке запуска которого возникает ошибка;
3. `/sbin/nologin` – системная команда, возвращающая при запуске код ошибки и сообщение о невозможности входа в сеанс.

Если необходимо произвести некоторые изменения в учетной записи уже зарегистрированного пользователя, то для этого предназначена команда `usermod` .

Пример: смена оболочки по умолчанию для пользователя:

```
# usermod -s /bin/false sinixuser
```

Большая часть опций команд `useradd` и `usermod` совпадают.

Пример: для добавления новой группы, в которых участвует пользователь, можно использовать следующую команду:

```
# usermod -G "`id -G sinixuser | tr ' ',`,users" sinixuser
```

```
# id sinixuser
uid=505(sinixuser) gid=107(sinix) groups=107(sinix),100(users)
```

Примечание: Команда `id -G` выводит список групп, в которые входит пользователь, разделенных пробелами. Далее пробелы заменяются запятыми с помощью команды `tr`, так как список групп для команды `usermod` должен быть задан через запятую. К списку групп, в которых пользователь уже принимает участие, добавляется список новых групп, а далее полученный список подставляется в командную строку `usermod` с помощью командной подстановки.

Используя команду `usermod` можно также указать для пользователя его новое имя с помощью опции `-l`.

Опции `-L` и `-U` позволяют, соответственно, блокировать и разблокировать возможность входа в сеанс для данного пользователя.

Для удаления учетной записи пользователя следует воспользоваться командой `userdel`.

Пример:

```
# userdel sinixuser # id sinixuser
```

id: sinixuser: No such user

Перед удалением учетной записи пользователя необходимо решить, что делать с файлами пользователя, если таковые в системе имеются.

Сама команда `userdel` удаления файлов пользователя не производит. Поэтому все файлы пользователя, учетная запись которого подлежит удалению, должны быть найдены и либо удалены, либо сохранены в архив, либо переданы другому пользователю.

Пример:

```
#find / -user 505 -exec rm -rf {} \;
```

Примечание: Здесь вместо имени пользователя использовался UID, поскольку пользователь с таким UID был уже удален.

Управление паролями.

Правила установки, использования и управления паролями являются важнейшей частью системной политики. Обычно они включают в себя, как минимум, следующее:

1. Определение категорий пользователей, которые имеют право самостоятельного выбора паролей с помощью команды `passwd`.
2. Правила выбора паролей и требования к их уровню сложности.
3. Сроки устаревания паролей.
4. Длительности периодов запрета на изменение паролей.

Четко сформулированная политика управления паролями значительно облегчает администрирование системы.

Для установки правила выбора паролей, их минимальной длины и требуемого уровня сложности, достаточно настроить модуль контроля паролей системы PAM для автоматической проверки соответствия выбираемого пользователем пароля системной политике.

Настройки PAM для команды `passwd` обычно находятся в файлах /etc/pam.d/passwd и, возможно, в /etc/pam.d/system-auth.

Команда `passwd` помимо изменения паролей предоставляет и другие возможности. Ниже приведен список наиболее часто применяемых опций команды `passwd` :

1. `-l` – блокирование учетной записи;

2. `-u` – разблокирование учетной записи;
3. `-S` – получение текущего состояния пароля;
4. `-d` – удалить пароль;
5. `-n` – установка периода запрета на смену пароля (минимальное время жизни пароля);
6. `-x` – установка максимального срока использования пароля;
7. `-w` – установка количества дней до момента устаревания пароля, начиная с которого пользователю будут выдаваться предупреждения о необходимости смены пароля;
8. `-i` – срок после устаревания пароля, по прошествии которого учетная запись блокируется.

Пример: блокирование учетной записи.

```
# id colobok
uid=503(colobok) gid=503(colobok) groups=503(colobok),22(cdrom)
```

cat /etc/shadow colobok:
$2a$08$Z4jZgPzM2GDVguFd4TRF3ubB.XWe4HHysgpYRwdogLj9WLlBXOJ:12316:
::::: # passwd -l colobok

```
# cat /etc/shadow
```

colobok:!
$2a$08$Z4jZgPzM2GDVguFd4TRF3ubB.XWe4HHysgpYRwdogLj9WLlBXOJ:12316:
:::::

Примечание: После блокирования учетной записи в первой позиции второго поля файла

`/etc/shadow` перед шифрованным паролем пользователя появляется знак восклицания. При разблокировании учетной записи он исчезает.

Обычно команда `chage`, которая управляет параметрами устаревания пароля, удобней чем `passwd`.

Пример: установка времени жизни учетной записи:

chage -l test

Last password change : Feb 10, 2021 Password expires : never

Password inactive : never

Account expires : never

Minimum number of days between password change : 0 Maximum number of days between password change: 99999 Number of days of warning before password expires : 7

chage **-E "01 May 2021"** test # chage -l test

Last password change : Feb 10, 2021 Password expires : never

Password inactive : never

Account expires : May 01, 2021

Minimum number of days between password change : 0 Maximum number of days between password change: 99999 Number of days of warning before password expires : 7

Управление группами пользователей.

С помощью создания групп пользователей системный администратор может эффективно управлять деятельностью в системе целыми коллективами пользователей, предоставляя им разрешения на доступ к системным ресурсам.

Примечание: Каждый файл располагает в метаданных триадой бит, кодирующей права доступа для группы пользователей. Следовательно, изменяя членство пользователя в группах, администратор изменяет, таким образом, привилегии пользователя на доступ к различным файлам в системе, не меняя при этом права пользователя на принадлежащие ему файлы.

Информация о группах пользователей хранится в файле `/etc/group` в виде строк.

Формат записи: name:password:GID:user(s), где:

1. Первое поле – имя группы.
2. Второе поле – пароль группы. Если он не используется, в этом поле ставится звездочка.
3. Третье поле - GID группы.
4. Четвертое поле содержит список пользователей, принадлежащих к данной группе, разделенных запятыми.

Для добавления новой группы необходимо воспользоваться командой `groupadd` , которая добавляет новую запись в файл `/etc/group` .

Пример:

```
# groupadd class
# grep class /etc/group
class:x:505:
```

Примечание: В этом примере добавлена новая группа `class`.

Пользователи для которых группа является первичной имеют информацию об этом в GID, который хранится в четвертом поле файла `/etc/passwd`.

Имена пользователей, входящих в группу, которая не является для них первичной, записываются через запятую в четвертом поле файла `/etc/group` .

Пример:

```
# grep sys /etc/group sys:x:3:root,bin,adm
```

Примечание: В группу `sys` в данном примере входят пользователи `root`, `bin` и `adm`.

Для явного указания идентификатора группы необходимо воспользоваться опцией `-g`.

Пример:

```
# groupadd –g 512 project
```

Примечание: В этом примере создана новая группа пользователей `project`, имеющая GID `512`.

Для удаления группы необходимо воспользоваться командой `groupdel`.

Пример:

```
# groupdel project
```

Примечание: В этом примере удалена группа пользователей `project`.

Группа пользователей может быть создана для работы над каким-либо проектом. В таком случае бывает удобно одного из пользователей сделать администратором группы и делегировать ему право добавлять уже зарегистрированных в системе пользователей в эту группу и удалять их из группы при необходимости.

Для назначения администратора группы суперпользователю необходимо использовать команду `gpasswd –A`

Пример:

```
# gpasswd –A ivanov developers
```

Примечание: Здесь администратором группы `developers` назначается пользователь `ivanov`.

Системный администратор или администратор группы может добавлять пользователей в группу с помощью команды `gpasswd -a`.

Пример:

gpasswd –a marta developers

Примечание: Пользователь `marta` была добавлена в группу `developers` системным администратором.

Удалить пользователя из группы можно командой `gpasswd -d`.

Пример:

$ gpasswd –a semko developers

Примечание: В этом примере администратор группы `developers` добавил в группу нового члена – пользователя `semko`.

Профили пользователей.

При входе пользователей в сеанс автоматически выполняются специальные файлы сценариев, называемые профилями пользователей.

Обычный подход к хранению настроек оболочки состоит в разделении настроек (профилей) на глобальный профиль (Master Profile) и пользовательские профили (Login Profiles).

Кроме профилей имеются еще и специальные файлы настроек оболочек (resource files), которые также являются сценариями оболочек.

Отличие профилей от файлов ресурсов состоит в том, что сценарии профилей исполняются единожды при входе пользователя в сеанс, а файлы ресурсов запускаются при каждом запуске оболочки.

Если оболочка Bash запущена интерактивно при входе пользователя в сеанс (то есть является оболочкой по умолчанию), то сначала исполняется общий для всех пользователей файл `/etc/profile` , а затем индивидуальный профиль пользователя, находящийся в его домашнем каталоге.

Для оболочки Bash индивидуальный профиль находится в файле, который может называться одним из следующих имен:

1. ~/.bash_profile
2. ~/.bash_login
3. ~/.profile

В файлах профилей чаще всего устанавливаются такие переменные окружения, как:

1. PATH - имена каталогов, в которых осуществляется поиск исполняемых файлов для запуска;
2. TERM - тип терминала;

3. USER - имя пользователя (устанавливается с помощью id -un);
4. HOME - путь к домашнему каталогу пользователя;
5. MAIL - путь к почтовому ящику пользователя;
6. HOSTNAME - имя системы.

Переменные окружения, устанавливаемые в файлах профилей, должны быть экспортированы с помощью команды `export`.

Пример: к списку каталогов в переменной окружения `PATH` добавляется каталог `bin`, находящийся в домашнем каталоге пользователя:

PATH=$PATH:$HOME/bin export PATH

Примечание: Имена каталогов, содержащихся в переменной `PATH` разделяются двоеточиями.

Помимо переменных окружения в файлах профиля часто устанавливается значение

umask.

При необходимости исполнить файл профиля из командной строки следует использовать команду `source .`

Пример:

source /etc/profile

Примечание: Эта команда является встроенной и выполняет в текущей оболочке команды из файла, указанного в качестве аргумента.

В противоположность профилям файл ресурсов оболочки `~/.bashrc` выполняется только при интерактивном запуске оболочки Bash из командной строки, а не при входе в сеанс.

Для того, чтобы дополнительные настройки оболочки срабатывали не только при запуске оболочки из командной строки (то есть из уже запущенной оболочки), но и при запуске Bash по умолчанию при входе в сеанс, вызов инструкций в файле `~/.bashrc` производится из пользовательского профиля.

Пример: Типичное содержимое файла пользовательского профиля таково:

$ cat .bash_profile # .bash_profile

Get the aliases and functions if [-f ~/.bashrc]; then

. ~/.bashrc

fi

User specific environment and startup programs BASH_ENV=$HOME/.bashrc

export BASH_ENV

Примечание: Здесь приведен пример содержимого файла пользовательского профиля, в котором проверяется наличие в домашнем каталоге пользователя файла ресурсов оболочки и, если он есть, содержимое его выполняется в контексте текущей оболочки. Это достигается с помощью так называемой inline подстановки - команды точка (.). Вызов `. ~/.bashrc` приводит к тому (обратите внимание на точку, с которой начинается команда), что переменные, псевдонимы и функции, определенные в файле ресурсов будут доступны в текущей оболочке. Inline подстановка всегда используется для передачи из одного файла сценария оболочки в другой скрипт переменных, псевдонимов и функций.

Переменная окружения `BASH_ENV` , определенная приведенном примере, предназначена для информирования оболочки, запускаемой не интерактивно (например, для выполнения сценария), что должны быть использованы ресурсы, определенные в файле, имя которого содержится в этой переменной.

Довольно часто в файле `/etc/profile` находится `inline` вызов общесистемного файла ресурсов `/etc/bashrc` (или `/etc/bash.bashrc`).

Примечание: Это не обязательно, но очень удобно, так как в этом файле можно определить, например, псевдонимы для команд, которыми часто пользуются различные пользователи системы, вместо определения этих псевдонимов в частных файлов ресурсов оболочки `~/.bashrc`.

Ниже приведен список действий, которые обычно выполняются автоматически при входе в сеанс Bash:

1. Исполняется общесистемный скрипт профиля `/etc/profile`.
2. Из `/etc/profile` проверяется наличие файла ресурсов оболочки `/etc/bashrc` и,

   при его наличии, он исполняется.
3. Выполняется пользовательский скрипт профиля в его домашнем каталоге (например,

   ~/.bash_profile).
4. В пользовательском профиле проверяется наличие в домашнем каталоге файла ресурсов оболочки `~/.bashrc` , и, при его наличии, он исполняется.

Примечание: При запуске оболочки из командной строки выполняется пункт 4 списка. В некоторых ОС из `~/.bashrc` производится запуск `/etc/bashrc`, если он еще не запускался.

Получение отчетов об активности пользователей.

Команда `who` позволяет получить список пользователей, находящихся в настоящее время в сеансе.

Информация об этом берется из специального двоичного файла `/var/run/utmp` .

Пример:

$ who

user1 :0 Jan 21 15:10

С помощью этой же команды, используя соответствующие опции, можно получать и другую информацию. Ниже приведены некоторые часто используемые опции команды `who` :

1. `-b` – время последней загрузки системы;
2. `-H` – печать заголовка;
3. `--login` – информация о системных процессах, контролирующих вход в сеанс;
4. `-q` – печатает имена всех пользователей в сеансе и их количество;
5. `-w` – текущий статус всех сеансов;
6. `-u` – подробная информация о сеансах;
7. `-a` – полная информация о статусе сеансов и процессов, контролирующих вход в сеанс.

Пример: приведенная ниже команда выведет информацию о сеансах пользователей:

$ who -uH

NAME LINE TIMEIDLE PID COMMENT user1 :0 Jan 21 15:10 ? 2014

root pts/1 Jan 21 15:21 . 2611 (localhost)

Просматривать и управлять сеансами пользователей в systemd можно с помощью команды loginctl.

Пример:

loginctl list-sessions SESSION UID USER SEAT TTY

3 1000 admuser seat0 tty1

4 0 root seat0 tty2

6 1000 admuser

3 sessions listed.

loginctl session-status 6 6 - admuser (1000)

Since: Sun 2021-02-07 21:10:01 +05; 3 days ago

Leader: 1526 (sshd)

Remote: 192.168.101.11

Service: sshd; type tty; class user State: active

Unit: session-6.scope

├─ 1526 sshd: admuser [priv]

├─ 1528 sshd: admuser@pts/0

├─ 1529 -bash

├─347639 sudo su -

├─347640 su -

├─347641 -bash

├─348247 loginctl session-status 6

└─348248 less

Feb 10 21:03:51 cent8-stream sudo[347637]: admuser : TTY=pts/0 ; PWD=/home/adm> Feb 10 21:03:51 cent8-stream sudo[347637]: pam_systemd(sudo:session): Cannot cr> Feb 10 21:03:51 cent8-stream sudo[347637]: pam_unix(sudo:session): session open> Feb 10 21:03:51 cent8-stream sudo[347637]: pam_unix(sudo:session): session clos> Feb 10 21:03:53 cent8-stream sudo[347639]: admuser : TTY=pts/0 ; PWD=/home/adm> Feb 10 21:03:53 cent8-stream sudo[347639]: pam_systemd(sudo:session): Cannot cr> Feb 10 21:03:53 cent8-stream sudo[347639]: pam_unix(sudo:session): session open>

loginctl terminate-session 6

Connection to 192.168.101.180 closed by remote host. Connection to 192.168.101.180 closed.

Для получения отчета о сеансах пользователей, которые уже завершились, необходимо воспользоваться информацией, сохраняемой в файле `/var/log/wtmp`.

Этот бинарный файл имеет ту же структуру, что и `/var/run/utmp` , поэтому его содержимое можно отобразить, указав его в качестве аргумента команды `who` . Однако, для этого предназначена специальная команда `last` .

Пример:

```
$ last
root     pts/1        localhost        Wed Jan 21 15:21   still logged in
user1    :0                            Wed Jan 21 15:10   gone - no logout
reboot   system boot  2.4.20-alt5-up   Wed Jan 21 15:09   (00:27)
postgres ???          localhost        Wed Jan 21 15:09 - 15:09 (00:00)
root     pts/1        localhost        Wed Jan 21 14:52 - down   (00:10)
```

Примечание: Эта команда выводит информацию об открытых и законченных сеансах в обратном хронологическом порядке.

В файл `/var/log/btmp` записывается информация о неудачных попытках входа. Для его просмотра используется команда `last`.

Пример:

last -f /var/log/btmp

admuser ssh:notty 192.168.101.11 Tue Feb 9 21:48 gone - no logout btmp begins Tue Feb 9 21:48:07 2021

Имеется также стандартный файл журнала `/var/log/lastlog` , в котором также в бинарном виде хранится информация о последних входах в сеанс.

Для получения информации, находящейся в этом файле, требуется использовать команду lastlog .

Модуль 6. Управление доступом к файлам.

Права владения файлами.

Каждый файл имеет два идентификатора определяющего его принадлежность – владелец файла и группа пользователей.

Эта информация сохраняется не в самом файле, а в его метаданных (inode).

Любой файл принадлежит одному единственному пользователю. Этот пользователь называется владельцем (или пользователем - user) файла.

Когда любой пользователь системы создает файл, то права владения этим файлом принадлежат именно этому пользователю.

Первичная группа пользователя (GID) в обычных условиях определяет группу пользователей, которая будет установлена для вновь создаваемого файла.

Права доступа к файлам могут быть определены с помощью команды `ls -l`.

Пример:

```
$ id
uid=501(user1) gid=100(users) группы=100(users)
$ > file
$ ls -l file
-rw-r--r--    1 user1    users  0 Дек 12 18:03 file
```

Примечание: Пример демонстрирует, что в сеансе находится пользователь `user1`, который создал файл. Из вывода команды `ls -l` заметно, что созданный файл принадлежит пользователю `user1` (третий столбец листинга) и имеет группу пользователей `users` (четвертый столбец). Первичная группа пользователя – `users` была установлена на файл в качестве группы его пользователей.

Примечание: Реально, в файловой системе сохраняется информация не об именах пользователя и группы, а `UID` и `GID` пользователя, создавшего файл.

В Linux существуют три базовых класса доступа к файлу:

1. User access (u) – права доступа владельца файла;
2. Group access (g) – права доступа группы владельцев файла;
3. Other access (o) – права доступа для всех остальных.

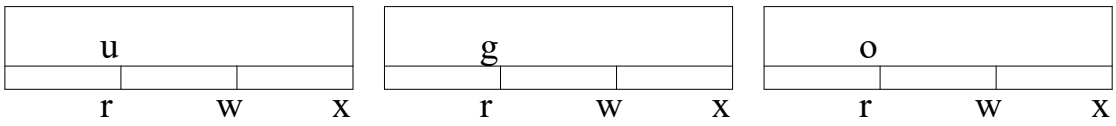
Примечание: Никаких других категорий не предусматривается, поэтому каждый пользователь может быть либо владельцем файла, либо входить в группу пользователей, либо относиться к категории всех остальных. Сами по себе права владения файлами не предоставляют информации о том, что может пользователь делать с данным файлом. Система прав доступа к файлу, описываемая ниже, определяет какие возможности работы с файлом имеет конкретный пользователь системы.

Права доступа, устанавливаемые на файлы.

Права доступа к файлу хранятся в метаданных файла и кодируются тремя триадами бит.

Права доступа можно задавать в символической и восьмеричной нотациях.

Символическая нотация основана на буквенных обозначениях прав владения и прав доступа, а восьмеричная связана с фактическим представлением этих прав в виде триад бит.



Порядок триад:

1. старшая триада соответствует правам доступа владельца файла (u);
2. средняя триада - правам доступа группы владельцев (g);
3. младшая триада – правам доступа всех остальных пользователей (o).

Порядок битов в триадах:

1. установленный в 1 старший бит в каждой триаде (4 в восьмеричной нотации) обозначает разрешение на чтение данного файла и в символической нотации обозначается r--;
2. установленный в 1 средний бит в каждой триаде (2 в восьмеричной нотации) обозначает разрешение на изменение данного файла: -w-;
3. установленный в 1 младший бит в каждой триаде (1 в восьмеричной нотации) обозначает разрешение на исполнение данного файла: --x.

Пример: запись `rxwxr-x--x` (`751`) обозначает, что пользователь файла имеет все права на доступ к нему (`rxw` или `7`), группа пользователей имеет права на чтение и исполнение файла (`r-x` или `5`), все остальные имеют права на исполнение файла

(`--x` или `1`).

Символьная и восьмеричная нотация записи прав доступа абсолютно эквивалентны.

Восьмеричное значение триады бит получается сложением степеней двойки, соответствующих номеру бита в триаде.

Пример: права доступа в символьной нотации `rwxr-xr--` в восьмеричной нотации записываются как `754`, где $7 = 2^2+2^1+2^0$, $5 = 2^2+2^0$, $4 = 2^2$.

Для того чтобы увидеть права доступа к файлу, достаточно набрать команду

ls –l, при этом права доступа к файлам выводятся в первой колонке.

Права доступа к каталогам.

Права доступа, устанавливаемые на каталоги имеют несколько иной смысл, чем права на файлы.

Для каталогов используются следующие права:

1. x – (search) – право обращаться к метаданным файлов в каталоге, что предоставляет возможность использовать имя этого каталога в имени пути до нужного файла.
2. r – право на чтение имен файлов, находящихся в каталоге, то есть на выполнение команды ls.
3. w – право на запись в каталог, то есть право переименовывать, удалять, создавать файлы и прочее.

Без наличия права на search (x), установленного на каталог, работа с находящимися внутри файлами невозможна.

Примечание: Поэтому, для каталогов права доступа должны быть либо нечетные, либо они должны отсутствовать.

Ниже приведены права доступа к каталогам, которые имеют практический смысл:

0 (---) – прав нет.

1 (--x) – имеется право перехода в каталог, можно обращаться к файлам в нем, однако, нельзя выполнять команду ls и осуществлять какие-либо манипуляции с файлами, например, переименование или удаление.

3 (-wx) - в каталог можно переходить, разрешены любые манипуляции с файлами и можно к ним обращаться, однако получить список файлов командой ls невозможно.

5 (r-x) - в каталог можно переходить и получать подробную информацию о файлах командой ls -l, разрешено обращаться к файлам, однако, нельзя осуществлять какие- либо манипуляции с файлами, например, переименование или удаление.

7 (rwx) – полные права.

Для получения прав доступа к каталогу следует использовать команду `ls -ld`.

Пример:

```
$ ls -ld
drwxr-x--x  10 user1      users  4096 Дек 12 18:03 .
```

Примечание: В примере, приведенном выше, на текущий каталог установлены права `751`, то есть владелец этого каталога (`user1`) имеет все права на этот каталог, группа (`users`) не имеет прав переименовывать и удалять файлы, поскольку не имеет прав на запись, а для всех остальных каталог является "темным". Все остальные могут переходить в этот каталог и могут обращаться к файлам, находящихся в нем, однако, при этом они должны знать, какие имена имеют файлы, к которым им необходим доступ. Это связано с тем, что командой `ls` остальные пользоваться не могут, так как прав на чтение каталога нет. Производить какие-либо манипуляции с файлами они также не имеют права, так как права на запись в каталог нет.

Изменение прав владения файлами.

Права владения файлами могут быть изменены с помощью следующих команд:

1. `chown` - эта команда позволяет менять как владельца файла или каталога, так и группу пользователей файла;
2. `chgrp` - позволяет менять группу пользователей файла.

В Linux этим команды обычно может выполнять только суперпользователь, так как передача прав владения, разрешенная для обычных пользователей, представляет собой существенную угрозу безопасности.

Примечание: При необходимости разрешить какому-либо уполномоченному пользователю исполнять эти команды, на них необходимо установить специальный бит (например, SUID - бит подмены владельца процесса), о которых будет рассказано в конце этой главы. Однако, даже не смотря на возможность для обычного пользователя с помощью такой манипуляции изменять права владения файлами, обычный пользователь может менять владельца или группу только у тех файлов, которыми он владеет.

Пример: Приведенная ниже команда меняет владельца файла:

```
# ls -l f1
-rw-r--r--      1 tania    prof        8 Окт 22 21:04 f1
# chown user1 f1
# ls -l f1
-rw-r--r--      1 user1    prof        8 Окт 22 21:04 f1
```

Примечание: Из приведенного выше примера заметно, что первый аргумент команды - имя нового владельца файла, а далее идут файлы или каталоги, права на владение которыми изменяются.

Пример: Ниже приведен пример смены группы пользователей файлов `f1` и `text.c` :

```
# ls -l f1 text.c
-rw-r--r--    1 user1      prof   8 Окт 22 21:04 f1
-rw-r--r--    1 profprof   175 Дек 13 21:24 text.c # chgrp tania f1 text.c
# ls -l f1 text.c
-rw-r--r--    1 user1      tania  8 Окт 22 21:04 f1
-rw-r--r--    1 proftania  175 Дек 13 21:24 text.c
```

С помощью команды `chown` можно одновременно изменить владельца и группу (через двоеточие или точку) одновременно, причем новый владелец вовсе не обязан быть членом той группы, которая будет установлена на файл

Пример:

chown tania:sys f1 # ls -l f1

-rw-r--r-- 1 tania sys 8 Окт 22 21:04 f1

Примечание: В этом примере продемонстрировано, как одновременно поменять владельца и группу с помощью команды `chown` . В команде новый владелец и новая группа указываются через знак двоеточия в соответствии с POSIX. Однако, в Linux допускается использование BSD стиля, в котором вместо двоеточия указывают точку. Например, приведенная ниже команда с помощью `chown` изменит только группу в BSD стиле (точка вместо двоеточия):

Пример:

chown .adm f1 # ls -l f1

-rw-r--r-- 1 tania adm 8 Окт 22 21:04 f1

Опция `-c` GNU версий команд `chown` и `chgrp` позволяет получать подробную информацию об изменяемых правах владения

Пример:

chgrp -c tania f1

изменена группа `f1' на tania

Обе команды `chown` и `chgrp` имеют опцию `-R`, позволяющую рекурсивно изменять права владения на каталоги и их содержимое.

Пример:

ls -Rl scores/ scores/:

итого 1

drwxrwxr-x 2 profprof 80 Авг 24 16:20 rnd_tutorial

scores/rnd_tutorial:

итого 4

-rw-rw-r-- 1 profprof 1040 Авг 24 16:20 000.score # chown -R tania.tania scores/

ls -Rl scores/ scores/:

итого 1

drwxrwxr-x 2 tania tania 80 Авг 24 16:20 rnd_tutorial

scores/rnd_tutorial:

итого 4

-rw-rw-r-- 1 tania tania 1040 Авг 24 16:20 000.score

Примечание: В этом примере владельцем и группой пользователей стали `tania` (владелец и группа). В команде был применен BSD стиль указания владельца и группы (точка вместо двоеточия).

При использовании рекурсивной смены прав владения бывает очень удобно получать подробную информацию об этом процессе. Для этого предназначена опция `-v` команд `chown` и `chgrp`

Пример:

chgrp -Rv users scores/ изменена группа `scores/' на users

изменена группа `scores//rnd_tutorial' на users

изменена группа `scores//rnd_tutorial/000.score' на users

Примечание: Пример демонстрирует, что с опцией `-v` для каждого файла, на который изменяются права владения, выдается подробная информация.

Примечание: Внимание! *Неосторожное использование команд `chown` и `chgrp`, особенно с опцией `-R`, может привести к выводу всей системы из строя!*

Установка прав доступа.

Команда `chmod` предназначена для изменения прав доступа к файлам и каталогам, указанным в качестве аргументов.

Права доступа должны быть указаны либо в восьмеричной, либо в символьной нотации.

Права указывают в качестве первого аргумента команды.

Изменять права доступа к файлу могут только суперпользователь и владелец файла.

Пример: Ниже приведен пример использования команды `chmod` для изменения прав доступа к файлу в восьмеричной нотации:

$ ls -l text.c

```
-rw-r--r--    1 prof      tania         175 Дек 13 21:24 text.c
$ chmod 660 text.c
$ ls -l text.c
-rw-rw----    1 prof      tania         175 Дек 13 21:24 text.c
```

Примечание: Использование команды `chmod 660 text.c` позволило установить права `660` на файл text.c .

Команда `chmod` позволяет также устанавливать права на доступ к файлу, указывая их в символической нотации. Для этого применяется следующая форма команды

chmod класс_изменение_права файлы

класс может принимать следующие значения:

1. u – доступ владельца;
2. g – доступ группы владельцев;
3. o – доступ всех остальных;
4. a – доступ всех групп пользователей.

изменение может принимать следующие значения:

1. + - разрешить;
2. - - запретить;
3. = - установить.

Права может принимать следующие значения:

5. r – чтение;
6. w – запись;
7. x – выполнение.

Если используются операции разрешения (+) или запрета (-) прав на файл, то они не изменяют те биты прав доступа, которые не относятся к требуемому изменению.

Пример: если для файла `text.c` требуется удалить право на изменение для группы, и добавить право на чтение для всех трех категорий пользователей (владелец, группа, все остальные), то выполним следующую команду:

```
$ chmod g-w,a+r text.c
$ ls -l text.c
-rw-r--r--    1 proftania  175 Дек 13 21:24 text.c
```

Примечание: приведенная выше команда отобрала права на запись для группы пользователей файла `f1`, но это не отразилось на правах на чтение этого файла, установленных для группы.

Использование операции назначения (=) стирает те права, которые были установлены ранее и назначает новые.

Пример: установим на файл `text.c` права на чтение и запись для владельца и группы, и запретим всем остальным какой-либо доступ к файлу:

```
$ chmod ug=rw,o= text.c
$ ls -l text.c
-rw-rw----   1 proftania  175 Дек 13 21:24 text.c
```

Аналогично командам `chown` и `chgrp` , команда `chmod` способна рекурсивно изменять права доступа к каталогам и всему их содержимому, если она вызвана с опцией `-R`.

Примечание: этой возможностью следует пользоваться с особой осторожностью, принимая во внимание концептуальные отличия прав на файлы от прав на каталоги - на файлы в большинстве случаев устанавливаются четные права (отсутствие прав на исполнение), а на каталоги наоборот - нечетные (без права на search каталоги не позволят обращаться к метаданным файлов внутри них).

Пример: Приведенная ниже команда снимает права на запись для каталога `scores` :

```
$ ls -lR scores scores:
итого 1
drwxrwxr-x 2 profusers  80 Авг 24 16:20 rnd_tutorial

scores/rnd_tutorial:
итого 4
-rw-rw-r--   1 profusers  1040 Авг 24 16:20 000.score

$ chmod -R g-w scores

$ ls -lR scores scores:
итого 1
drwxr-xr-x  2 profusers  80 Авг 24 16:20 rnd_tutorial

scores/rnd_tutorial:
итого 4
-rw-r--r--     1 profusers  1040 Авг 24 16:20 000.score
```

Примечание: Заметно, что и с самого каталога, и с всего его содержимого были удалены права на запись для группы пользователей.

Примечание: В обычной практике права на каталоги и на файлы устанавливаются отдельно. При необходимости рекурсивного изменения прав на каталог и его содержимое опцию `-R` команды `chmod` обычно не используют. Вместо этого пользуются командой `find` с установкой `-exec chmod` (или `xargs chmod`).

Пример: в каталоге `scores` требуется установить для файлов права `644`, не затрагивая при этом права на каталоги:

```
$ find scores -type f -exec chmod 644 {} \;
$ ls -lR scores scores:
итого 1
drwxr-xr-x  2 profusers  80 Авг 24 16:20 rnd_tutorial
scores/rnd_tutorial:
итого 4
-rw-r--r--    1 profusers  1040 Авг 24 16:20 000.score
```

GNU версия команды `chmod` позволяет использовать опцию `-v` для получения информации о файлах, права доступа к которым изменяются, и опцию `-c` для получения подробностей изменения прав.

Пример:

```
$ chmod -Rv g-w scores
права доступа `scores' изменены на 0755 (rwxr-xr-x)
права доступа `scores/rnd_tutorial' изменены на 0755 (rwxr-xr-x)
права доступа `scores/rnd_tutorial/000.score' изменены на 0644 (rw-r--r--)
$ chmod -c g-w text.c
права доступа `text.c' изменены на 0640 (rw-r    )
$ chmod -v 660 text.c
права доступа `text.c' изменены на 0660 (rw-rw )
```

Примечание: опция `-v` выдает подробную информацию всегда, а `-c` только тогда, когда права действительно изменяются.

Автоматическая установка прав доступа к вновь создаваемым файлам.

Команда `umask` предназначена для автоматической установки прав доступа

к вновь создаваемым файлам и каталогам.

Команда `umask` позволяет задавать значение битовой маски, которая будет “вычитаться” из прав 777 для каталогов и 666 для файлов.

При вызове этой команды без аргумента она возвратит текущее значение маски

Пример:

$ umask 0022

Установка другого значения `umask` никоим образом не отразится на уже существующих файлах и каталогах, она участвует только в процессе определения прав на вновь создаваемые файлы и каталоги.

Пример:

$ umask 002

$ mkdir dir1

$ > file1

$ ls -ld dir1 file1

drwxrwxr-x 2 prof prof 48 Дек 14 20:43 dir1

-rw-rw-r-- 1 prof prof 0 Дек 14 20:43 file1

$ umask 077

$ mkdir dir2

$ > file2

$ ls -ld dir2 file2 prof 48 Дек 14 20:44 dir2

drwx------ 2 prof

-rw------- 1 prof prof 0 Дек 14 20:44 file2

Примечание: В этом примере продемонстрировано, что при установленном значении `umask` равном `002` на каталоги устанавливаются права `775`, а на файлы - `664`. В то же время `umask` , установленная в `077` дает в результате, соответственно, `700` - для каталогов и `600` - для файлов.

Ниже приведена таблица наиболее часто применяемых значений `umask`.

umask	Каталоги	Файлы
002	775	664

007	770	660
022	755	644
027	750	640
077	700	600

Значение `umask` можно задавать также и в символьной нотации

Пример:

$ umask g=rwx,g=rx,o=

$ umask 0027

Примечание: При задании значения `umask` в символьной нотации всего лишь требуется указать в качестве аргумента права, которые должны будут иметь новые каталоги.

Специальные биты прав доступа: SUID, SGID и sticky bit.

Помимо битов, устанавливающих разрешения на доступ к файлу, существуют специальные атрибуты, которых образуют еще одна триаду битов:

1. Sticky bit (Save text mode) - бит “липучка”;
2. SUID (Set User ID) – бит подмены UID;
3. SGID (Set Group ID) – бит подмены GID.

Sticky bit кодируется восьмеричной 1 (двоичная 001), SGID кодируется восьмеричной 2 (010), а SUID - 4 (100).

В символьной нотации применяются символы T для Sticky bit, S для SUID и SGID. Эти символы всегда выводятся в позиции, где должен находиться флаг разрешения на исполнение (x).

Если одновременно установлены и бит x и бит S, то отображается символ s

Биты T и x всегда устанавливаются вместе, поскольку бит T используется только для каталогов. Поэтому в символьном отображении прав доступа встречается только символ t

SUID отображается в виде буквы s или S в старшей триаде бит, отображающей права владельца

SGID отображается в виде буквы s или S в средней триаде бит, отображающей права группы

Sticky bit отображается в виде буквы t в младшей триаде бит, отображающей права для всех остальных.

Пример:

$ ls -ld /tmp /bin/ping

```
-rws--x--x   1 root      root        32908 Окт 16   2002 /bin/ping
drwxrwxrwt  94 root      root         3488 Дек 14 20:31 /tmp
```

Примечание: Приведенный выше пример демонстрирует, что на файл системной команды `ping` установлен бит SUID (символ `s` в старшей триаде вместо прав на исполнение), а на каталог `/tmp` установлен Sticky bit (символ `t` в триаде бит для прав всех остальных).

Примечание: Атрибут Sticky bit для файлов не используется, в ранних версиях UNIX он был предназначен для того, чтобы оставить в памяти образ программы (Save text mode).

Процесс наследует права доступа к системным ресурсам от пользователя (UID), запустившего процесс, и его первичной группы (GID), если для исполняемых файлов, не установлены биты SUID и/или SGID

При установленном на исполняемый файл бите SUID процесс исполняется не от имени пользователя, запустившего его, а от имени владельца исполняемого файла команды.

При установленном бите SGID процесс исполняется не от имени первичной группы пользователя, запустившего процесс, а от имени группы пользователей файла.

У каждого процесса имеется четыре идентификатора:

1. RUID - Real UID, который всегда равен UID пользователя, выполнившего команду.
2. RGID - Real GID, который всегда равен GID пользователя, выполнившего команду.
3. EUID - Effective UID, который либо равен RUID, либо если на исполняемый файл установлен бит SUID, то UID владельца файла.
4. EGID - Effective GID, который либо равен RGID, либо если на исполняемый файл установлен бит SGID, то GID владельца файла.

Примечание: В подавляющем большинстве случаев подмена владельца или группы осуществляется на root или какого-либо высокопривелигированного пользователя или группу. Например, при выполнении команды `ping` (см. пример выше), несмотря на то, что ее запустил обычный пользователь, она будет исполняться от имени `root`, так как он владеет ее исполняемым файлом. Это используется, например, в таких программах, как `passwd`, которые требуют временного предоставления доступа обычному пользователю к тем ресурсам, к которым он не имеет доступа. Естественно, такие программы требуют особого подхода к разработке, так как предоставляют серьезную угрозу для безопасности системы. На файлы скриптов Shell биты SUID и SGID устанавливать можно, но они действовать не будут.

Установка Sticky bit на каталог, в отношении которого пользователь имеет права на чтение и на запись, позволяет запретить удалять и изменять пользователю чужие файлы в этом каталоге.

Примечание: Это используется при установке прав доступа к каталогу

/tmp, открытому на запись всем, поскольку иначе пользователь может удалить чужие временные файлы, находящиеся в этом каталоге, что может повлечь плачевные последствия.

При установке атрибута SGID на каталог, вновь созданные файлы в этом каталоге будут наследовать группу владельцев по группе владельцев каталога (так называемый “стиль BSD”), вместо RGID процесса, создающего файл по версии System V.

Пример:

```
$ cd dir1

$ ls -ld
drwxrwsr-x  2 tania         users  48 Дек 14 20:43 .

$ id
uid=500(prof) gid=500(prof) группы=500(prof),100(users)

$ > file

$ ls -l итого 0
-rw-r-----      1 profusers  0 Дек 14 22:00 file
```

Примечание: В каталоге, на который установлен бит SGID, был создан файл. При этом группа владельцев файла была назначена не по первичной группе пользователя, создавшего файл, а по группе пользователей каталога, в котором файл был создан.

Команда `chmod` позволяет установить особые биты доступа на файлы и каталоги. 16.Для установки специальных битов в символьном режиме команда `chmod` должна быть выполнена со следующими аргументами:

1. u+s - для установки на файл бита SUID;
2. g+s - для установки на файл или каталог бита SGID;
3. o+t - для установки на каталог бита Sticky bit.

Для установки специальных битов в числовом режиме команде `chmod` в качестве первого аргумента передается число, состоящее из 4 цифр, где левая цифра представляет собой сумму специальных битов.

Пример: Приведенная ниже команда `chmod 2775 d1` устанавливает бит SGID на каталог:

```
$ mkdir d1
$ ls -ld d1
drwxr-x---  2 prof  prof  48 Дек 14 22:31 d1
$ chmodd1
2770
$ ls -ld d1
drwxrws---  2 prof  prof  48 Дек 14 22:31 d1
```

Ниже приведена таблица, в которой указаны результаты установки различных специальных прав доступа на файлы и каталоги.

Права	Эффект для каталогов	Эффект для файлов
-rws--x--x		Комада исполняется от имени владельца файла.
-rwx--s--x		Комада исполняется от имени группы пользователей файла.
drwxrws---	На файлы, создаваемые в каталоге, будет установлена такая же группа, как у каталога.	
drwxrwxrwt	В каталоге можно удалять или переименовывать только собственные файлы.	

Модуль 7. Управление пакетами ПО в РЕД ОС

В чем состоит управление программным обеспечением.

Одной из основных задач системного администрирования, возникающей сразу после установки системы и актуальной в течении всей ее эксплуатации, является управление программным обеспечением. Этот процесс имеет следующие составляющие:

Установка нового программного обеспечения. В течении эксплуатации системы неизбежно будет возникать необходимость установки нового программного обеспечения. Во-первых, по причине того, что маловероятно установить систему так, чтобы она вечно удовлетворяла постоянно изменяющимся требованиям эксплуатации. Во-вторых, в течении длительной эксплуатации системы могут проявиться концептуальные проблемы (например, с безопасностью) для каких-либо программных компонент. Для устранения этих проблем может потребоваться отказаться от данной программы вообще и перейти к другой аналогичной.

Обновление программного обеспечения. В коде программ постоянно обнаруживаются более или менее серьезные проблемы. Поэтому системный администратор должен отслеживать сообщения об ошибках и проблемах в программном обеспечении и обновлять его. Одна из наиболее распространенных причин взлома систем безопасности – использование устаревшего программного обеспечения, имеющего известные проблемы с безопасностью.

Сопутствующая установке и обновлению программного обеспечения проблема состоит в **проверке подлинности** нового программного обеспечения. Если новые программы или пакеты обновления поступают из ненадежных источников, то при этом резко увеличивается вероятность попадания в систему программ-закладок (троянских коней), выполняющих в системе несанкционированные действия.

Удаление программного обеспечения. Необходимость удаления продиктована требованием наличия в системе только того программного обеспечения, которое действительно необходимо. В большинстве современных дистрибутивов GNU/Linux при установке системы на диск копируется заранее предопределенный поставщиком дистрибутива набор программ. Многие из них в работе не используются и просто занимают место на диске. Еще хуже, если в системе работает программа, открывающая сокеты для ненужных сетевых соединений. Такие программы становятся удобной мишенью для взломщиков. Также неиспользуемое программное обеспечение может требовать наличия в системе неиспользуемых учетных записей пользователей, которые также потенциально могут быть использованы взломщиками. Еще одна причина необходимости удаления программ заключается в возможных конфликтах нужных

в системе программ с уже установленным программным обеспечением.

Проверка целостности программного обеспечения. Эта задача связана с защитой от возможной порчи программного обеспечения при сбоях в системе или в результате чьей-либо несанкционированной деятельности в системе. Проверка целостности заключается в проверке размеров файлов, прав доступа и владения, контрольных сумм, времени модификации и прочее.

Создание собственных пакетов или **пересборка** существующих. Сборка новых пакетов больше связана с деятельностью разработчиков программного обеспечения или создателей пакетов, отвечающих за их поддержку (maintainers). Однако пересборка существующего пакета требуется довольно часто и в работе обычного системного администратора. Это может быть связано, например, с желанием получить оптимизированное программное обеспечение для конкретной аппаратной платформы.

Менее распространенная задача в мире свободного программного обеспечения – **регистрация** и **лицензирование** программного обеспечения. В последнее время GNU/Linux часто используется для работы проприетарного (коммерческого несвободного) программного обеспечения, например, в качестве платформы для СУБД Oracle. Проприетарное программное обеспечение должно быть зарегистрировано и лицензировано. Имеется также обилие дистрибутивов GNU/Linux, таких, как Red Hat Enterprise Linux и SUSE Linux, требующих регистрации и лицензирования. Другой пример – большинство антивирусных пакетов для GNU/Linux являются проприетарными и требуют регистрации и лицензирования.

Существует несколько вариантов установки программного обеспечения:

Сборка и установка из архивов с исходным кодом (**tarballs**).

Установка из архивов с бинарным машинным кодом (**binaries**).

Установка из бинарных пакетов (**package**) с помощью систем управления пакетами (**package manager**).

Сборка бинарного пакета из пакета с исходным кодом (**source package**) с последующей установкой из получившегося бинарного пакета с помощью систем управления пакетами (**package manager**).

Сборка и установка программного обеспечения из архивов с исходным кодом под управлением специального скрипта автоматизации – порта (основной способ установки пакетов в Gentoo, где порты называются portage по аналогии с port во FreeBSD).

В подавляющем большинстве GNU/Linux дистрибутивов имеется система управления пакетами.

Такая система в значительной мере упрощает и стандартизирует процесс

управления программным обеспечением.

Основываясь на информации, предоставляемой на сайте www.distrowatch.org можно утверждать, что наиболее распространены четыре системы управления пакетами:

RPM – Red Hat Package Manager. Применяется в RH и подобных ему системах, SUSE и многих других дистрибутивах. Предоставляет возможности как установки бинарных пакетов (файлы .rpm), так и с помощью предварительной пересборки из пакетов с исходным кодом (файлы .src.rpm или, реже, .srpm).

Система управления пакетами Debian. Кроме Debian используется в собранных на его основе дистрибутивах, например в Ubuntu. Предоставляет широкие возможности по управлению пакетами. Файлы пакетов имеют расширению .deb.

Система портов Gentoo. Этот дистрибутив ориентирован на сборку высоко оптимизированного программного обеспечения с помощью специальных скриптов для сборки программного обеспечения из архивов с исходным кодом. Позволяет также устанавливать заранее собранные пакеты (фактически, пакет – это заранее скомпилированное ПО из порта). Преимуществом этой системы является непревзойденные возможности постоянного обновления программного обеспечения.

Система управления пакетами, принятая в SlackWare и ему подобных дистрибутивах. Здесь применяется установка программного обеспечения из заранее собранных бинарных архивов в формате tar.

Первые две системы из приведенного выше списка имеют превалирующее распространение. В операционной системе РЕД ОС используется система RPM

Система управления программным обеспечением предоставляет следующие преимущества:

Обеспечивается единообразное управление программным обеспечением.

Программы устанавливаются в стандартные места файловой системы.

Управление программным обеспечением значительно облегчено.

Во многих системах предоставляются возможности разграничения ролей пользователей, могущих выполнять те или иные функции в управлении программным обеспечением.

Легко обеспечивается возможность проверки целостности программного обеспечения.

Однако имеются и недостатки, связанные с использованием систем управления пакетами, ориентированных на бинарные пакеты:

Часто бывает затруднительно установить не все программное обеспечение из

пакета, а только его часть.

Трудно устанавливать программы в нестандартные места файловой системы, например, в домашние каталоги пользователей.

Трудно, а иногда и невозможно устанавливать программы из других дистрибутивов или из предыдущей версии этого же дистрибутива.

Пакеты необходимо пересобирать для оптимизации под данную систему, а также для добавления или удаления некоторой функциональности.

В соответствии со стандартом FHS программное обеспечение, устанавливаемое с помощью систем управления пакетами, размещается в каталогах:

/bin

/sbin

/lib

/usr/bin

/usr/sbin

/usr/lib

/usr/X11R6

/opt

Примечание: Каталог `/opt` обычно используется для программного обеспечения, не поставляемого в составе дистрибутива (опциональное ПО). В каталог `/usr/X11R6` помещаются файлы, относящиеся к системе X Window.

Файлы помощи для программного обеспечения, устанавливаемого с помощью систем управления пакетами, должны размещаться в `/usr/share/man`, а документация – в /usr/share/doc.

Программное обеспечение, устанавливаемое самостоятельно с помощью сборки из архивов с исходным кодом, размещается в подкаталогах каталога `/usr/local`.

При установке программного обеспечения очень часто возникает конфликт пакетов или зависимостей, который может иметь следующие причины:

Два пакета взаимно исключают совместную работу.

Пример: нельзя использовать два сервера SMTP (Simple Mail Transfer Protocol). При попытке установить программу `postfix` в системе, где установлена почтовая программа `sendmail`, возникнет конфликт и менеджер пакетов выведет сообщение об ошибке.

Основная библиотека `libc` (`glibc`), используемая в системе, имеет версию, отличную от версии библиотеки, с которой собран устанавливаемый пакет. Этот вариант конфликта может быть также и для других библиотек. Однако, конфликт с `libc` разрешить можно, фактически только перейдя на другой дистрибутив GNU/Linux.

Устанавливаемый пакет может требовать наличие других программ или библиотек, отсутствующих в настоящий момент в системе.

Может возникать также и конфликт версий конфигурационных файлов. Однако такой вид конфликта разрешается довольно легко – достаточно переименовать старые версии (или же просто одноименные конфигурационные файлы) файлов конфигурации, оставшиеся, например, от предыдущей версии программы.

Менеджер пакетов RPM.

В Red Hat Linux и подобных ему дистрибутивах используется менеджер пакетов RPM.

Менеджер пакетов RPM реализован с помощью группы программ, главная из которых

/bin/rpm.

Ниже приведен список важнейших режимов работы RPM:

1. *Запрос*. Включается опцией `-q` или `--query`.
2. *Проверка целостности файлов*, установленных из пакета. Включается опцией `-V` или

   --verify.
3. *Проверка электронной подписи пакета*. Включается при использовании опции `-K` или

   --checksig.
4. *Установка пакета*. Требует использования опции `-i` или `--install`.
5. *Обновление пакета*. Работает с опцией `-U` или `--upgrade`.
6. *Обновление версии пакета*. Запускается при установленной опции `-F` или

   --freshen.
7. *Удаление установленных пакетов*. Этот режим включается при использовании опции `- e` или `--erase`.

Используя режим запроса (опция `-q`) можно, например, узнать точную версию установленного в системе пакета:

Пример:

$ rpm -q bash

bash-5.0.17-2.fc33.x86_64

Примечание: В этом примере был осуществлен запрос к базе данных RPM. В качестве ключа запроса использовалось краткое название установленного в системе пакета - `bash`. В результате исполнения запроса было получено полное имя пакета.

Для получения более подробной информации о пакете следует воспользоваться опциями -qi :

Пример:

$ rpm -qi bash

Name : bash

Version : 5.0.17

Release : 2.fc33 Architecture: x86_64

Install Date: Ср 28 окт 2020 19:37:44 Group : Unspecified

Size : 7709818

License : GPLv3+

Signature : RSA/SHA256, Вт 28 июл 2020 03:10:09, Key ID 49fd77499570ff31 Source RPM : bash-5.0.17-2.fc33.src.rpm

Build Date : Пн 27 июл 2020 18:17:35

Build Host : buildhw-x86-14.iad2.fedoraproject.org Packager : Fedora Project

Vendor : Fedora Project

URL : https://www.gnu.org/software/bash Bug URL : https://bugz.fedoraproject.org/bash Summary : The GNU Bourne Again shell Description :

The GNU Bourne Again shell (Bash) is a shell or command language interpreter that is compatible with the Bourne shell (sh). Bash incorporates useful features from the Korn shell (ksh) and the C shell (csh). Most sh scripts can be run by bash without modification.

Примечание: Использование опций `-qi` выводит подробную информацию о пакете, в том числе суммарный размер файлов, установленных из пакета (поле `Size`), и группу пакетов, к которой этот пакет принадлежит (поле `Group`).

Для получения списка файлов, установленных из пакета, необходимо установить опции -ql :

Пример:

```
$ rpm -ql bash
/etc/skel/.bash_logout
/etc/skel/.bash_profile
/etc/skel/.bashrc
/usr/bin/alias
/usr/bin/bash
/usr/bin/bashbug
/usr/bin/bashbug-64
/usr/bin/bg
...
```

Можно также ограничиться лишь выводом списка конфигурационных файлов, установленных из заданного в качестве аргумента пакета, используя для этого опции -qc :

Пример:

```
$ rpm -qc bash
/etc/skel/.bash_logout
/etc/skel/.bash_profile
/etc/skel/.bashrc
```

С помощью опций `-qd` можно получить информацию о файлах помощи и документации, установленных из этого пакета:

Пример:

```
$ rpm -qd bash
/usr/share/doc/bash/FAQ
/usr/share/doc/bash/INTRO
/usr/share/doc/bash/RBASH
/usr/share/doc/bash/README
/usr/share/doc/bash/bash.html
/usr/share/doc/bash/bashref.html
/usr/share/info/bash.info.gz
/usr/share/man/man1/..1.gz
```

/usr/share/man/man1/:.1.gz

/usr/share/man/man1/[.1.gz

/usr/share/man/man1/alias.1.gz

...

Сочетание опций `-qa`, позволяет получить список из всех пакетов, установленных в системе. Этим удобно пользоваться, если название пакета не известно.

Пример: можно получить информацию о всех пакетах, в имени которых имеется строка

ftp:

$ rpm -qa | grep ftp ftplib-4.0-13.fc33.x86_64 ncftp-3.2.5-21.fc33.x86_64 ftp-0.17-84.fc33.x86_64

tftp-server-5.2-31.fc33.x86_64 lftp-4.9.2-1.fc33.x86_64

python3-requests-ftp-0.3.1-20.fc33.noarch tftp-5.2-31.fc33.x86_64

Сочетание опций `-qf` позволяет в качестве ключа запроса указать имя файла и узнать из какого пакета установлен этот файл:

Пример:

$ rpm -qf /usr/bin/passwd passwd-0.80-9.fc33.x86_64

Для получения информации о еще не установленном в системе пакете можно использовать опции -qp, позволяющие извлекать информацию из файла пакета (из

.rpm файла):

Пример:

$ rpm -qip Загрузки/zoom_x86_64.rpm

warning: Загрузки/zoom_x86_64.rpm: Header V4 RSA/SHA1 Signature, key ID 61a7c71d: NOKEY

Name : zoom

Version : 3.5.374815.0324

Release : 1

Architecture: x86_64

Install Date: (not installed) Group : default

Size : 269036822

License : see https://www.zoom.us/

Signature : RSA/SHA1, Wed 25 Mar 2020 10:01:53 AM +05, Key ID b903bf1861a7c71d Source RPM : zoom-3.5.374815.0324-1.src.rpm

Build Date : Wed 25 Mar 2020 09:58:57 AM +05 Build Host : localhost

Relocations : /

Packager : Zoom Linux Team <linux-dev@zoom.us> Vendor : Zoom Video Communications, Inc.

URL : https://www.zoom.us

Summary : Zoom, #1 Video Conferencing and Web Conferencing Service Description :

Zoom, #1 Video Conferencing and Web Conferencing Service

Zoom, the cloud meeting company, unifies cloud video conferencing, simple online meetings, and group messaging into one easy-to-use platform. Our solution offers the best video, audio, and screen-sharing experience across Zoom Rooms, Windows, Mac, Linux, iOS, Android, and H.323/SIP room systems.

Опция `-V` переключает `rpm` в режим проверки целостности установленных файлов из пакетов.

Пример: проверки целостность пакета `mozilla` :

$ rpm -V bash

S.5....T. c /etc/skel/.bashrc

Примечание: Выводимая в первом столбце информация означает следующее: `S` - у файла не совпадает размер с указанным в базе данных, `5` - не совпадает сигнатура `md5`, `T` - время модификации изменено.

Установка или обновление пакета осуществляется с помощью опций:

`-i` - пакет будет установлен в случае отсутствия в системе его предыдущей версии;

`-U` - если в системе существовала предыдущая версия пакета, она будет заменена на устанавливаемую. В противном случае пакет будет просто установлен;

`-F` - возможно только обновление версии пакета. Если предыдущей версии в системе не обнаружено, то установка произведена не будет.

При установке или обновлении пакетов в качестве аргумента для `rpm` указывают имена файлов пакетов (`.rpm` файлы).

Имеется возможность проверить последствия установки без проведения реальной установки пакетов. Для этого предназначена опция `--test` :

Эта же опция полезна для проверки последствий удаления пакетов из системы без реального удаления.

Пример: можно узнать результаты удаления пакета `postfix`:

$ rpm -e --test postfix

ошибка: удаление этих пакетов нарушит зависимости: MTA нужен для fetchmail-6.2.1-alt1

MTA нужен для mutt-1.4i-alt3

MTA нужен для gnome2-media-cddbslave-2.2.1-alt1 MTA нужен для gnome2-media-gcdplayer-2.2.1-alt1

MTA нужен для pilot-mailsync-0.7.1-alt1

Примечание: Информация, выведенная этой командой, сообщает о зависимости нескольких пакетов в системе от пакета `postfix`.

Бывают, все же, случаи, когда установка или удаление пакета должно быть произведено не взирая на нарушение зависимостей. Для этого предназначена опция `--nodeps`.

Как правило команду `rpm` не применяют напрямую в RH подобных системах. Вместо этого используются специальные системы упрощающие получение, обновление, поиск, установку, удаление и т. д. пакетов.

В таких системах не рекомендуется манипулировать пакетами напрямую командой `rpm`. Среди основных это менеджеры пакетов `yum` (устарел) и `dnf`.

Данные утилиты используют собственные базы данных для работы с пакетами, именно по этой причине не рекомендуется использовать команду `rpm` для прямой установки, удаления или обновления пакетов.

Данный менеджеры используют понятие репозитория — некого хранилища с пакетами. В каждом репозитории помимо самих пакетов имеются специальные базы данных с информацией о пакетах.

Репозитории могут быть как сетевыми так и локальными.

Добавление репозитория обычно осуществляется установкой соответствующего пакета. Описание репозиториев находится в каталоге `/etc/yum.repos.d`

В файле, который описывает репозиторий, например, `/etc/yum.repos.d/CentOS- Base.repo`, необходимо убедиться, что репозиторий включен — опция `enabled=1`

Основной файл конфигурации для `yum` `/etc/yum.conf`. Для `dnf`

/etc/dnf/dnf.conf.

Основные команды и опции `yum` и `dnf` одинаковые:

1. `list` — список пакетов в репозиториях
2. `install` — установка пакетов, которые указаны в команде
3. `remove` — удаление пакетов
4. `update` — обновление пакетов
5. `disto-sync` — обновление системы
6. `grouplist` — список групп пакетов
7. `groupinstall` — установка группы
8. `groupremove` — удаление группы
9. `provides` — поиск пакета, в котором находится нужный файл
10. `localinstall` — установка пакета не из репозитория (только в `yum`)

AppImage, Snap и Flatpak

В традиционной системе управления программным обеспечением имеется «подводный камень». Проблема заключается в том, что бывает очень трудно или иногда невозможно установить программу не совместимую с данным конкретным дистрибутивом. Например, вы хотите установить программу из исходных кодов, а библиотеки с нужной версией нет.

Проблему иногда можно решить с помощью виртуальных машин или контейнеров, но это на всегда правильный и удобный подход.

Чтобы преодолеть проблему не удовлетворенных зависимостей придумали несколько механизмов:

AppImage — приложения получаем в виде единого файла. Каждое приложение самодостаточно: оно включает в себя все библиотеки, от которых зависит приложение. Стандарт AppImage 1.0 представлял собой iso-образ стандарта Rock Ridge (zisofs), включая в себя минимальный AppDir и небольшую библиотеку среды выполнения. Вторая версия может использовать другие файловые системы, такие как SquashFS.

Flatpak — это утилита для развёртывания, управления пакетами и виртуализации для Linux. Предоставляет песочницу, в которой пользователи могут запускать приложения без влияния на основную систему. Приложения, использующие Flatpak, требуют дополнительных разрешений на использование

дискового пространства.

Snap — проект с похожими идеями, что и в Flatpak. Разработан в Canonical. В отличие от Flatpak использует специальный демон — `snapd`.

Проекты основаны на схожих принципах: переносимость, простота использования, минимальное влияние на хостовую ОС.

В Snap и Flatpak используют принципы изоляции такие же как в контейнерах.

Flatpak поддерживает дополнительные репозитории пакетов.

Flatpak ориентирован в основном на графические пользовательские приложения.

В Snap доступно как прикладное, так и системное ПО.

Преимущества этих решений:

Простота в использовании.

Некоторая независимость от ОС.

Можно устанавливать сразу несколько версий приложения. Недостатки:

Изоляция на полная.

Пакеты занимают больше места на диске.

Сборка пакета занимает больше времени и отнимает больше усилий.

Сборка и установка программного обеспечения из пакетов с исходным кодом.

Очень часто бывает необходимо самостоятельно собрать программу из архива с исходным кодом, например, потому, что пакеты с этим ПО еще не собраны.

В подавляющем большинстве случаев, хотя и не всегда, для сборки и установки программы из архива с исходным кодом достаточно выполнить следующие действия:

Получить файл архива.

Разархивировать его.

Прочесть файл README или INSTALL.

Сконфигурировать скрипт сборки.

Собрать программу.

Перейдя в сеанс суперпользователя установить программу.

При работе с программным обеспечением, собранным из архивов с исходным кодом, надо быть особо осторожным в системах, ориентированных на использование менеджеров пакетов, так как размещение конфигурационных и вспомогательных файлов в фирменных пакетах и в программах, собранных самостоятельно, обычно значительно отличается.

Управление библиотеками.

Практически все программы требуют для своей сборки или работы наличия в системе файлов библиотек.

Библиотеки представляют собой файлы со специальной структурой, содержащие скомпилированный бинарный код (так называемый объектный код), предназначенный для подключения к машинному коду, содержащемуся в исполняемых файлах программ.

Процесс подключения кода в библиотеках к коду самой программы называется компоновкой или связыванием (linking).

В зависимости от типа связывания различают библиотеки разных видов:

Статические библиотеки (static library).

Разделяемые библиотеки (shared library).

По соглашению файлы статических библиотек заканчиваются суффиксом `.a` (archive), а файлы разделяемых библиотек - `.so` (shared object).

Статические библиотеки представляют собой наборы объектных файлов, объединенных вместе с помощью утилиты `ar`.

Скомпилированный код, находящийся в библиотеке может быть использован в коде программ. Это позволяет избавиться от необходимости многократного повторения написания хорошо известного и часто используемого кода.

Так как требуемый код из статических библиотек помещается на стадии компоновки в результирующий программный код, находящийся в исполняемом файле, то для выполнения таких программ никаких библиотек уже не требуется. То есть, библиотека используется только на стадии сборки программы, но не на стадии исполнения, что является существенным преимуществом таких программ.

Статически собранные программы удобны для применения в случае аварийного восстановления системы и во всех остальных случаях, когда доступ к требуемым библиотекам затруднен или невозможен. Однако размер статически собранной программы обычно значительно превышает размер этой же программы, собранной с использованием разделяемых библиотек.

Идея разделяемой библиотеки состоит в том, что один и тот же код,

находящийся в файле библиотеки, используется при исполнении разных программ.

То есть, код из библиотеки не помещается в код программ на стадии компоновки, он вызывается из файла библиотеки по мере его необходимости. Тем не менее, код, находящийся в библиотеке жестко связан с кодом программы.

Использование разделяемых библиотек существенно экономит дисковое пространство, так как множество программ используют код гораздо меньшего числа библиотек.

В GNU/Linux при использовании компилятора gcc для сборки программ, написанных на языке C, по умолчанию производится сборка с использованием разделяемых библиотек.

Модуль 8. Понятие о процессах в РЕД ОС

Процессы и задания.

GNU/Linux – многопользовательская и многозадачная операционная система.

Примечание: Это значит, что одновременно в системе может выполняться множество программ, запущенных различными пользователями. Для обеспечения такой работы центральный процессор компьютера последовательно переключается на обработку программного кода различных приложений, системных и прочих программ, создавая впечатление, что все они выполняются одновременно. Ядро GNU/Linux спроектировано таким образом, что имеется гарантия выполнения центральным процессором кода каждой работающей в системе программы за некоторое конечное время. Естественно, чем больше загружена система, тем большим становится этот промежуток времени. Сильно загруженная система может (с точки зрения пользователя) потерять интерактивность, то есть она может настолько медленно отвечать на запросы пользователя, что он будет уверен в том, что система не работает.

Программы представляют собой исполняемые файлы двух типов:

1. Скомпилированные бинарные файлы, содержащие инструкции на машинном языке.
2. Интерпретируемые сценарии (например, сценарии Bash или программы Perl).

Примечание: Когда от пользователя поступает команда выполнить некоторую программу, операционная система в начале определяет с помощью "магических чисел" (magic numbers см. `man file`) к какому из этих двух типов относится данная программа. Если она относится к первому типу, то программа пополняет очередь других работающих программ, ожидающих своей очереди постановки на исполнение процессором. В противном случае сначала запускается программа – интерпретатор, предназначенный для исполнения данного вида сценариев (например, `/bin/bash`), который, в свою очередь, относится к программам первого типа.

Программа – это некоторый код, хранящийся в обычном (регулярном) исполняемом файле.

Процесс – это экземпляр программы, находящийся на исполнении в процессоре, либо ожидающий этого момента в очереди заданий.

Примечание: Программа – это исполняемый файл, а процесс – это исполняющийся машинный код.

Задание (task, job) – это процесс(ы), запущенный(е) пользователем с помощью одной команды.

Процессоры, аппаратно обеспечивающие многозадачность, поддерживают не менее двух режимов (уровней) выполнения.

Примечание: Процесоры архитектуры IA-32 поддерживаю четыре уровня выполнения.

Нулевой уровень соответствует привилегированному режиму, который используется ядром GNU/Linux

Пользовательские процессы исполняются в непривилегированном режиме и не могут повредить ядро.

GNU/Linux использует концепцию виртуальной памяти, т.е. адреса ячеек памяти, с которыми работает процесс не являются адресами физической памяти.

Адреса виртуального пространства трансформируются в реальные (физические) адреса с помощью набора карт трансляции адресов, которые реализуются в виде таблиц страниц.

Страница – это выделенный и защищенный блок памяти фиксированного размера.

Примечание: Современные процессоры архитектуры IA-32 поодерживают страницы размеров 4Kb и 4Gb.

Регистры блока управления памятью (часть процессора, memory management unit, MMU) хранят данные необходимые для трансляции адресов для процесса, который в данный момент исполняется на процессоре.

При вытеснении процесса новым процессом происходит замена указателей на новые карты трансляции (так называемое, переключение контекста)

Регистры MMU доступны только в режиме ядра.

Примечание: Это делает возможным изоляцию процессов, поскольку каждый пользовательский процесс имеет доступ только к своим страницам памяти и не может получить доступа к чужим.

Каждый процесс работает в своем собственном виртуальном адресном пространстве. Общий размер всех виртуальных пространств может превышать объем физической памяти.

Механизм подкачки (swapping) используется для освобождения места в физической памяти и помещения неиспользуемых страниц памяти или процессов целиком в область подкачки (swap), которая находится на жестком диске

Определенная часть виртуального пространства процесса отображается на пространство, в котором находится ядро.

Процессы не могут напрямую обращаться к ядру, обращение происходит с помощью интерфейса системных вызовов.

Обращение к системному вызову переводит процесс в режим ядра, которое производит работу от имени процесса, после завершения работы системного вызова исполнение процесса возвращается в пользовательский режим.

При запуске процесс переводится в состояние готовности к дальнейшей работе.

По истечении некоторого промежутка времени ядро выбирает запущенный ранее процесс для исполнения, переключает контекст и передает управление процессу.

Основные состояния процесса: готов к работе (R, running), спящее (S, sleeping), выполнение

Процессор выделяет каждому процессу кванты времени (time slices), в течение которых инструкции процесса выполняются процессором.

Только один процесс в каждый момент времени может выполняться на

процессоре, но одновременно может существовать много процессов

Используемая в Linux модель управления пользовательскими процессами относится к классу вытесняющей многозадачности (preemptive multitaskings). В рамках этой модели каждый процесс имеет свой уровень важности в системе или приоритета. Более высоко приоритетные процессы могут вытеснять с исполнения на процессоре менее приоритетные процессы.

Адресное пространство процесса состоит из следующих областей:

1. Заголовок процесса, содержащий, в частности, специальные сигнатуры, показывающие формат процесса (например, формат ELF или COFF).
2. Инструкции (text).
3. Инициализированные данные (data)
4. Неинициализированные данные (bss, block static storadge)
5. Разделяемая память (shared memory)
6. Разделяемые библиотеки (shared libraries)
7. Куча (heap) – область памяти, предназначенная для динамически выделяемой памяти, выделение и увеличение размера кучи производится ядром
8. Стек (stack) – структура типа LIFO (Last Input – First Output), предназначенная для вызовов подпрограмм и хранения некоторых переменных, выделяется ядром.

Заголовок процесса содержит размеры областей, точку входа (адрес первой инструкции), магическое число.

Управляющая информация о процессе поддерживается с помощью структур:

1. область u (u-area)
2. структуры proc (p-area)

Структура proc называется таблицей процессов и находится в системном пространстве (пространстве ядра, kernel space)

Область u является частью пространства пользовательского процесса, но процесс не может изменять информацию в этой области.

В области u хранится следующая информация:

1. аппаратный контекст (значения регистров MMU)
2. указатель на proc
3. RUID и EUID – реальный и эффективные идентификаторы пользователя (кто запустил процесс и от имени кого идет выполенние)
4. RGID и EGID – реальный и эффективные идентификаторы группы

5. таблица дескрипторов открытых файлов
6. информация из заголовка процесса 33.B p области хранится

PID (Process ID) – уникальный порядковый номер (идентификатор) процесса в системе.

PPID (Parent Process ID) – PID родительского процесса.

Текущее состояние процесса

приоритеты и флаги

расположение области u

При создании процесса ядро ассоциирует с ним три стандартных потока (открытых файла):

1. Стандартный поток ввода (`stdin`) – дескриптор `0`;
2. Стандартный поток вывода (`stdout`) – дескриптор `1`;
3. Стандартный поток ошибок (`stderr`) – дескриптор `2`.

Процессы создаются другими процессами с помощью системного вызова `fork()`.

Процесс, созданный другим процессом, называется по отношению к последнему потомком или дочерним процессом. И, наоборот, тот процесс, который создал некоторый дочерний процесс, является для него родительским.

У каждого процесса есть один и только один родитель, но у процесса может быть более одного потомка.

Иерархия процессов представляет собой перевернутое дерево, в корне которого находится процесс init (PID=1)

Когда процесс – родитель завершает свою работу, он должен завершить работу своих дочерних процессов.

Процессы можно разделить на три типа:

1. Процессы ядра или системные процессы.

Примечание: Они происходят в kernel space и не видны пользователю. Примером такого процесса в Linux является процесс постановки процессов в очередь (sheduling). Процессы ядра, разумеется, не связаны ни с каким терминалом, однако, они при желании могут выводить информацию на терминал. Для этого в коде ядра есть специальные подпрограммы. Процессы ядра всегда располагаются в оперативной памяти.

2. Демоны (сервисы) – это фоновые неинтерактивные процессы несвязанные с терминалами и запускаемые системными пользователями.
3. Прикладные процессы - порождаются в рамках сеанса работы пользователя, связаны с терминалом.

Фоновый режим выполнения заданий.

В GNU/Linux пользователь может запускать процессы или в интерактивном (foreground) или в фоновом (background) режиме.

Только одно задание в рамках сеанса может находиться в интерактивном режиме , все остальные активные задания выполняются в фоновом режиме.

Для запуска команды в фоновом режиме в конце командной строки необходимо поставить символ `&`, при этом выводится номер задания:

Sleep 2000&

Пример:

$find ~/ –name "*t" & [1]546

Примечание: В этом примере команда `find` запущена в фоновом режиме, так как в конце командной строки установлен символ `&`. Номер задания выводится в квадратных скобках, в этом примере – `1`. Число, выводящееся после квадратных скобок - PID процесса задания.

Для мониторинга состояний фоновых заданий предназначена команда `jobs`, которая позволяет просмотреть статус фоновых заданий.

Она отображает номер задания, имя команды и статус задания:

Пример:

$find / –empty –user basile –exec rm –f {} \; & [1]548

$find / –empty –user anna –exec rm –f {} \; & [2]551

$jobs

[1]-Done find / –empty –user basile rm –f {} \; [2]+Running find / –empty –user anna rm –f {} \;

Примечание: В этом примере были запущены два задания на поиск и удаление пустых файлов пользователей `basile` и `anna`. Заданиям назначены номера `1` и `2`. Команда `jobs` показала, что задание `1` выполнено, а задание `2` выполняется.

Обозначения `%%` и `%+` показывают последнее запущенное фоновое задание, а `%-` - предпоследнее задание.

В выводе команды `jobs` символы `+` и `−` для индикации последнего задания и предпоследнего заданий.

Команда `fg %номерзадания` переводит задание с номером `номерзадания` в интерактивный режим из фонового (можно пользоваться командой `%номерзадания`).

Пример: Так, команды `fg %1` и `%1` переводят задание с номером `1` в интерактивный режим.

Наоборот, для перевода задания в фоновый режим необходимо приостановить его выполнение сочетанием `Ctrl-Z`, а затем выполнить команду

`bg` с аргументом

`%номерзадания` (можно пользоваться командой `%номерзадания &`).

Пример: Команды `bg %1` и `%1 &` переводят (после приостановки нажатием `Ctrl-Z`) задание с номером `1` в фоновый режим из интерактивного режима.

Вместо использования конструкции `%номерзадания` можно обращаться к заданиям по первым символам их команд, предваряя их символом `%`:

Пример:

```
$jobs
[2]+Running        find / –empty –user anna rm –f {} \;
$fg %fi
```

Примечание: В этом примере выполняющееся в фоновом режиме задание было переведено в интерактивный режим командой `fg %fi`, которая использовала идентификацию задания не по его номеру, а по двум первым буквам в имени команды.

Для прекращения работы фонового задания необходимо использовать команду `kill`

```
%номерзадания
```

Пример:

```
$jobs
[2]+Running        find / –empty –user anna rm –f {} \;
$kill %2
$jobs
[2]+Terminated     find / –empty –user anna rm –f {} \;
```

Примечание: В этом примере задание с номером `2` было завершено командой `kill`.

Жизненный цикл процесса.

Родительский процесс создает дочерний процесс с помощью системного вызова

`fork()` (рис 1.).

Процесс-потомок является точной копией родительского процесса.

Вызов `fork()` приводит к тому, что ядро дублирует адресное пространство процесса, произведшего этот вызов, в свободное адресное пространство в памяти.

В дочернем процессе, как правило, незамедлительно выполняется системный вызов

`exec()`, который заменяет код только что созданного процесса на код новой программы

Примечание: вызов exec()может не осуществляться, если дочерний процесс должен выполнять ту же работу, что и родительский процесс, т.е. если не нужно менять код дочернего процесса.

В то же время работа родительского процесса приостанавливается системным вызовом

`wait()` до завершения работы дочернего процесса.

Примечание: родительский процесс может и не останавливаться на время работы дочернего, если дочерний процесс запускается в фоновом режиме (background)

Процесс завершается путем обращения к системному вызову `exit()`

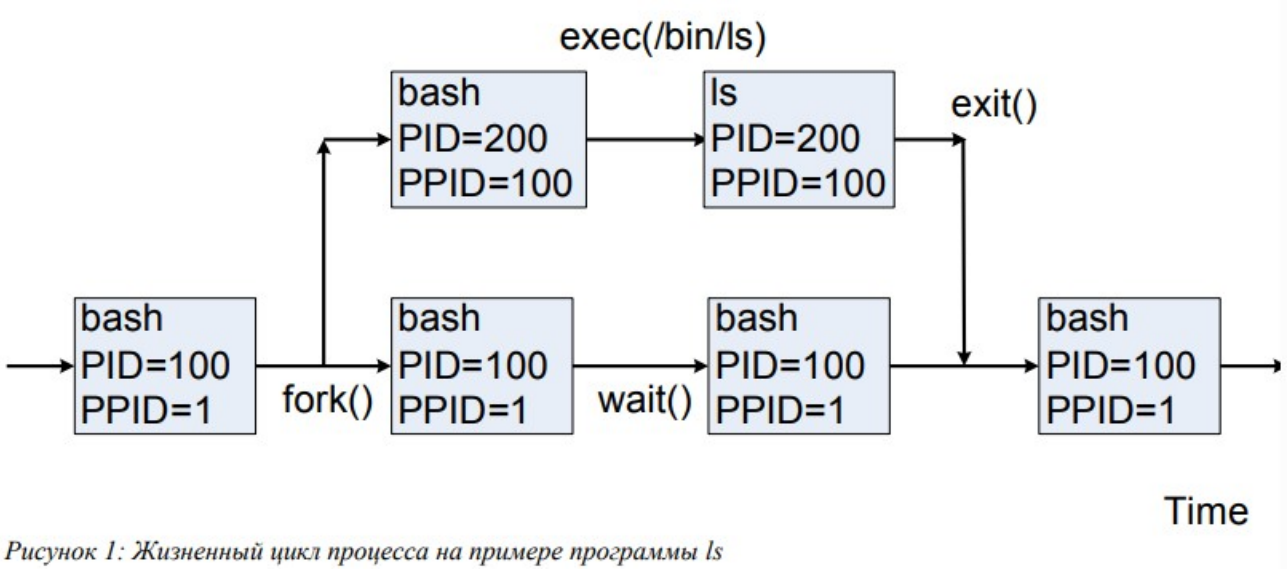


Рисунок 1: Жизненный цикл процесса на примере программы ls

Системный вызов `exit()` освобождает адресное пространство, закрывает все открытые файлы и переводит процесс в состояние “зомби” (zombie, defunct), будит, если нужно, родительский процесс.

Вызов `exit()` не освобождает структуру proc. За эту операцию отвечает родительский процесс.

Примечание: Вполне возможна ситуация, когда дочерний процесс уже завершил свою работу, а родительский процесс еще не успел произвести системный вызов `wait()`. В таком случае информация об уже несуществующем дочернем процессе (структура proc) сохраняется в таблице процессов, а запись в таблице о завершившемся дочернем процессе так и остается помеченной как `defunct` или, иначе, процесс – зомби (`zombie`). Структура proc, помеченная как zombie, остается в таблице до перезагрузки системы

Если родительский процесс завершается раньше процесса-потомка, то потомок удочеряется процессом `init`.

Мониторинг процессов.

Основным инструментом для исследования процессов является команда `ps`,

которая выводит мгновенное состояние процессов.

Команда `ps` без аргументов выводит список процессов, связанных с текущим терминалом.

Пример:

```
$ ps
PID TTY     TIME CMD
1751 pts/0   00:00:00 bash
1890 pts/0   00:00:00 ps
```

Столбец `PID` отображает идентификаторы процессов.

Столбец `TTY` – имена терминалов

Столбец `TIME` – суммарное процессорное время, затраченное процессом с момента его старта.

Столбец `CMD` – командная строка, соответствующая данному процессу.

Более подробную информацию можно получить с помощью опции подробного вывода (full format) `-f`.

Пример:

```
$ ps -f
UID PID   PPID   C STIME TTY TIME CMD
user1      1751      1745    0 21:01 pts/0          00:00:00 bash
user1      1970      1751    0 23:14 pts/0          00:00:00 ps -f
```

`UID` – владелец процесса.

`STIME` показывает время запуска процесса столбец `C` – уровень загрузки процессора

Величины, находящиеся в столбце `C` целочисленные, они участвуют в вычислении приоритета процесса планировщиком. Чем выше эта величина, тем ниже приоритет.

Примечание: Когда процесс ожидает своей очереди постановки на процессор, эта величина постепенно снижается на единицу на каждом цикле работы планировщика, то есть приоритет процесса постепенно повышается. После работы процесса на процессоре в этом столбце устанавливается положительное значение, которое затем постепенно снижается.

Детальную информацию о процессах можно получить, используя опцию `-l` (long format).

Пример:

```
$ ps -l
F S UID   PID  PPID  C PRI  NI ADDR  SZ  WCHAN TTY TIME CMD
0 S 500   1751 1745  0  76  0  -  781 11c541 pts/0 00:00:00 bash
0 R 500   1988 1751  0  77  0  -  617 -      pts/0 00:00:00 ps
```

Столбец `F` – показывает флаги процесса, отображающие его особые свойства (см. man ps).

Столбец `S` – статус процесса, который может быть:

1. `D` – процесс приостановлен и не может быть прерван (например, ожидает окончания ввода-вывода).
2. `R` – процесс выполняется или находится в очереди.
3. `S` – процесс “спит” (доступ к процессору не требуется).
4. `T` – процесс трассируется.
5. `Z` – процесс defunct (zombie).

Столбец `NI` показывает величину Nice Number. Эта константа устанавливается пользователем или администратором и участвует в вычислении приоритета процесса планировщиком.

Примечание: Другое ее название – относительный приоритет.

Столбец `SZ` – количество памяти, занимаемое процессом

`WCHAN` – это адрес или имя функции ядра, обслуживающей текущее состояние спящего процесса (статус `S`).

Опция `-e` позволяет вывести список всех процессов в системе.

Примечание: Также для этого можно пользоваться опцией `-A`. Чаще всего для получения списка всех процессов используют команду `ps -ef`, дающую подробную информацию о процессах.

Наиболее популярные опции программы `ps` для предназначенные для фильтрации выводимой информации:

1. `-u` – фильтрация по UID.
2. `-t` – по терминалу.
3. `-p` – по PID искомого процесса.
4. `-C` – по командной строке.

Пример: Требуется вывести список процессов, запущенных на второй виртуальной консоли:

```
$ ps -ft tty2
UID        PID  PPID  C STIME TTY          TIME CMD
root      1557     1  0 06:18 tty2       00:00:00 /sbin/mingetty tty2
```

Все приведенные выше опции команды ps соответствуют POSIX формату, однако GNU версия программы `ps` поддерживает также опции и в BSD стиле.

Наиболее популярным набором опций команды ps в таком формате является `ps aux` – она выводит список всех процессов в системе с указанием их владельцев.

Пример: Ниже приведен небольшой фрагмент из ее вывода:

```
$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY    STAT START   TIME COMMAND
root   1   0.0  0.1   1268  60 ?    S      Oct08 0:04 init
```

Примечание: Эта команда отобразила в виде процентов уровень загрузки процессора и памяти данным процессом. Столбец `VSZ` – объем используемой процессом виртуальной памяти и `RSS` – физической памяти.

Опции POSIX, BSD и длинные опции команды `ps` можно комбинировать.

Пример:

```
$ ps   -f  U root
UID        PID  PPID  C  STIME TTY          STAT    TIME CMD
root         1     0     0  06:18 ?        S       0:04   init
root         2     1     0  06:18 ?        SW      0:00   [keventd]
root         3     1     0  06:18 ?        SWN     0:00   [ksoftirqd_CPU0]
root         4     1     0  06:18 ?        SW      0:00   [kswapd]
root         5     1     0  06:18 ?        SW      0:00   [bdflush]
root         6     1     0  06:18 ?        SW      0:00   [kupdated]
root         7     1     0  06:18 ?        SW      0:00   [kinoded]
root         8     1     0  06:18 ?        SW<     0:00   [mdrecoveryd]
```

```
root        11     1      0  06:18 ?        SW       0:00 [kreiserfsd]
root        846    1      0  06:18 ?        SW       0:00 [khubd]
root        1016   1      0  06:18 ?        S        0:00 /sbin/cardmgr
root        1290   1      0  06:18 ?        SW       0:00 [kapmd]
root        1314   1      0  06:18 ?        S        0:00 crond
root        1499   1      0  06:18 ?        S        0:00 /usr/lib/postfix/master
root        1518   1      0  06:18 ?        S        0:00 gpm -m /dev/psaux -t
imps
2
root        1556   1      0  06:18 tty1     S        0:00 /sbin/mingetty tty1
root        1557   1      0  06:18 tty2     S        0:00 /sbin/mingetty tty2
root        1558   1      0  06:18 tty3     S        0:00 /sbin/mingetty tty3
root        1559   1      0  06:18 tty4     S        0:00 /sbin/mingetty tty4
root        1560   1      0  06:18 tty5     S        0:00 /sbin/mingetty tty5
root        1561   1      0  06:18 tty6     S        0:00 /sbin/mingetty tty6
root        1563   1      0  06:18 ?        S        0:00 kdm -nodaemon
root        1590   1563   0  06:18 ?        R        0:03 /etc/X11/X -auth
/var/run/xaut
root   1591        1563   0  06:18 ?        S        0:00 -:0
```

Примечание: В этом примере для выбора процессов root была использована опция в BSD стиле `U`, а POSIX опция `-f` была использована для указания подробного формата вывода.

Для постоянного мониторинга процессов используется утилита `top`, которая отображает исполняющиеся процессы, использующие большую часть процессорного времени и наиболее сильно использующие память.

Утилита `top` регулярно обновляет информацию о процессах. Для выхода из `top` необходимо набрать `q`.

Команда `top` позволяет, не выходя из интерактивного просмотра процессов, посылать процессам сигналы с помощью нажатия на клавишу `k`.

Нажатие на `i` отключает вывод утилитой `top` неактивных процессов.

В первой строке экрана вывода команды приводятся данные о средней загруженности системы `load averages` за последние 1, 5 и 15 минут.

Команда `w`, демонстрирующую список всех вошедших в сеанс пользователей

и запущенные ими процессы.

Пример:

```
$ w
18:17:10   up 22 min,   2 users,   load average: 0.30, 0.38, 0.37
USER        TTY   FROM        LOGIN@    IDLE JCPU PCPU   WHAT
user1 pts/0  :0.0    5:56pm   1.00s   0.05s   0.00s   w
```

Примечание: В этом случае команда `w` отобразила наличие в системе одного пользователя. Строка `2 users`, выведенная командой `w`, "обманута" тем, что пользователь вошел в графический сеанс.

Для получения информации о процессах можно использовать каталог `/proc`. `/proc` – это псевдофайловая система, порождаемая ядром.

`/proc` позволяет также получать и устанавливать (суперпользователю) параметры ядра на лету.

Пример:

```
$ ls  /proc
1     1508  1590  3710  846     driver       kcore   mtrr        sys
1016  1518  1591  3718  966     execdomains  kmsg    net         sysvipc
11    1537  1620  4     967     fb           ksyms   partitions  tty
1260  1556  1687  5     apm     filesystems  loadavg pci         uptime
1277  1557  1745  5090  asound  fs           locks   scsi        version
1290  1558  1751  5233  bus     ide          mdstat  self
1296  1559  2     6     cmdline interrupts   meminfo slabinfo
1314  1560  3     7     cpuinfo iomem        misc    splash
1499  1561  3429  7156  devices ioports      modules stat
1507  1563  3709  8     dma     irq          mounts  swaps
```

Примечание: Подкаталоги `/proc` с именами, состоящими из цифр, соответствуют исполняемым процессам.

Пример: Определим PID текущей оболочки и исследуем соответствующий ее процессу каталог.

```
$ ps
PID TTY     TIME CMD
```

```
1751 pts/0   00:00:00 bash
7157 pts/0   00:00:00 ps
$ ls /proc/1751
cmdline cwd environ exe fd   maps   mem   mounts   root   stat statm   status
$ cat /proc/1751/cmdline bash
$ cat /proc/1751/status Name:       bash
State: S (sleeping) Tgid:    1751
Pid:  1751
PPid:         1745
TracerPid:         0
Uid:500            500         500         500
Gid:500            500         500         500
FDSize: 256
Groups: 500 4 10 51 55 100 103
VmSize:     3124 kB
VmLck:      0 kB
VmRSS:      716 kB
VmData:     628 kB
VmStk:      24 kB
VmExe:      428 kB
VmLib:      1668 kB
SigPnd:   0000000000000000
SigBlk:   0000000000010000
SigIgn:   8000000000384004
SigCgt:   000000004b813efb
CapInh:   0000000000000000
CapPrm:   0000000000000000
CapEff:   000000000000000
```

Примечание: Видно, что файл `cmdline` содержит в себе командную строку, породившую процесс, а файл `status` – подробную информацию о статусе процесса.

Подкаталог `fd` предназначен для мониторинга файлов, открытых процессом.

В каталоге `fd` содержатся символические ссылки на реально открытые

файлы процессом.

Имена этих символических ссылок соответствуют номерам файловых дескрипторов открытых файлов.

Команда `fuser` позволяет определить, какие процессы используют какой-либо файл.

Пример: пользователь смонтировал дискету в каталоге `/mnt/floppy`, перешел в этот каталог и получил информацию о процессе, использующем этот каталог.

```
$ mount /mnt/floppy
$ cd /mnt/floppy/
$ /sbin/fuser .
.:      1751c
$ ps
PID TTY     TIME CMD
1751 pts/0   00:00:00 bash
7188 pts/0   00:00:00 ps
```

Примечание: Команда `fuser` вывела PID процесса, соответствующему оболочке, так как для нее этот каталог является текущим. Этот факт команд `fuser` подтверждает, выводя букву `c` после PID процесса.

Команда `fuser` выводит следующие символы после PID процессов:

1. `c` – текущий каталог.
2. `e`- исполняемый файл в момент его работы.
3. `f` – открытый файл.
4. `r` – корневой каталог.
5. `m` – разделяемая библиотека, либо отображаемый в память файл.

Опция `-m` команды `fuser` позволяет указать, что имя файла является именем смонтированного блочного устройства:

Пример:

```
$ /sbin/fuser -m /dev/fd0
/dev/fd0:     1751c
```

Примечание: В этом примере указана опция `-m`, так как в качестве аргумента команды используется имя смонтированного блочного устройства, соответствующего флоппи диску.

Для просмотра списка процессов в виде иерархического дерева, отображающего отношения родительских и дочерних процессов, необходимо выполнить команду `pstree`

Пример:

```
$pstree
init-+-2*[automount]
|-cron
|-6*[getty]
|-inetd
|-kdm-+-XF86_SVGA
|     `-kdm-+-kwm-+-kaudioserver---maudio
|     |     |-kbgndwm
|     |     |-kfm---konsole---bash---pstr
|     |     |-kpanel
|     |     |-krootwm
|     |     `-kwmsound
|     `-netserv
|-kflushd
|-klogd
|-kpiod
|-kswapd
|-lpd
|-portmap
|-proftpd
|-sleep
|-syslogd
`-xfs
```

Примечание: Иерархический список процессов, выведенный командой `pstree`.

Сигналы.

Сигналы – это один из способов межпроцессного взаимодействия.

Примечание: Они обеспечивают возможность передачи процессами друг-другу команд и сообщений, таких, как, например, сообщение о завершении работы дочернего процесса, или команда перечитать файл конфигурации, или сообщение об ошибке операции с плавающей запятой. Одно из главных направлений использования сигналов состоит в снятии процессов с исполнения.

Список сигналов, используемых в системе, можно получить с помощью команды

kill -l

Подробное описание сигналов доступно с помощью `man 7 signal`.

Наиболее часто приходится использовать следующие сигналы:

`1` – `HUP` – разрыв связи с терминалом (Hang Up – положить трубку). Многие демоны используют этот сигнал, как команду перечитать их конфигурационный файл и продолжить работу с измененными настройками. Оболочка Bash реагирует на этот сигнал завершением сеанса.

`2` – `INT` – клавиатурное прерывание процесса. В GNU/Linux генерируется при нажатии `Ctrl-C`.

`3` – `QUIT` – "отключение" клавиатуры. Получение этого сигнала обычно снимает процесс с исполнения.

`15` – `TERM` – сигнал для завершения процесса. Получив этот сигнал приложение должно (но не обязано) завершить свою работу корректно, закрыв потоки ввода- вывода. Этот сигнал `команда kill` посылает по умолчанию.

`9` – `KILL` – безусловное и немедленное снятие процесса с исполнения без корректного завершения процесса.

Примечание: Каким образом то или иное приложение реагирует на получение некоторого сигнала зависит от того, как эта программа написана. В программе получение сигнала может перехватываться и обрабатываться специальным образом. Сигнал `KILL` не может быть перехвачен. Этот сигнал приводит к немедленному и, таким образом, часто некорректному снятию процесса с исполнения. При этом файлы, открытые процессом не закрываются нормальным способом, что может привести к существенной потере данных или даже сбою настроек терминала.

Команда `kill` посылает заданный сигнал процессу, номер которого указывается после дефиса.

Если номер сигнала или его имя не заданы после дефиса, то команда `kill` посылает целевым процессам сигнал `15` (`TERM`).

Пример: Посылаем сигнал оболочке Bash:

```
$ ps
PID TTY     TIME CMD
2179 pts/0   00:00:00 bash
2180 pts/0   00:00:00 ps
$ kill 2179
$ kill -2 2179

$ kill -1 2179 login:
```

Примечание: Этот пример демонстрирует, что оболочка Bash игнорирует сигналы `15 (TERM)` и `2`

(INT) , а получив сигнал 1 (HUP) она завершает работу и выходит из сеанса.

Посылать сигналы процессам могут только их владелец и суперпользователь.

Если родительским процессом получен сигнал, приводящий к его остановке, то сняты с выполнения будут также и все его дочерние процессы.

Команда killall послать требуемый сигнал процессам, в командных строках которых присутствует заданная строка.

Пример: следующая команда приведет к немедленному останову всех копий демона httpd :

```
# killall -9 httpd
```

Примечание: В результате передачи сигнала 9 (KILL) всем процессам httpd они будут сняты с исполнения, так как перехватить такой сигнал невозможно.

Перехват и обработка сигналов в Bash.

Встроенная команда оболочки Bash trap позволяет перехватывать сигналы и реагировать на них каким-либо заданным способом.

Первым аргументом является команда, которую следует выполнить при получении оболочкой сигнала.

Второй аргумент задает сигнал, который должен быть обработан.

Пример: выполняются следующие команды для установки ловушек сигналов INT, QUIT и на событие выхода из оболочки EXIT:

```
$ trap "echo Получен сигнал INT" INT
$ trap "echo А это был QUIT" QUIT
$ trap "echo Пока!" EXIT
$ trap -p
trap -- 'echo Пока!' EXIT
trap -- 'echo Получен сигнал INT' SIGINT trap -- 'echo А это был QUIT' SIGQUIT
$ Получен сигнал INT

$ ps
PID TTY     TIME CMD
1811 pts/0   00:00:00 bash
1812 pts/0   00:00:00 ps
```

$ kill -3 1811 А это был QUIT

$ exit Пока! login:

Примечание: Команды `trap` установили ловушки для сигналов – команды `echo`. Для сигнала `INT` на экран будет выведено `Получен сигнал INT`, для `QUIT` - `А это был QUIT`, а при выходе из оболочки на экране будет отображаться `Пока!` . Команда `trap -p` вывела список установленных обработчиков сигналов. Далее пользователь нажал сочетание `Ctrl-C`, передающее сигнал `INT` оболочке. При этом сигнал был перехвачен обработчиком и сработала соответствующая команда `echo`. Далее оболочке был передан сигнал `QUIT`, и снова сработал соответствующий обработчик. Выход из оболочки также привел к срабатыванию ловушки.

Управление приоритетом процессов.

Часть ядра, называемая планировщиком (scheduler), определяет, какой из готовых к работе процессов (runnable process) будет выполняться на центральном процессоре (CPU)

Планировщик в ядре Linux поддерживает три различных класса (политики планирования, scheduling policy) процессов:

1. SCHED_FIFO (First In – First Out scheduling)
2. SCHED_RR (Round Robin scheduling)
3. SCHED_OTHER

Классы SCHED_FIFO и SCHED_RR используются для процессов программного (псевдо) реального времени (soft real time)

Процессы класса SCHED_OTHER используются для стандартных процессов, работающих в режиме разделения времени и является классом процесса по умолчанию.

Команда `chrt` позволяет запустить процесс с заданным классом или изменить класс у существующего процесса.

Опция `-c` команды `ps` выводит класс процесса в столбце `CLS`

Пример:

```
# ps -c
  PID  CLS PRI TTY      TIME CMD
4644    -   24 pts/6    00:00:00 bash
4730    -   24 pts/6    00:00:00 ps
```

chrt 10 bash [root@trainer root]# ps -c

PID CLS PRI TTY TIME CMD

4644	-	24 pts/6	00:00:00	bash
4731	RR	50 pts/6	00:00:00	bash
4763	RR	50 pts/6	00:00:00	ps

Каждому процессу присваивается некоторое значение статического приоритета (static sched_priority)

Статический приоритет может изменяться в пределах от 0 до 99.

Процессы класса SCHED_OTHER имеют статический приоритет равный 0

Процессы класса SCHED_FIFO и SCHED_RR имеют статический приоритет от 1 до 99

Для вывода статического приоритета можно использовать опцию `-o rtprio` команды ps

Пример:

```
# ps -o pid,rtprio,cmd PID    RTPRIO CMD
46440 bash
473110 bash
476410 ps -o pid,rtprio,cmd
```

Планировщик для каждого статического приоритета поддерживают свою очередь процессов готовых к вполнению (runnable process)

При выборе процесса для исполнения планировщик просматривает непустую очередь с наивысшим статическим приоритетом, исполняя процесс, находящийся в начале очереди.

Планирование имеет вытесняющий характер. Если у готового к работе процесса более высокий статический приоритет, то он вытеняет с процессора текущий процесс.

Политика планирования (класс) определяет перемещение процесса внутри очереди. Процесс класса SCHED_FIFO занимает процессор до тех пор, пока не будет вытеснен более приоритетным процессом или не будет заблокирован в ожидании завершения выполнения запроса на операцию ввода/вывода.

Процесс класса SCHED_RR практически соответствует классу SCHED_FIFO за исключением того, что каждому процессу этого класса позволено занимать процессор не более максимального кванта (slice, quant) времени.

Все процессы класса SCHED_OTHER находятся в одной очереди,

соответствующей статическому приоритету 0.

Выбор процесса из очереди с нулевым статическим приоритетом происходит на основе динамического приоритета.

Динамический приоритет вычисляется с использованием постоянного числа nice и фактора, характеризующего как давно готовый к работе процесс не был на процессоре.

Пример:

```
$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S   500  1747  1741  0  74   0    -   783 11c541 pts/0    00:00:00 bash
0 R   500  2510  1747  4  77   0    -   618      - pts/0    00:00:00 ps
```

Чем ниже значение nice number, тем более высокий приоритет будет у процесса. В GNU/Linux значение nice number задается в пределах от -20 до 19.

По умолчанию nice number устанавливается в 0.

Обычных пользователи могут только увеличивать nice, уменьшать nice может только суперпользователь.

Значение nice number можно установить с помощью команды `nice`, после которой в качестве аргумента задается команда, которая должна быть исполнена с измененным приоритетом.

По умолчанию команда `nice` увеличивает значение nice number на 10.

Если требуется указать иное значение увеличения nice number , то его следует указать после опции `-n`.

Команда `nice`, вызванная без аргументов, отображает заданное значение nice number для данной оболочки.

Пример: Запустим Bash с пониженным приоритетом:

```
$ nice -n  19 bash
$ ps -l
F S  UID   PID     PPIDC PRI  NI ADDR SZ   WCHAN TTY  TIME     CM
                                                             D
```

```
0 S  500    1747    1741 0  74   0    -      784   11c541 pts/0    00:00:00    bash
0 S  500    2559    1747 0  78   19   -      780   11c541 pts/0    00:00:00    bash
     0 R    500  2560  2559  0  79   19   -      618   - pts/0        00:00:00 ps
     $ nice 19
```

Примечание: Здесь продемонстрировано, как с помощью команды `nice -n 19 bash`, была запущена оболочка Bash с пониженным приоритетом. В поле `NI` для этой оболочки команда `ps -l` выводит значение 19. Поле `PRI` этого же листинга показывает, что приоритет запущенной оболочки Bash действительно снижен (78 у Bash с пониженным приоритетом против 74 у исходной оболочки).

Для установки иного значения nice number для уже исполняющегося процесса следует использовать команду `renice`.

Пример: для изменения nice number оболочки Bash из предыдущего примера, суперпользователь может выполнить следующую команду:

```
# renice 10 2559
2559: old priority 19, new priority 10 # ps -l -p 2559
F S   UID   PID   PPID   C PRI   NI ADDR SZ WCHAN   TTY        TIME CMD
0 S   500   2559  1747   0  76   10   -      782 read_c pts/1    00:00:00 bash
```

Примечание: Данная команда установила новое значение приоритета (поле `NI`) для оболочки, которая была исходно запущена с nice number 19. Заметно, что приоритет процесса (см. поле `PRI`) также изменился.

С помощью команды `renice` можно изменять приоритет всех процессов для заданного после опции `-u` пользователя, или после `-g` группы пользователей.

Пример:

```
# renice 0 -u user1
```

Примечание: Эта команда установит значение nice number для всех процессов, принадлежащих пользователю `user1`, в 0.

Модуль 9. Графическая система пользователя в РЕД ОС.

Организация X Window.

Разработка системы X Window началась в первой половине 80-х в Массачусетском Технологическом Институте (MIT) в рамках проекта Athena, который финансировался фирмами IBM и DEC. Целью проекта являлось построение распределенной графической среды, позволяющей единообразно работать с различным оборудованием и различными операционными системами.

Текущая версия X протокола - 11, выпуск 6, поэтому он называется X11R6.

Помимо протокола X в настоящее время развивается новый протокол Wayland, в котором отказались от многих неиспользуемых функций протокола X.

В Ubuntu так же велась работа на проектом Mir, еще одной замены X, но работа над ним прекратилась в пользу Wayland.

Система X Window построена в рамках архитектуры клиент-сервер.

Сервер предназначен для взаимодействия с устройствами ввода-вывода.

Клиентские программы взаимодействуют с X сервером с помощью X протокола.

Он представляет собой обычный протокол прикладного уровня TCP/IP, которому по умолчанию присвоен порт 6000 TCP. Поэтому X клиент и X сервер могут с успехом работать по сети. То есть, программы клиент и сервер вполне могут находиться на различных машинах и никаких дополнительных программ для этого не требуется.

Из соображений безопасности современные версии Xorg запускаются без прослушивания порта TCP.

X клиенты могут быть условно подразделены на четыре категории:

Обычные X приложения (X applications). Примерами таковых являются браузер firefox, программа просмотра PDF xpdf, текстовый редактор emacs.

Оконные менеджеры (Window Managers). Они предназначены для обеспечения возможности управления окнами с помощью устройств ввода и предоставления удобного пользовательского интерфейса. Примерами оконных менеджеров являются Window Maker (исполняемый файл wmaker), Black Box (blackbox), Ice WM (icewm).

Рабочие среды (Desktop Environment), представляющие собой большие комплексы программного обеспечения, включающие в себя собственные оконные менеджеры, файловые менеджеры, средства офисной работы и иное пользовательское прикладное программное обеспечение. X приложения,

созданные для работы в составе рабочей среды обеспечивают единообразный пользовательский интерфейс, что и отличает рабочие среды от оконных менеджеров. Наиболее распространены KDE (K Desktop Environment) и GNOME (GNU Objects Model Environment).

Менеджеры сеанса (X Session Managers). Программы особого рода, предназначенные для обеспечения возможности непосредственного запуска X сеанса без необходимости предварительного входа в обычный неграфический сеанс Shell. Кроме этого менеджеры X сеанса отвечают за перезапуск X сервера в случае его остановки. Наиболее распространены xdm, kdm и gdm. Менеджеры X сессий используют протокол XDMCP (X Display Manager Control Protocol). Запуск X сеанса осуществляется иначе, чем при использовании обычного текстового терминала. При обычном входе в сеанс запускается Shell, связанный с текстовым терминалом, то в X – менеджер окон, эмулятор терминала, рабочая среда (desktop) или иное X приложение.

X протокол открыт и, поэтому, существует множество его реализаций, предназначенных прежде всего для запуска на различных аппаратных платформах. На платформе IA-32 могут быть использованы различные X системы, например:

XFree86 - до последнего времени применялась в GNU/Linux повсеместно;

Xorg - иная реализация X системы, которая может быть использована в GNU/Linux, причем в настоящее время в ведущих дистрибутивах GNU/Linux наблюдается миграция на данную реализацию X системы;

Metro X - одна из коммерческих реализаций X системы.

Написание программ для X системы обычно требует интенсивного использования инструментальных библиотек. Различные библиотеки предоставляют различные средства и, самое главное, требуют использования различных прикладных интерфейсов программирования (API - Application Program Interface). То есть, не смотря на то, что X протокол является стандартным существует множество несовместимых платформ для X системы.

В GNU/Linux имеется две наиболее распространенные платформы для создания X приложений:

Qt - эту платформу использует KDE;

Gtk+ - эта платформа используется множеством различных X приложений и средой GNOME.

Конфигурирование X Window.

В Xorg конфигурационный файл называется `xorg.conf`.

Как правило в современных Linux файл не применяется, вместо этого конфигурация создается «на лету». Более того создание файла конфигурации может привести к невозможности запустить графическую оболочку.

Эти файлы в GNU/Linux размещаются в каталоге `/etc/X11`.

Для создания этого файла обычно применяются автоматизированные процедуры, хотя файл конфигурации может быть создан вручную.

В Xorg для конфигурации можно использовать `sudo Xorg -configure`. В результате будет создан файл `/root/xorg.conf.new`.

Опция `-config` позволяет указать файл конфигурации для запуска `Xorg`.

Пример: Ниже приведен пример содержимого файла `xorg.conf` :

$ sudo cat /root/xorg.conf.new Section "ServerLayout"

Identifier "X.org Configured" Screen 0 "Screen0" 0 0

Screen 1 "Screen1" RightOf "Screen0" InputDevice "Mouse0" "CorePointer" InputDevice "Keyboard0" "CoreKeyboard"

EndSection

Section "Files"

ModulePath "/usr/lib64/xorg/modules" FontPath "catalogue:/etc/X11/fontpath.d" FontPath "built-ins"

EndSection

Section "Module"

Load "glx" EndSection

Section "InputDevice" Identifier "Keyboard0" Driver "kbd"

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 "Monitor0" VendorName "Monitor Vendor" ModelName "Monitor Model"
    EndSection

    Section "Monitor"
    Identifier "Monitor1" VendorName "Monitor Vendor" ModelName "Monitor Model"
    EndSection

    Section "Device"
    ### Available Driver options are:-
    ### Values: <i>: integer, <f>: float, <bool>: "True"/"False", ### <string>: "String", <freq>: "<f> Hz/kHz/MHz",
    ### <percent>: "<f>%" ### [arg]: arg optional
      #Option "Accel"                # [<bool>]
      #Option "AccelMethod"          # <str>
      #Option "Backlight"            # <str>
      #Option "CustomEDID"           # <str>
      #Option "DRI"                  # <str>
      #Option "Present"              # [<bool>]
      #Option "ColorKey"             # <i>
      #Option "VideoKey"             # <i>
      #Option "Tiling"               # [<bool>]
      #Option "LinearFramebuffer"    # [<bool>]
      #Option "HWRotation"           # [<bool>]
      #Option "VSync"                # [<bool>]
      #Option "PageFlip"             # [<bool>]
      #Option "SwapbuffersWait"      # [<bool>]
      #Option "TripleBuffer"         # [<bool>]
      #Option "XvPreferOverlay"      # [<bool>]
     #Option "HotPlug"               # [<bool>]
```

```
  #Option     "ReprobeOutputs"        #  [<bool>]
  #Option     "XvMC"                  #  [<bool>]
  #Option     "ZaphodHeads"           #  <str>
  #Option     "VirtualHeads"          #  <i>
  #Option     "TearFree"              #  [<bool>]
  #Option     "PerCrtcPixmaps"        #  [<bool>]
  #Option     "FallbackDebug"         #  [<bool>]
  #Option     "DebugFlushBatches"     #  [<bool>]
  #Option     "DebugFlushCaches"      #  [<bool>]
  #Option     "DebugWait"             #  [<bool>]
  #Option     "BufferCache"           #  [<bool>]
  Identifier  "Card0"
  Driver      "intel"
  BusID       "PCI:0:2:0"
EndSection

Section "Device"
### Available Driver options are:-
### Values: <i>: integer, <f>: float, <bool>: "True"/"False", ### <string>: "String", <freq>: "<f> Hz/kHz/MHz",
### <percent>: "<f>%" ### [arg]: arg optional
  #Option     "SWcursor"              #     [<bool>]
  #Option     "HWcursor"              #     [<bool>]
  #Option     "NoAccel"               #     [<bool>]
  #Option     "ShadowFB"              #     [<bool>]
  #Option     "VideoKey"              #     <i>
  #Option     "WrappedFB"             #     [<bool>]
  #Option     "GLXVBlank"             #     [<bool>]
  #Option     "ZaphodHeads"           #     <str>
  #Option     "PageFlip"              #     [<bool>]
EndSection Section "Screen"
```

Identifier "Screen0" Device "Card0" Monitor "Monitor0" SubSection "Display"

Viewport 0 0

Depth 1

EndSubSection SubSection "Display"

Viewport 0 0

Depth 4

EndSubSection SubSection "Display"

Viewport 0 0

Depth 8

EndSubSection SubSection "Display"

Viewport 0 0

Depth 15

EndSubSection SubSection "Display"

Viewport 0 0

Depth 16

EndSubSection SubSection "Display"

Viewport 0 0

Depth 24

EndSubSection EndSection

Section "Screen"

Identifier "Screen1" Device "Card1" Monitor "Monitor1" SubSection "Display"

Viewport 0 0

Depth 1

EndSubSection SubSection "Display"

Viewport 0 0

Depth 4

EndSubSection SubSection "Display"

Viewport 0 0

Depth 8

EndSubSection SubSection "Display"

Viewport 0 0

Depth 15

EndSubSection SubSection "Display"

Viewport 0 0

Depth 16

EndSubSection SubSection "Display"

Viewport 0 0

Depth 24

EndSubSection EndSection

Обычно в файле `xorg.conf` находятся следующие секции:

ServerLayout - указывает идентификаторы используемого X сервером экрана и устройств ввода;

Module - подключает модули расширения X сервера;

InputDevice - описывает используемые устройства ввода, например, мышь или клавиатуру;

Files - задает пути к библиотеке цветов RGB и каталогам шрифтов;

ServerFlags - устанавливает дополнительные флаги X сервера;

Monitor - определяет параметры используемого монитора;

Device - описывает видеоадаптер;

Screen - задает параметры отображения информации на экране, например, глубину цвета.

Хотя конфигурационный файл и не создается, но все же для индивидуальных настроек могут создаваться части конфигурации X сервера в каталоге `/etc/X11/xorg.conf.d/`.

Пример:

$ ls /etc/X11/xorg.conf.d/

00-keyboard.conf 11-evdev-trackpoint.conf

$ cat /etc/X11/xorg.conf.d/00-keyboard.conf

Read and parsed by systemd-localed. It's probably wise not to edit this file # manually too freely.

Section "InputClass"

Identifier "system-keyboard" MatchIsKeyboard "on"

Option "XkbLayout" "us,ru" Option "XkbVariant" ","

Option "XkbOptions" "grp:ctrl_shift_toggle" EndSection

Примечание: Опция `Option "XkbLayout" "us,ru"` задает раскладку клавиатуры, а `Option "XkbOptions" "grp:ctrl_shift_toggle`- метод переключения с русского на английский - сочетание Ctrl-Shift.

Запуск X сервера из командной строки.

X сервер без каких-либо X клиентов можно запустить командой `X`.

Пример:

$ X

X.Org X Server 1.20.10

X Protocol Version 11, Revision 0

Build Operating System: 5.7.11-200.fc32.x86_64

Current Operating System: Linux vlesk-nb 5.10.12-200.fc33.x86_64 #1 SMP Mon Feb 1 02:40:52 UTC 2021 x86_64

...

К системе может быть подключено множество терминалов, а в каждом из них может быть несколько мониторов.

Но даже в системе, обладающей единственным физическим экраном можно запустить несколько X серверов, если для этого имеется достаточное количество ресурсов в системе.

Зачастую запуск дополнительных X серверов нужен для удаленного доступа по протоколам VNC или RDP.

Пример: запуск второго X сервера

$ X :1.0 &

Примечание: оба запущенных сервера готовы принимать соединения по X протоколу для отображения

графики на экране:

Запущенные X серверы используют свободные виртуальные терминалы. Если, например, если в системе используется шесть постоянно работающих виртуальных терминалов, то первый виртуальный X терминал будет доступен с помощью сочетания Ctrl-Alt-F7, а второй: Ctrl-Alt-F8.

Запуск X клиента, который должен обслуживаться первым X сервером, возможен, если указана опция `-display :0.0` :

Пример:

```
$ xterm -display :0.0 &
```

Клиент, который будет обслуживаться вторым сервером, можно запустить так:

Пример:

```
$ xeyes -display :1.0 &
```

Если X сервер должен прослушивать порт TCP вместо этого unix socket для обмена информацией с клиентами, то его следует запустить, используя опцию -nolisten tcp :

Пример:

```
$ X -listen :1.0 &
```

Использование порта TCP для работы X сервера не приветствуется с точки зрения безопасности системы и должно использоваться лишь в системах, где X сервер и X клиенты запускаются на различных компьютерах.

Программа `xinit` позволяет запустить X сервер и эмулятор терминала `xterm`.

Пример:

```
$ xinit
```

Примечание: Эта команда запустит X сервер и эмулятор терминала - программу `xterm`.

Файл `.xinitrc` в домашнем каталоге позволяет указать команды для запуска X клиентов вместо запуска программы `xterm`.

Пример содержимого `~/.xinitrc` :

```
xclock -g 50x50-0+0 & blackbox
```

Примечание: В этом примере после запуска X сервера автоматически стартуют два клиента: программа `xclock` (отображает системное время) и менеджер окон `blackbox`. Опция `-g` программы `xclock` (и многих других X клиентов) указывает размер и положение окна программы на экране. Обратите внимание, что программа `xclock` запущена в фоновом режиме. В противном случае менеджер окон будет запущен лишь после завершения работы `xclock`.

Если файл `~/.xinitrc` отсутствует, то по умолчанию запускается эмулятор терминала xterm :

Пример:

```
xterm -geometry +1+1 -n login -display :0
```

Примечание: Опция `-n` команды `xterm` задает строку заголовка для окна терминала.

Если при запуске X сервера программой `xinit` необходимо задать какие-либо опции, то их можно указать в файле `~/.xserverrc`. В этом файле указывают имя программы X сервера для его старта и требуемые опции.

Пример:

```
exec X :0.0 -listen
```

Примечание: Если такая строка будет присутствовать в файле `~/.xserverrc` , то команда `xinit`запустит X сервер в режиме с поддержкой TCP сетевых соединений.

В командой строке `xinit` можно задавать клиентское приложение для старта, а также указывать параметры запуска `X` сервера:

Пример:

```
$ xinit icewm -- X :0.0
```

Примечание: В этом примере будет запущен X сервер без поддержки соединений по протоколу TCP, после чего будет запущен менеджер окон `icewm`.

Кроме команды `xinit` запустить `X` сервер позволяет также скрипт `startx`, являющийся по сути улучшением `xinit`.

Пример:

```
$ startx
```

Особенностью скрипта `startx` по сравнению с `xinit` является то, что он позволяет обработать общесистемные файлы `xinitrc` и `xserverrc`. В GNU/Linux эти файлы располагаются обычно в каталоге `/etc/X11/xinit`.

Менеджер X сеанса gdm.

Менеджеры X сессии самостоятельно запускают X сервер и запускают X приложение, представляющее собой диалоговое окно для ввода имени пользователя и его пароля.

В GNU/Linux чаще всего используются три менеджера сеанса:

`lightdm` - «легкий» и просто настраиваемый;

`gdm` - поставляется с GNOME;

`kdm` - в составе KDE.

Примечание: Запуск X сервера из командной строки на рабочих станциях не удобен тем, что после входа в обычный сеанс текстовой оболочки приходится либо вручную вызывать команду `startx` или вызывать ее с помощью какого-либо скрипта.

Все менеджеры X сеанса поддерживают специальный протокол XDMCP (X Display Manager Control Protocol). С помощью этого протокола менеджеры X сеанса могут управлять X дисплеем как на локальной, так и на удаленной машине.

Менеджеры сеанса обычно запускаются при переходе в цель `graphical.target`

режим, обеспечивая возможность входа в X сеанс.

Основной каталог конфигурации `gdm` в GNU/Linux - `/etc/gdm` .

Пример: содержимого каталога конфигурации `gdm`

```
# ls -F /etc/gdm/
custom.conf  Init/  PostLogin/  PostSession/  PreSession/  Xsession@
```

Файл `custom.conf` содержит настройки пользователя.

Файл `Xsession` представляет собой скрипт, запускаемый после успешной аутентификации пользователя `gdm`. Он предназначен для настройки окружения и запуска оконных менеджеров или рабочих окружений, а также программ, которые должны запускаться автоматически.

В каталогах `Init`, `PostLogin`, `PreSession` и `PostSession` находятся сценарии, которые запускаются соответственно: при инициализации демона, после входа пользователя, перед запуском сессии и после сеанса.

В каталогах сценарии должны называться или `Default`, или имя соответствовать номеру X сераера, например `:0`

Справку по настройке `gdm` можно посмотреть здесь

https://help.gnome.org/admin/gdm/stable/configuration.html.ru

X приложения.

Многие X приложения поддерживают стандартные опции командной строки. Наиболее часто используют следующие опции:

`-bg` - цвет фона окна X приложения;

`-fg` - цвет текста по умолчанию в окне приложения;

`-bd` - цвет обрамления окна;

`-bw` - толщина обрамления в пикселях;

`-display` - указывает, на каком узле, какие X сервер и экран используются для вывода окна X приложения;

`-fn` - основной шрифт приложения;

`-geometry` - расположение и размер окна X приложения;

`-iconic` - запуск приложения в “иконном” режиме, то есть без раскрытого окна;

`-name` - указывает имя приложения для поиска его ресурсов, если исполняемый файл имеет другое имя, чем указано в файле ресурсов;

`-title` - строка заголовка окна X приложения;

`-xnllanguage` - указывает кодировку, применяемую в приложении (например, ru_RU.KOI8-R);

`-xrm` - определяет имя ресурсов для приложения в файле ресурсов.

Приложения для GNOME или KDE не поддерживают опции X приложений. Вместо этого настройки в них выполняются через различные меню в самих приложениях.

Пример: В качестве примера X приложений можно привести эмуляторы X терминалов, в изобилии имеющиеся в GNU/Linux. Графические эмуляторы терминала – X приложения, предназначенные для работы в командной строке непосредственно из X сессии. Среди них, например, такие:

1. `xterm` - стандартный эмулятор терминала Xorg;
2. `rxvt` - по сравнению с `xterm` обладает урезанными возможностями, зато чрезвычайно экономно расходует ресурсы системы;
3. `aterm` - базируется на `rxvt` и предоставляет по сравнению с ним расширенные возможности;
4. `Eterm` – поставляется в пакете Enlightenment;

5. `gnome-terminal` - поставляется в составе GNOME и обладает удобной возможностью использования вкладок, позволяющих получать доступ к нескольким оболочкам Shell без необходимости открытия новых дополнительных окон;
6. `konsole` - эмулятор терминала, используемый в KDE и также позволяющий создавать вкладки.

Так для запуска `xterm` с заданными размерами окна, темно синим цветом фона и голубым цветом шрифта по умолчанию следует использовать команду:

```
$ xterm -bg navy -fg cyan -geometry 100x40+20+10  &
```

Примечание: Амперсанд, установленный в конце командной строки, запускает `xterm` в фоновом режиме. Если этого не сделать, командная строка будет доступна в окне, из которого произведен вызов, только после остановки `xterm`.

В некоторых случаях встречаются зависшие X приложения, окна которых, возможно, не отображаются на экране вообще. Тем не менее, эти “невидимые” X приложения потребляют ресурсы системы.

Для идентификации зависших приложений удобно воспользоваться командой `ps` с опцией `-u`, позволяющей получить список процессов, запущенных заданным пользователем. Также можно использовать команду `pstree`, выводящую список процессов в древовидном отображении.

Пример: фрагмента вывода команды `pstree`, выполненной в эмуляторе терминала `kterm`

рабочего окружения KDE:

```
|-kdeinit-+-artsd
|         |-evolution-alarm
|         |-2*[kdeinit]
|         |-kdeinit---bash---pstree
|         `-soffice.bin---soffice.bin---4*[soffice.bin]
|-9*[kdeinit]
|-kdm-+-X
|         `-kdm---kde-3.1.5---startkde---kwrapper
```

Примечание: Фрагмент, приведенный выше демонстрирует, что X сервер был запущен менеджером сеанса `kdm`, входящим в состав KDE. Приложения, запущенные в KDE, видны в этом фрагменте выше. Среди них, например, виден процесс `soffice.bin`, соответствующий выполняющемуся приложению Open Office.

Шрифты.

Все шрифты принято разделять на три категории:

Шрифты без засечек (Sans Serif), например, Helvetica и Lucida.

С засечками (Serif), например, Courier и Times.

Специальные шрифты, например, Symbol.

По критерию постоянства ширины символов шрифты подразделяются на:

Пропорциональные, классический пример которых – Times.

Моноширинные, например, Courier. В таких шрифтах размер символов одинаков, что позволяет более удобно читать тексты программ, например.

Примечание: Известно, что шрифты, в которых для различных букв используется своя, наилучшая для данного символа, ширина, читаются лучше.

В X Window используется четырнадцать различных (иногда связанных друг с другом) характеристик шрифтов, задающих имя шрифта:

Foundry – обладатель прав на данный шрифт;

Family Name – имя типа шрифта;

Weight Name – толщина линий: Medium – обычные символы, Bold или Demibold – жирные символы;

Slant – наклон: r – обычные символы (regular), i – курсив (italic), o – с наклоном (oblique);

Setwidth Name – плотность расположения символов: normal – обычная плотность, semicondensed и condensed – уплотненное расположение символов;

Add Style Name – стиль шрифта (как правило не указывается);

Pixel size – величина символов в пикселях;

Point size – величина символов в десятках типографских пунктов;

Resolution X – разрешение экрана по горизонтали в пунктах на дюйм; Resolution Y – разрешение экрана по вертикали в пунктах на дюйм;

Spacing – для пропорциональных шрифтов p, m – для моноширинных;

Average Width – средняя ширина символа;

Charset Registry – алфавит шрифта;

Charset Encoding – кодировка шрифта.

Имя шрифта указывается в виде строки, содержащей необходимое число из

указанных выше четырнадцати параметров, разделенными тире. Параметры, которые не надо указывать, должны быть заменены звездочками.

Пример:

-*-symbol-*-*-*-*-*-240-*-*-*-*-*-*

Примечание: Здесь указан шрифт Symbol с размером символов 24 типографских пункта.

В случае, если под спецификация шрифта подходит более одного шрифта, то X Window использует первый шрифт из подходящих.

X Window способна подбирать наиболее подходящий шрифт на основании имени шрифта, если специфицированный шрифт не найден.

Утилита `xfontsel` позволяет выбрать требуемый шрифт, действуя как фильтр, в котором можно указывать требуемые характеристики шрифтов.

Команда `xlsfonts –fn` шрифт выводит на экран требуемый шрифт для просмотра.

Пример:

$ xlsfonts -fn -*-lucida-*-i-*-*-12-*-*-*-*-*-*-koi8-*

-b&h-lucida-bold-i-normal-sans-12-120-75-75-p-79-koi8-r

-b&h-lucida-bold-i-normal-sans-12-120-75-75-p-79-koi8-r

-b&h-lucida-medium-i-normal-sans-12-120-75-75-p-71-koi8-r

-b&h-lucida-medium-i-normal-sans-12-120-75-75-p-71-koi8-r

Примечание: В этом примере выводятся все шрифты, установленные в системе, с именем `lucida`, с наклонным начертанием, размером `12` пикселей, и с кодировкой `KOI8-R`.

С помощью опции `–fn` можно указать шрифт, который должен использоваться по умолчанию X приложением.

Пример:

xterm –fn -*-courier-bold-*-*-*-*-90-*-*-*-*-cyr-*

Примечание: В окне эмулятора терминала будет использован кириллический шрифт Courier с жирным начертанием символов и величиной 9 пунктов.

Удаленный запуск X приложений.

Поскольку система X Window построена на технологии клиент-сервер, то X сервер, выполняющийся на некотором компьютере, способен обслуживать X

приложения, выполняющиеся на других компьютерах.

Это позволяет запускать клиентские X приложения, требовательные к ресурсам, на мощных компьютерах, отображая при этом их графику с помощью X сервера, запущенного на маломощной рабочей станции.

Для взаимодействия X сервера удаленным X клиентам необходимо пройти авторизацию, то есть получить права на использование данного X сервера.

В простейшем случае при отсутствии необходимости идентификации пользователей, работающих на удаленных компьютерах, авторизация X клиентов может быть произведена с помощью программы `xhost`.

Вызванная без аргументов эта команда отображает список имен узлов, на которых разрешен запуск X приложений, взаимодействующий с данным X сервером.

Пример:

$ xhost

access control enabled, only authorized clients can connect

Примечание: В этом случае команда `xhost` сообщила, что контроль доступа включен. Так как данная команда не отобразила ни одного имени узла, с которого разрешено взаимодействие X клиентов, то данный X сервер не будет отображать графику удаленных приложений.

Для разрешения доступа X приложений отовсюду следует выполнить команду:

$ xhost +

Примечание: Эта команда выключает контроль доступа к X серверу. Для его восстановления необходимо выполнить команду:

$ xhost -

access control enabled, only authorized clients can connect

Используя команду `xhost` можно указывать имена узлов, авторизованных для запуска на них X приложений, взаимодействующих с данным X сервером:

Пример:

$ xhost +classfw

classfw being added to access control list

$ xhost

access control enabled, only authorized clients can connect INET:classfw.net-burg.com

Примечание: В этом примере в список авторизованных узлов внесен узел `classfw`. Его имя указано в файле `/etc/hosts`. Если имя узла не было бы указано в этом файле, то программа `xhost` сделала бы попытку обращения к серверу DNS по имени этого узла.

Запуск X приложений, взаимодействующих с удаленным X сервером, требует указания узла, на котором запущен X сервер. Это может быть осуществлено с помощью указания требуемого узла после стандартной опции X приложений `-display`.

Пример:

classfw:~> xterm -display bamboo.net-burg.com:0.0 &

Примечание: В этом примере X приложение `xterm`, запущенное на узле `classfw` будет отображать свои окна с помощью X сервера, запущенного на узле `bamboo.net-burg.com`.

Вместо указания имени узла с помощью опции `-display` можно воспользоваться переменной окружения DISPLAY, которой назначается имя удаленного узла в качестве значения:

Пример:

classfw:~> DISPLAY=”bamboo.net-burg.com” classfw:~> export DISPLAY

classfw:~> xterm &

Dconf и gsettings

Dconf – низкоуровневая система конфигурации. При помощи **dconf** в **GNOME** и **Unity** хранятся настройки большинства программ.

Dconf это простая система конфигурации основанная на ключах. Ключи находятся в неструктурированной базе данных (ключи, логически связанные между собой объединены в категории). База данных хранится в бинарном файле, который располагается в **~/.config/dconf**.

В большинстве случаев пользователю не нужно вручную редактировать настройки хранящиеся в **dconf**. Но иногда графическое представление тому или иному параметру отсутствует и единственным способом изменить его значение является редактирование ключа напрямую. Это можно осуществить несколькими способами.

GSettings - это консольная утилита, при помощи которой можно управлять ключами **dconf**.

Синтаксис использования **GSettings**:

gsettings [--schemadir <КАТАЛОГ_СХЕМ>] <КОМАНДА> [<АРГУМЕНТЫ…>]

Команда	**Описание**
help	Показать справку
list-schemas	Список установленных схем
list-relocatable-schemas	Список перемещаемых схем
list-keys	Список ключей схемы
list-children	Список потомков схемы
list-recursively	Список ключей и значений, рекурсивно
range	Запросить диапазон значений ключа
get	Получить значение ключа
set	Изменить значение ключа
reset	Сбросить значение ключа
reset-recursively	Сбросить все значения в заданной схеме
writable	Проверить ключ на запись
monitor	Следить за изменениями

Для того чтобы получить более подробную справку по интересующей команде, выполните:

gsettings help <КОМАНДА>

Поиск соответствий схемы настройкам рабочего стола.

Здесь необходимо выяснить пути и названия элементов схемы, отвечающих за конкретные настройки рабочего стола. Например, что именно определяет фон или цвет рабочего стола.

Для начала открываем консоль и набираем от имени обычного пользователя:

gsettings list-schemas | grep background

org.mate.background

org.mate.SettingsDaemon.plugins.background

org.gnome.desktop.background

Тем самым мы получим список схем, в которых может содержаться интересующая нас информация. Теперь выясняем, что именно из перечисленного

отвечает за интересующие нас настройки рабочего стола:

gsettings monitor org.mate.background

Откройте параметры рабочего стола и попробуйте изменить фон, цвет, или иной пункт. В терминале появится информация вида схема/ключ/значение.

gsettings list-recursively org.mate.background

org.mate.background picture-opacity 100

org.mate.background secondary-color '#3C8F25'

org.mate.background show-desktop-icons true

org.mate.background background-fade true

org.mate.background primary-color '#5891BC'

org.mate.background picture-filename '/usr/share/design/current/backgrounds/default.png'

org.mate.background color-shading-type 'vertical-gradient'

org.mate.background picture-options 'zoom'

org.mate.background draw-background true

Установка настроек рабочего стола по умолчанию

Здесь административно установить настройки рабочего стола, которые будут применяться у всех вновь создаваемых пользователей. Например, так можно определить конкретный фон или цвет рабочего стола. Эти изменения не затронут ранее созданных пользователей, которые уже используют графическую среду. Кроме того, пользователи смогут изменить установленные администратором настройки, если захотят. Чтобы заблокировать им такую возможность, читайте следующий раздел.

Все действия выполняются с правами пользователя **root**:

su -

Создаём файл:

nano /etc/dconf/profile/user

user-db:user
system-db:local

Создаём директорию:

mkdir -p /etc/dconf/db/local.d

Создаём файл:

nano /etc/dconf/db/local.d/00_background

Сохраняем в него только те ключи и значения, полученные на шаге 1, которые нас интересуют. В квадратных скобках указываем название схемы.

[org/mate/desktop/background]
secondary-color='#858 5d4d45b5b'
background-fade=true
primary-color='#babadada5555'
picture-filename='/usr/share/backgrounds/mate/desktop/MATE-Stripes-Dark.png'
color-shading-type='vertical-gradient'
picture-options='zoom'
draw-background=true

Блокировка настроек рабочего стола

Пока были определены значения, которые в дальнейшем будут использоваться по умолчанию. Пользователь всё ещё имеет возможность их поменять. Чтобы пользователь не мог внести изменения в настройки рабочего стола никакими средствами, создадим ещё один файл:

nano /etc/dconf/db/local.d/locks/00_background

/org/mate/desktop/background/secondary-color
/org/mate/desktop/background/background-fade
/org/mate/desktop/background/primary-color
/org/mate/desktop/background/picture-filename
/org/mate/desktop/background/color-shading-type
/org/mate/desktop/background/picture-options
/org/mate/desktop/background/draw-background

и выполним команду:

dconf update

Здесь стоит обратить внимание, что схема указана другая - не **/org/mate/background**, а **/org/mate/desktop/background**. В случае с **MATE** это связано с особенностью обработки схем **dconf** и **gsettings**. Посмотреть текущие схемы можно в директории **/usr/share/glib-2.0/schemas**.

Новые настройки вступают в силу немедленно.

Пример настройки dconf - настройка общих для всех пользователей горячих клавиш

Для понимания, какие параметры **dconf** требуется изменить, выполните в

терминале команду для «прослушивания» изменений в системе:

dconf watch /

Параллельно с запущенным **dconf** зайдите в графическую утилиту «**Сочетание клавиш клавиатуры**» («**Главное меню**» — «**Параметры**») и настройте нужные вам сочетания клавиш.

В терминале вы увидите применяемые изменения, например:

dconf watch /

/org/mate/marco/global-keybindings/run-command-window-screenshot

'<Alt>F3'

Далее создайте файл (если уже существует, то оставьте без изменений):

nano /etc/dconf/profile/user

с содержимым:

user-db:user
system-db:local

Создайте файл, который будет отвечать за настройки по умолчанию:

nano /etc/dconf/db/local.d/00_session

[org/mate/marco/global-keybindings]
run-command-window-screenshot='<Alt>F3'

И в конце обновите **dconf** командой:

dconf update

После этого для новых пользователей будут вступать в силу параметры нового **dconf** по умолчанию.

Dconf-editor – является наиболее часто используемым графическим приложением для редактирования ключей **dconf**. В приложении все ключи представлены в виде дерева, можно искать ключи по их названию, жирным текстом выделяются ключи, значение которых было изменено.

По умолчанию **dconf-editor** не установлен на РЕД ОС. Чтобы установить его нужно использовать специальную команду в консоли:

sudo dnf install dconf-editor

Для запуска используется консольная команда:

dconf-editor

С помощью **dconf-editor** можно получить быстрый доступ к множеству "скрытых" настроек, как системных, так и внешнего вида, позволяя быстро сделать

в системе комфортное и удобное для себя рабочее окружение.

UDEV и каталоги /dev и /sys

Для работы с устройствами в Linux имеются два каталога:

`/dev` – каталог, в котором по умолчанию создаются файлы устройств. Эти файлы устройств позволяют нам «общаться» с устройствами посредством стандартных системных вызовов для работы с файлами: открыть файл open(), считать read() или записать write() данные.

`/sys` – виртуальная файловая системы, которая представляет структуры ядра для работы с устройствами в виде файлов.

В первых ОС на основе ядер Linux каталога `/sys` не было, а в каталоге `/dev` файлы устройств создавались вручную или во время инсталляции системы. В результате имелись файлы устройств, а самих устройств не было. Другой проблемой было отсутствие поддержки горячего подключения/отключения (hotplug) устройств.

Для решения проблемы с горячим подключениям и «лишними» файлами устройств изначально использовались такие утилиты как `devfs`, `hotplug` и `HAL` Начиная с ядер версии 2.5 был представлен новый механизм — `udev`.

В апреле 2012 исходный код `udev` слился с исходным кодом `systemd` `udev` — работающая в пространстве пользователя система, с помощью которой системный администратор может создавать обработчики событий.

События, получаемые `udev`, обычно генерируются ядром Linux в ответ на физические события, происходящие с периферийными устройствами. Например, при обнаружении периферийных устройств или "горячем" подключении `udev` может выполнить определённые действия, в том числе и вернуть управление ядру, если необходима загрузка модулей или прошивок.

Подобно предшественникам, утилитам `devfsd` и `hotplug`, `udev` управляет файлами устройств в каталоге `/dev`, добавляя их, переименовывая и создавая символические ссылки. `udev` полностью замещает функционал `hotplug` и `hwdetect`.

Благодаря `udev` в каталоге `/dev` находятся файлы только тех устройств, которые в настоящий момент подключены к системе. Каждое устройство имеет свой соответствующий файл. Если устройство отключается от системы, то данный файл удаляется.

Содержимое каталога `/dev` хранится на виртуальной файловой системе, и все файлы, находящиеся в нём, создаются при каждом запуске системы.

Обработка событий в `udev` происходит параллельно, что теоретически

улучшает производительность старых систем. С другой стороны, это может усложнить администрирование. Так, при перезапуске системы порядок загрузки модулей ядра может измениться, а при наличии в машине нескольких блочных устройств могут поменяться названия их файлов. Например, для системы с двумя жёсткими дисками файл `/dev/sda` после перезагрузки может превратиться в `/dev/sdb udev` входит в состав `systemd` и установлен по умолчанию. Подробнее см. `systemd- udevd.service(8).`

Существует также отдельный от `systemd` форк, который можно установить с пакетом eudev или eudev-git

Для настройки `udev` создаются специальные правила.

Правила должны располагаться в каталоге `/etc/udev/rules.d/` и иметь названия с суффиксом `.rules`

$ cat /etc/udev/rules.d/70-persistent-ipoib.rules

This is a sample udev rules file that demonstrates how to get udev to # set the name of IPoIB interfaces to whatever you wish. There is a

16 character limit on network device names.

Important items to note: ATTR{type}=="32" is IPoIB interfaces, and the # ATTR{address} match must start with ?* and only reference the last 8

bytes of the address or else the address might not match the variable QPN # portion.

#

Modern udev is case sensitive and all addresses need to be in lower case.

ACTION=="add", SUBSYSTEM=="net", DRIVERS=="?*", ATTR{type}=="32", ATTR{address}=="?*00:02:c9:03:00:31:78:f2", NAME="mlx4_ib3"

Утилита `udevadm` предназначена для мониторинга и управления `udev`.

Пример: создадим специальные правила для подключения USB Flash накопителей определенной модели. Во-первых понаблюдаем за процессом подключения USB Flash:

udevadm monitor

monitor will print the received events for:

UDEV - the event which udev sends out after rule processing KERNEL - the kernel uevent

KERNEL[158.634005] add /devices/pci0000:00/0000:00:0b.0/usb1/1-1 (usb)

KERNEL[158.637827] add /devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0 (usb)

...

UDEV [161.309815] add

/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0:0/3:0:0:0/block/**sdb** (block)

UDEV [162.368973] add

/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0:0/3:0:0:0/block/sdb/sdb3 (block)

UDEV [162.519992] add

/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0:0/3:0:0:0/block/sdb/sdb2 (block)

UDEV [162.574735] add

/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0:0/3:0:0:0/block/sdb/sdb1 (block)

Посмотрим характеристики устройства. Обратите внимание, что к тем путям, что выводит `udevadm` нужно просто в начале дописать `/sys`:

ls

/sys/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0:0/3:0:0:0/block/**sdb**

alignment_offset	discard_alignment	hidden	power	sdb1	stat
bdi	events	holders	queue	sdb2	subsystem
capability	events_async	inflight	range	sdb3	trace
dev	events_poll_msecs	integrity	removable	size	uevent
device	ext_range	mq	ro	slaves	

ls **/sys/**devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0 authorized bInterfaceProtocol ep_01 power bAlternateSetting bInterfaceSubClass ep_81 subsystem bInterfaceClass bNumEndpoints host3 supports_autosuspend bInterfaceNumber driver modalias uevent

ls **/sys/**devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/driver -l lrwxrwxrwx. 1 root root 0 Feb 6 12:28

/sys//devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/driver -

> ../../../../../../bus/usb/drivers/**usb-storage**

Или более простой путь к устройству:

ls /sys/block/sdb/

alignment_offset discard_alignment hidden powersdb1 stat

bdi events holders queue sdb2 subsystem capability events_async inflight range sdb3 trace

dev events_poll_msecs integrity removable size uevent device ext_range mq ro slaves

cat /sys/block/sdb/size

15769600

cat /sys/block/sdb/uevent MAJOR=8

MINOR=16 DEVNAME=sdb

DEVTYPE=disk

cat /sys/block/**sdb**/removable

1

cat /sys/block/**sda**/removable

0

Теперь мы с помощью `udevadm` определим производителя устройства:

udevadm info /dev/sdb | egrep 'VENDOR|SUBSYS' E: ID_VENDOR=Generic

E: ID_VENDOR_ENC=Generic\x20

E: ID_VENDOR_ID=cd12

E: SCSI_VENDOR=Generic

E: SCSI_VENDOR_ENC=Generic\x20

E: SUBSYSTEM=block

Или

udevadm info -a /dev/sdb | egrep '/devices/|Vendor|SUBSYSTEM' looking at

device

'/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0:0/3:0:0:0/

block/sdb':

SUBSYSTEM=="block"

looking at parent device

'/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0:0/3:0:0:0': SUBSYSTEMS=="scsi"

looking at parent device

'/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3/target3:0:0': SUBSYSTEMS=="scsi"

looking at parent device '/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0/host3':

SUBSYSTEMS=="scsi"

looking at parent device '/devices/pci0000:00/0000:00:0b.0/usb1/1-1/1-1:1.0': SUBSYSTEMS=="usb"

looking at parent device '/devices/pci0000:00/0000:00:0b.0/usb1/1-1': SUBSYSTEMS=="usb"

ATTRS{idVendor}=="cd12"

looking at parent device '/devices/pci0000:00/0000:00:0b.0/usb1': SUBSYSTEMS=="usb"

ATTRS{idVendor}=="1d6b"

looking at parent device '/devices/pci0000:00/0000:00:0b.0': SUBSYSTEMS=="pci"

looking at parent device '/devices/pci0000:00': SUBSYSTEMS=""

Создаем правило и проверяем его:

cat /etc/udev/rules.d/10-myflash.rules

ACTION=="add", SUBSYSTEM=="block", ATTRS{idVendor}=="cd12", **RUN+="/usr/bin/chgrp myflash /dev/%k", SYMLINK+="myflsh%n"**

Не забудьте создать группу:

groupadd myflash

gpasswd -a admuser myflash

ls -l /dev/sdb*

brw-rw----. 1 root **myflash** 8, 16 Feb 6 14:56 /dev/sdb

brw-rw----. 1 root **myflash** 8, 17 Feb 6 14:56 /dev/sdb1

brw-rw----. 1 root **myflash** 8, 18 Feb 6 14:56 /dev/sdb2

brw-rw----. 1 root **myflash** 8, 19 Feb 6 14:56 /dev/sdb3

ls -l /dev/myflsh*

lrwxrwxrwx. 1 root root 3 Feb 6 14:56 /dev/**myflsh** -> sdb

lrwxrwxrwx. 1 root root 4 Feb 6 14:56 /dev/**myflsh1** -> sdb1

lrwxrwxrwx. 1 root root 4 Feb 6 14:56 /dev/**myflsh2** -> sdb2

lrwxrwxrwx. 1 root root 4 Feb 6 14:56 /dev/**myflsh3** -> sdb3

$ dd if=/dev/**sdb1** of=/dev/null count=10 10+0 records in

10+0 records out

5120 bytes (5.1 kB, 5.0 KiB) copied, 0.0251715 s, 203 kB/s

$ dd if=/dev/**sda1** of=/dev/null count=10

dd: failed to open '/dev/sda1': **Permission denied**

Для получения списка устройств PCI и USB вы можете воспользоваться утилитами lspci и lsusb:

$ lspci

00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02) 00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II] 00:01.1 IDE interface: Intel Corporation 82371AB/EB/MB PIIX4 IDE (rev 01) 00:02.0 VGA compatible controller: VMware SVGA II Adapter

00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 02)

00:04.0 System peripheral: InnoTek Systemberatung GmbH VirtualBox Guest Service 00:05.0 Multimedia audio controller: Intel Corporation 82801AA AC'97 Audio Controller (rev 01)

00:06.0 USB controller: Apple Inc. KeyLargo/Intrepid USB

00:07.0 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 08)

00:0b.0 USB controller: Intel Corporation 82801FB/FBM/FR/FW/FRW (ICH6 Family) USB2 EHCI Controller

00:0d.0 SATA controller: Intel Corporation 82801HM/HEM (ICH8M/ICH8M-E) SATA Controller [AHCI mode] (rev 02)

00:0e.0 Non-Volatile memory controller: InnoTek Systemberatung GmbH Device 4e56

$ lsusb

Bus 001 Device 003: ID cd12:ef18 SMART TECHNOLOGY INDUSTRIAL LTD.

Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub

$ lsusb -v | head -5

Bus 001 Device 003: ID cd12:ef18 SMART TECHNOLOGY INDUSTRIAL LTD.

Device Descriptor:

bLength 18

bDescriptorType 1

Работа с устройствами

Диски и другие накопители

Правила именования устройств описаны в документации к ядру: https://www.kernel.org/doc/html/latest/admin-guide/devices.html

В современных Linux почти все дисковые накопители не зависимо от реального интерфейса подключения: SAS, SATA, USB и др., работают как SCSI устройства.

Для обозначения SCSI дисков используются файлы устройств вида `/dev/sda, b, c,` Где `a` — первый диск, `b` – второй и т. д.

Карты памяти SD/MMC, которые обозначаются как `/dev/mmcblk0, 1, 2, …`

Синтетические дисковые устройства в виртуальных машинах kvm типа virtio. Такие устройства обозначаются как `/dev/vda, b, c,…`

Для NVMe контроллеров используется файл устройства вида: `/dev/nvme0, 1, …` Диски NVMe получают файлы вида `/dev/nvme0n1, 2, ...`

Для взаимодействия с дисками на диски и разделы на них создаются символьные ссылки в подкаталогах каталога `/dev/disk/`

$ ls /dev/disk/

by-id by-partlabel by-partuuid by-path by-uuid

ls /dev/disk/by-id/ -l | head -3 total 0

lrwxrwxrwx 1 root root 9 Feb 7 16:42 ata-VBOX_CD-ROM_VB2-01700376 -> ../../sr0

lrwxrwxrwx 1 root root 9 Feb 7 16:42 ata-VBOX_HARDDISK_VB25e0ca90-16d06b2e -

> ../../sda

При работе с дисками важно знать состояние дисков. В этом может помочь SMART.

Для работы с информацией SMART в Linux применяется специальный пакет smartmontools.

В составе пакета имеются утилита `smartctl` и демон `smartd`.

Утилита `smartctl` используется для управления и получения данных от SMART устройств.

Пример: просканируем диски, которые поддерживают SMART и получим о них сведения:

smartctl --scan

/dev/sda -d scsi # /dev/sda, SCSI device

/dev/nvme0 -d nvme # /dev/nvme0, NVMe device

smartctl --scan | awk '{print "smartctl -a "$1}' | sh | egrep 'Model| Capacity| Serial|SECTION'

=== START OF INFORMATION SECTION ===

Device Model: VBOX HARDDISK Serial Number: VB25e0ca90-16d06b2e

User Capacity: 42,949,672,960 bytes [42.9 GB]

=== START OF INFORMATION SECTION ===

Model Number: ORCL-VBOX-NVME-VER12

Serial Number: VB1234-56789

Namespace 1 Size/Capacity: 8,589,934,592 [8.58 GB]

=== START OF SMART DATA SECTION ===

То же самое но на реальном компьютере:

smartctl --scan

/dev/sda -d scsi # /dev/sda, SCSI device

/dev/sdb -d scsi # /dev/sdb, SCSI device

smartctl --scan | awk '{print "smartctl -a "$1}' | sh | egrep 'Model| Capacity| Serial|SECTION|**overall-health**'

=== START OF INFORMATION SECTION === Device Model: WDC WD10SPCX-60KHST0

Serial Number: WD-WX81A943YYZP

User Capacity: 1 000 204 886 016 bytes [1,00 TB]

=== START OF READ SMART DATA SECTION ===

SMART overall-health self-assessment test result: **PASSED**

=== START OF INFORMATION SECTION ===

Model Family: SandForce Driven SSDs Device Model: KINGSTON SKC300S37A180G

Serial Number: 50026B723404FDD5

User Capacity: 180 045 766 656 bytes [180 GB]

=== START OF READ SMART DATA SECTION ===

SMART overall-health self-assessment test result: **PASSED**

Служба `smartd` предназначена для мониторинга и выполнения действий в случае возникновения проблем с дисками.

Конфигурационный файл службы smartd - /etc/smartmontools/smartd.conf

Сетевые интерфейсы.

Для сетевых интерфейсов не создаются файлы устройств.

В современных системах может использоваться система предсказуемых имен сетевых устройств (PNIDN - Predictable Network Interface Device Names).

В соответствии с PNIDN интерфейсы именуются так:

1. Встроенные сетевые устройства: `enoX` (`X` — номер устройства)
2. Устройства PCI Express с горячим подключением: `ensX` (`X` — номер слота)
3. Имена по расположению оборудования: `enpXsY` (`X` — номер слота PCI, `Y` — номер устройства)
4. Именование по MAC адресу: `enx112233445566`
5. Классический способ именования: `ethX`

Если вы хотите использовать классическое именование сетевых интерфейсов, то вам необходимо задать опцию загрузки ядра: `net.ifnames=0`

Служба `udev` позволяет давать любые удобные вам названия для сетевых интерфейсов.

Пример: Назначение имени сетевой карте в соответствии с MAC адресом.

ifconfig myownnetname

myownnetname: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500 inet6 fe80::4dbf:94b7:62c6:1b2b prefixlen 64 scopeid 0x20<link> ether **52:54:00:e3:15:ae** txqueuelen 1000 (Ethernet)

RX packets 25 bytes 5288 (5.1 KiB)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 11 bytes 1650 (1.6 KiB)

TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

cat /etc/udev/rules.d/70-persistent-net.rules SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="**52:54:00:e3:15:ae**", NAME="**myownnetname**"

Для активизации сетевого интерфейса предназначена команда `/sbin/ip`

Пример:

ip link

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

2: **enp0s3**: <BROADCAST,MULTICAST,**UP,LOWER_UP**> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000

link/ether 08:00:27:82:36:83 brd ff:ff:ff:ff:ff:ff # lspci | grep Ether

00:03.0 Ethernet controller: Intel Corporation 82540EM Gigabit Ethernet Controller (rev 02)

ip address show

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00 inet 127.0.0.1/8 scope host lo

valid_lft forever preferred_lft forever inet6 ::1/128 scope host

valid_lft forever preferred_lft forever

2: **enp0s3**: <BROADCAST,MULTICAST,**UP,LOWER_UP**> mtu 1500 qdisc fq_codel state UP group default qlen 1000

link/ether 08:00:27:82:36:83 brd ff:ff:ff:ff:ff:ff

inet 192.168.101.180/24 brd 192.168.101.255 scope global dynamic noprefixroute enp0s3

valid_lft 75411sec preferred_lft 75411sec

inet6 fe80::b80e:e6d7:98ae:a40b/64 scope link noprefixroute valid_lft forever preferred_lft forever

Команда ip позволяет так же и настроить IP адрес, но этот адрес будет не постоянным.

Для постоянной настройки адресов нужно либо настраивать соответствующие файлы (`/etc/network/interfaces`, `/etc/sysconfig/network-scripts/ifcfg-*`, `/etc/netplan/*.yaml`, ...), либо запускать специальные утилиты (`nmcli`, `nmtui`, `gnome-control-center`, ...).

Поддержка USB.

Ядра Linux серий 2.4 и более поздних включают поддержку USB устройств без необходимости наложения на ядро каких-либо пакетов обновлений. Причем поддерживается возможность горячего подключения устройств (Hot Plug).

USB устройства работают через эмуляцию SCSI.

Поддерживаются следующие стандарты:

1.1 драйверы OHCI и UHCI

2.0 драйвер EHCI

3.0 драйвер XHCI

Wireless WHCI

Virtual VHCI

В некоторых дистрибутивах драйверы USB включены в ядро статически.

Пример:

grep USB_.HCI_HCD /boot/config-$(uname -r) CONFIG_USB_XHCI_HCD=y

CONFIG_USB_EHCI_HCD=y

CONFIG_USB_EHCI_HCD_PLATFORM is not set

CONFIG_USB_OHCI_HCD=y CONFIG_USB_OHCI_HCD_PCI=y

CONFIG_USB_OHCI_HCD_PLATFORM is not set CONFIG_USB_UHCI_HCD=y

CONFIG_USB_WHCI_HCD is not set

В последней строке видно, что поддержка беспроводного USB в ядро не включено.

grep USBIP_.HCI_HCD /boot/config-$(uname -r)

Считывание и изменение настроек USB устройств можно производить с помощью файловой системы `/sys` в подкаталоге `bus/usb`.

Проверить подключенные USB устройства можно командой `lsusb`.

Виртуальные устройства

В Linux (Unix) часто используются виртуальные устройства.

В файловой системе `/sys` для них имеется специальный подкаталог —/sys/devices/virtual

Наиболее часто используются следующие устройства:

1. `/dev/null` — Черная дыра. Устройство, которое уничтожает любые данные в него записанные.
2. `/dev/zero` — Генератор нулей.
3. `/dev/random` и `/dev/urandom` — Генераторы случайных и

псевдослучайных чисел.

Примечание: При чтении данных из устройства `/dev/random` выводятся только случайные байты, полностью состоящие из битов шума «хаотичного» пула ОС. Если «хаотичный» пул опустел, `/dev/random` ничего не выдаст, пока необходимое количество битов в пуле не будет создано, читающая `/dev/random` программа будет ждать появления очередного случайного байта.

В ядре Linux «хаотичный» пул получает энтропию из нескольких источников, в том числе из аппаратного генератора случайных чисел современных процессоров Intel.

Устройство `/dev/random` может быть необходимо пользователям, которые требуют очень высокого коэффициента случайности, например, при создании ключа шифрования, предполагающего длительное использование.

Чтение данных устройства `/dev/urandom` возвратит столько байтов, сколько было запрошено. В результате, если в пуле было недостаточно битов, теоретически возможно найти уязвимость алгоритма, использующего это устройство (на настоящее время нет опубликованных работ о такой атаке). Если это важно, следует использовать `/dev/random`.

dd if=/dev/**random** of=/dev/null count=1000 iflag=fullblock

1000+0 records in

1000+0 records out

512000 bytes (512 kB, 500 KiB) copied, **6.84952 s, 74.7 kB/s**

dd if=/dev/**urandom** of=/dev/null count=1000 iflag=fullblock 1000+0 records in

1000+0 records out

512000 bytes (512 kB, 500 KiB) copied, **0.00643111 s, 79.6 MB/s**

`/dev/loop*` - Закольцованные устройства — это блочные устройства, которые отображают блоки данных обычного файла в файловой системе или другое блочное устройство. Начиная с Linux 3.1, ядро предоставляет устройство `/dev/loop- control`, которое позволяет приложению динамически находить свободное устройство, добавлять и удалять закольцованные устройства из системы. Управление loop устройствами осуществляется командой `losetup`.

Пример: создадим файл образ и подключим его.

dd if=/dev/zero of=~/file.img bs=1024k count=10 # mkfs.ext2 file.img

mount file.img /mnt/

losetup -l

NAME SIZELIMIT OFFSET AUTOCLEAR RO BACK-FILE DIO LOG-SEC

/dev/loop0 0 0 1 0 /root/file.img 0 512

Канонический пример того же самого:

dd if=/dev/zero of=~/fileloop.img bs=1024k count=10

losetup --find --show ~/fileloop.img

/dev/loop1

mkfs.ext2 /dev/loop1

mount /dev/loop1 /mnt/ # umount /mnt

losetup -l

NAME SIZELIMIT OFFSET AUTOCLEAR RO BACK-FILE DIO LOG-SEC

/dev/loop1 0 0 0 0 /root/fileloop.img 0 512

/dev/loop0 0 0 1 0 /root/file.img 0 512

losetup --detach **/dev/loop1**

losetup -l

NAME SIZELIMIT OFFSET **AUTOCLEAR** RO BACK-FILE DIO LOG-SEC

/dev/loop0 0 0 **1** 0 /root/file.img 0 512

umount /mnt # losetup -l

`/dev/tty*` и `/dev/pts/*` - виртуальные терминалы. Первые `/dev/tty*` создаются заранее и используются для работы непосредственно с терминалом компьютера (клавиатура, монитор и мышь). Устройства `/dev/pts/*` создаются по мере необходимости, например, когда подключаются в удаленную сессию (telnet или ssh).

Система печати CUPS.

Изначально в Linux была реализована система печати LPD.

В настоящее время в Linux преобладает использование системы печати CUPS (Common UNIX Printing System).

Преимуществом CUPS является ее поддержка современного протокола IPP (Internet Printing Protocol), предназначенного для управления принтерами и базирующегося на протоколе HTTP.

Эта система так же позволяет использовать файлы описания принтеров PPD (PostScript Printer Description), которые предоставляют информацию о возможностях принтеров.

В основе системы CUPS находится демон - планировщик `cupsd`, обслуживающий очередь заданий на печать.

Конфигурационные файлы для CUPS находятся в каталоге `/etc/cups`. Наиболее важные из них:

1. `cupsd.conf` - основной файл настроек сервера;
2. cups-files.conf — файл с опциями cupsd.
3. `printers.conf` - описания и настройки принтеров;
4. `classes.conf` - описания для целых классов (групп) принтеров;
5. `client.conf` - индивидуальные настройки для клиентов.

Формат файлов конфигурации аналогичен используемому в сервере Apache, многие настройки имеют совершенно одинаковые названия директив.

При запуске демон `cupsd` начинает прослушивает порт 631 TCP.

Пример:

```
# netstat -tanp | grep cups
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      437/cupsd
tcp6       0      0 ::1:631                 :::*                    LISTEN      437/cupsd
```

Если обратиться с помощью браузера по адресу `http://localhost:631` , то при этом будет запущена программа управления сервером CUPS, с помощью которой можно посмотреть настройки, принтеры, очередь печати.

Для администрирования нужно обращаться по HTTPS (`https://localhost:631`). Базовое управление `cupsd` можно производить с помощью `cupsctl`.

`cupsctl` вызванная без аргументов показывает текущие настройки.

Изменить настройки можно с помощью опций или аргументов типа `name=value`.

Пример: включим удаленный доступ и удаленное управление

```
# cupsctl
_debug_logging=0
_remote_admin=0
_remote_any=0
_share_printers=0
_user_cancel_any=0
BrowseLocalProtocols=dnssd
DefaultAuthType=Basic
JobPrivateAccess=default
JobPrivateValues=default
MaxLogSize=0
PageLogFormat=
SubscriptionPrivateAccess=default
SubscriptionPrivateValues=default
WebInterface=Yes

cupsctl _remote_admin=1 _remote_any=1 # netstat -tanp | grep cups
tcp        0      0 0.0.0.0:631             0.0.0.0:*               LISTEN      1027/cupsd
tcp6       0      0 :::631                  :::*                    LISTEN      1027/cupsd

# cupsctl | grep _remote
_remote_admin=1
_remote_any=1
```

Чтобы управлять cups пользователь должен состоять в специальной группе - wheel.

```
# grep SystemGroup /etc/cups/cups-files.conf SystemGroup sys root wheel
```

id admuser

uid=1000(admuser) gid=1000(admuser) groups=1000(admuser),**10(wheel)**,1001(myflash)

Продвинутые настройки сервера `cupsd` такие как разрешенные сети или адрес, на котором работает `cupsd` необходимо изменять непосредственно в файлах конфигурации.

Печать в CUPS.

Основная команда для печати в CUPS - это `lp`.

Пример: печати на принтер по умолчанию документа `test.txt` достаточно выполнить команду:

$ lp test.txt

request id is dj-1 (1 file(s))

Ниже приведены основные опции команды `lp` :

`-d` - для указания имени принтера, в очередь которого должно быть послано задание;

`-h` - указывает имя узла сети для отправки задания на него;

`-i` - предназначена для идентификации задания, что необходимо в случае, если, например, требуется изменить приоритет задания;

`-n` - количество копий;

`-q` - устанавливает приоритет задания в очереди;

`-u` - указывает имя пользователя, задания на печать которого должны быть сняты;

`-H` - задает время, когда задание будет напечатано, либо используется для указания дополнительных опций для данного здания (например, для немедленной печати задания);

`-P` - задает список страниц для печати.

Пример: вывода на печать документа `test.txt` в очередь принтера `dj` и печати двух его экземпляров следует использовать следующую команду:

lp -d dj -n 2 test.txt

request id is dj-2 (1 file(s))

Для печати в системе CUPS можно также использовать команду `lpr`.

Пример: та же задача, что и в предыдущем примере, может быть выполнена и с помощьюlpr :

lpr -#2 -P dj test.txt

Управление принтерами в CUPS.

Для конфигурирования и управления принтерами в системе CUPS может быть вызвана графическая утилита, которая запускается при обращении по адресу `https://localhost:631`.

Если же необходимо использовать командную строку, то можно воспользоваться командой `lpadmin`.

Пример: установка ограничения на максимальное количество страниц в задании:

lpadmin -p laser -o job-page-limit=100 # cat /etc/cups/printers.conf

<DefaultPrinter laser> Info HP1100

Location

DeviceURI usb:/dev/usblp0 State Idle

Accepting Yes JobSheets none none QuotaPeriod 0

PageLimit 100

KLimit 0

</Printer>

Примечание: Команда `lpadmin` установила ограничения для задач, помещаемых на принтер `laser` (что указано опцией `-p`), на максимальное количество страниц в задании равное `100`. Содержимое файла `/etc/cups/printers.conf` подтверждает это.

Пример: показывающий, как можно разрешить заданному пользователю печатать на принтере:

lpadmin -p laser -u allow:test1

Примечание: Это изменение также будет отражено в файле `/etc/cups/printers.conf`:

cat /etc/cups/printers.conf

<DefaultPrinter laser> Info HP1100

Location

DeviceURI usb:/dev/usblp0 State Idle

Accepting Yes JobSheets none none QuotaPeriod 0

PageLimit 100

KLimit 0 AllowUser test1

</Printer>

Современные принтеры предоставляют широкие возможности для конфигурирования, задаваемые в файлах PPD (PostScript Printer Definition).

Для просмотра опций настроек принтера можно выполнить команду:

Пример:

lpoptions -l

Resolution/Output Resolution: 150dpi 300dpi *600dpi 1200dpi Duplex/Double-Sided Printing: *None DuplexNoTumble DuplexTumble

PageSize/Media Size: Letter Legal Executive Tabloid A3 *A4 A5 B5 EnvISOB5 Env10 EnvC5 EnvDL EnvMonarch

InputSlot/Media Source: *Default Tray1 Tray2 Tray3 Tray4 Manual Envelope Auto PageRegion/PageRegion: Letter Legal Executive Tabloid A3 A4 A5 B5 EnvISOB5 Env10 EnvC5 EnvDL EnvMonarch

Option1/Duplexer: True *False

Для изменения каких-либо настроек PPD можно также использовать команду `lpadmin .`

Пример:

lpadmin -p laser -o Resolution=300dpi # lpoptions -l

Resolution/Output Resolution: 150dpi *300dpi 600dpi 1200dpi Duplex/Double-Sided Printing: *None DuplexNoTumble DuplexTumble

PageSize/Media Size: Letter Legal Executive Tabloid A3 *A4 A5 B5 EnvISOB5 Env10 EnvC5 EnvDL EnvMonarch

InputSlot/Media Source: *Default Tray1 Tray2 Tray3 Tray4 Manual Envelope Auto PageRegion/PageRegion: Letter Legal Executive Tabloid A3 A4 A5 B5 EnvISOB5 Env10 EnvC5 EnvDL EnvMonarch

Option1/Duplexer: True *False

Примечание: В этом примере разрешение принтера было изменено на `300dpi`.

Управление очередью печати в CUPS.

Получить информацию о состояние очередей печати CUPS можно с помощью команды

lpstat -a :

Пример:

lpstat -a

laser accepting requests since Jan 01 00:00

Более подробную информацию можно получить, используя опцию `-t` команды `lpstat`:

Пример:

lpstat -t scheduler is running

system default destination: laser device for laser: usb:/dev/usblp0

laser accepting requests since Jan 01 00:00 printer laser is idle. enabled since Jan 01 00:00

Для установки запрета вывода заданий на печать, необходимо воспользоваться командой cupsdisable .

Пример:

cupsdisable laser

lpstat -t scheduler is running

system default destination: laser device for laser: usb:/dev/usblp0

laser accepting requests since Jan 01 00:00 printer laser disabled since Jan 01 00:00 -

Paused

Примечание: После этого задания можно будет ставить на печать, но печататься они не будут:

$ lp -d laser lsmod.asp

request id is laser-1 (1 file(s))

$ lpstat

laser-1 test1 2048 Птн 30 Июл 2004 01:06:39

$ lp -d laser smbldap-howto.fr.html request id is laser-2 (1 file(s))

$ lpstat

laser-1 test1 2048 Птн 30 Июл 2004 01:06:39

laser-2 test1 139264 Птн 30 Июл 2004 01:08:57

Используя команду `cupsreject` можно запретить постановку заданий на печать.

Опция `-r` команды `cupsreject` позволяет указать причину отказа в приеме задания на печать.

Пример:

cupsreject -r 'Ushel na bazu!' laser

lpstat -t scheduler is running

system default destination: laser device for laser: usb:/dev/usblp0

laser not accepting requests since Jan 01 00:00 - Ushel na bazu!

printer laser disabled since Jan 01 00:00 - Ushel na bazu!

laser-1 test1 2048 Птн 30 Июл 2004 01:06:39

laser-2 test1 139264 Птн 30 Июл 2004 01:08:57

Задания в очереди можно перемещать.

Пример: для немедленной печати задания `laser-2` необходимо выполнить команду:

lp -i laser-2 -H immediate

lpstat -u test1

laser-2 test1 13926 Птн 30 Июл 2004 4 01:08:57

laser-1 test1 2048 Птн 30 Июл 2004

01:06:39

Примечание: Опция -i команды lp определяет номер здания, а -H immediate - перемещает его вперед.

Снять задание с печати можно командой `cancel`, причем задания могут быть указаны как индивидуально, так и группой. Так, например,

Пример: удаление из очереди всех заданий от пользователя test1 указывается после опции `- u :`

```
# lpstat   -u  test1
laser-2              test1        139264      Птн  30  Июл  2004
                                              01:08:57
laser-1              test1        2048        Птн  30  Июл  2004
                                              01:06:39

# cancel  -u  test1
# lpstat   -u  test1
```

Для разрешения ставить задания на печать должна быть выполнена команда `cupsaccept`

.

Пример:

lpstat -t scheduler is running

system default destination: laser device for laser: usb:/dev/usblp0

laser not accepting requests since Jan 01 00:00 - Ushel na bazu!

printer laser disabled since Jan 01 00:00 -

Ushel na bazu! # cupsaccept laser

lpstat -t scheduler is running

system default destination: laser device for laser: usb:/dev/usblp0

laser accepting requests since Jan 01 00:00 printer laser disabled since Jan 01 00:00 -

reason unknown

Команда `cupsenable` позволяет разрешить печать.

Пример:

cupsenable laser

lpstat -t scheduler is running

system default destination: laser device for laser: usb:/dev/usblp0

laser accepting requests since Jan 01 00:00 printer laser is idle. enabled since Jan 01 00:00

Система поддержки сканирования SANE.

Прежде всего стоит проверить, поддерживается ли подключенный или планируемый к покупке сканер на уровне драйверов. Данную информацию можно получить на сайте проекта SANE:sane-project.org

В частности, на странице раздела со стабильной поддержкой:http://www.sane-project.org/sane-mfgs, где проще всего искать по производителю (заголовок «Scanners», список «Manufacturers».)

Данные о сканерах представлены в таблицах, где наибольший интерес представляет столбец «Статус». Вот его расшифровка:

• **Complete** — полная поддержка.

• **Good** — поддерживается большинство функций.

• **Basic** — поддерживается только базовый функционал, по факту - хорошо если вообще будет работать.

• **Unsupported** — не поддерживается.

• **Untested** — не тестировался, скорее всего работать не будет, но можно поэкспериментировать самостоятельно или поискать драйвера в нестабильной ветке SANE : http://www.sane-project.org/lists/sane-mfgs-cvs

Если «Статус» имеет последние два значения, то сканер не заработает.

Если данной модели сканера просто нет в списке, необходимо проверить на сайте производителя.Если вендор предоставляет драйвера для сканера, тогда необходимо их скачать и установить (обычно в архиве с драйвером предоставляется скрипт, который разносит все драйвера куда надо), чтобы всё работало через SANE сразу.

Если сканер поддерживается, но, в данный момент, не сканирует, то можно применить следующие рекомендации. И так, сначала самое простое.

Ряд устройств поддерживается драйверами sane-airscan, список поддерживаемых устройств доступен по ссылке https://github.com/alexpevzner/sane-airscan

Ряд устройств HP поддерживаются libsane-hpaio из пакета hplip, список устройств доступен по ссылке https://developers.hp.com/hp-linux-imaging-and-printing/supported_devices/index

Проверка распознавания системой сканера, как устройства.

Сначала нужно посмотреть, определяется ли сканер физически. Большинство сканеров сейчас подключается по USB, поэтому необходимо открыть Терминал и ввести команду, показывающую все подключенные USB-устройства:

lsusb

или

sane-find-scanner

Если среди них нет искомого сканера, то, скорее всего, проблема аппаратная. Стоит проверить, подключён ли сканер по USB, не переломился ли кабель и вообще, исправен ли сам сканер. Так же могут быть проблемы с распознаваем сканера в BIOS или UEFI компьютера. Часто помогает отключение XHCI в UEFI. Если же в выводе Терминала есть строчка подобная этой:

Bus 003 Device 005: ID 04a9:2220 Canon, Inc. CanoScan LIDE 25

то уже хорошо — система видит сканер как USB-устройство и можно двигаться дальше. Естественно, что все цифры и наименование сканера могут быть другими. Важно то, что такая строка есть в принципе.

Теперь нужно ввести в Терминале:

scanimage -L

Если система не может выполнить команду, то, вероятно, не установлен пакет sane-backends. Установить этот пакет можно командой:

dnf install sane-backends

а затем повторить ввод:

scanimage -L

Положительным ответом будет считаться строка, аналогичная этой:

device `plustek:libusb:003:008' is a Canon CanoScan LiDE25 flatbed scanner

Если же Терминал выдаёт отрицательный ответ примерно в таком виде:

No scanners were identified. If you were expecting something different,
check that the scanner is plugged in, turned on and detected by the
sane-find-scanner tool (if appropriate). Please read the documentation
which came with this software (README, FAQ, manpages)

то это может означать:

• аппаратную проблему;

• отсутствие прав на работу со сканером у активной в данный момент учётки пользователя;

• сканеру запрещено обращаться к ядру (где обычно и находятся драйвера).

Решать проблемы лучше в этом же порядке. Про решение аппаратных проблем уже было сказано выше, поэтому можно сразу перейти к настройке прав учётной записи пользователя.

Введённая в Терминал команда

whoami

покажет логин активного в данный момент пользователя.

Далее, нужно узнать, в каких группах состоит этот пользователь:

groups <имя_пользователя>

где <имя_пользователя> — логин, полученный по команде whoami.

Если в этом списке не указана группа lp, необходимо добавить пользователя в группу:

sudo usermod -G lp -a user1

где user1 — логин добавляемого пользователя.

Для тестирования сканера выполните команду:

scanimage -T

Разрешение сканеру обращаться к ядру системы.

Если ответ Терминала по прежнему отрицательный, то, возможно, сканеру запрещено обращаться к ядру. Разрешение можно дать, отредактировав конфигурационный файл «40-libsane.rules», или если такого файла нет, «60-libsane.rules» в текстовом редакторе, запущенном с правами Суперпользователя:

sudo nano /lib/udev/rules.d/40-libsane.rules

В открывшемся тексте наверняка найдётся строка с параметрами сканера со схожим названием. Нужно скопировать строчку с этими данными и вставить её копию сразу под найденной, заменив в копии название сканера и номера idVendor и idProduct на полученные ранее по команде lsusb.

Вот как это будет выглядеть подробнее.

В качестве примера продолжает рассматриваться сканер «CanoScan LIDE 25». При подключении другого сканера, его название и значения idVendor и dProduct будут другими, так же может отличаться номер USB-порта.

Строка с похожим сканером в файле «40-libsane.rules»:

Canon CanoScan LiDE 60

ATTRS{idVendor}=="04a9", ATTRS{idProduct}=="221c", ENV{libsane_matched}="yes"

Данные от lsusb:

Bus 003 Device 005: ID 04a9:2220 Canon, Inc. CanoScan LIDE 25

В файле «40-libsane.rules» в копии строки, в заголовке меняется номер модели сканера с 60 на 25, а двойной ID 04a9:2220, полученный от lsusb нужно разделить на idVendor — 04a9 и idProduct — 2220. В результате, основная и новая строки, друг за другом, будут выглядеть так:

Canon CanoScan LiDE 60

ATTRS{idVendor}=="04a9", ATTRS{idProduct}=="221c",

ENV{libsane_matched}="yes"

Canon CanoScan LiDE 25

ATTRS{idVendor}=="04a9", ATTRS{idProduct}=="2220",
ENV{libsane_matched}="yes"

Остаётся только сохранить файл и закрыть его.

Теперь снова вводим в Терминале:

scanimage -L

Теперь уже ответ должен быть положительным:

device `plustek:libusb:003:008' is a Canon CanoScan LiDE25 flatbed scanner

Если сканер так и не находит, тогда нужно объяснить, как SANE работает со сканером:

У SANE есть конфиг, который обеспечивает доступ к драйверам. Находится он в каталоге /etc/sane.d/dll.conf.

Выглядит он как список из других конфигов, которые находятся в той же папке, что и dll.conf

Когда мы запускаем команду scanimage -L, а она в свою очередь обращается к конфигу dll.conf, который, в свою очередь, обращается к конфигам, отвечающие непосредственно за определенные модели сканера. Сами конфиги уже ведут драйверам, которые находятся в /usr/lib64/sane/

Решением проблемы может быть такое:

В конфиге /etc/sane.d/dll.conf должен быть указан файл-конфига (причем он не должен быть закомментирован символом «#»). Далее в этом конфиге должен быть указан «Vendor ID» и «Product ID»

Пример:

Bus 003 Device 005: ID **04a9:2220** Canon, Inc. CanoScan LIDE 25

Bus 003 Device 005: ID **«Vendor ID»:«Product ID»** Canon, Inc. CanoScan LIDE 25

Запись в конфиге, с записью поддерживаемых сканеров должна выглядеть следующем образом для вашего сканера.

#Samsung X7600 Series

usb 0x04e8 0x3326

Решение проблемы, если сканер виден под root, но не виден под пользователем:

[root@localhost rules.d]# lsusb

Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub

Bus 003 Device 002: ID 03f0:622a Hewlett-Packard

Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub

Здесь строка "Bus 003 Device 002: ID 03f0:622a Hewlett-Packard " - это наше МФУ устройство

Заходим в /dev/bus/usb/ смотрим там через ls -al и находим 003 (по аналогии из вывода lsusb -это Bus 003), заходим в 003 и для устройства 002 (Device 002) устанавливаем права 666.

cd /dev/bus/usb/003

chmod 666 002

Тогда сканер будет виден под пользователем.

Настройка сервера для сканирования по сети

1) установить все пакеты sane:

dnf install sane-backends-daemon sane-frontends

2) Установить xinetd:

dnf install xinetd

3) вписать подсеть которая может сканировать, в нашем случае - 192.168.100.0/24:

nano /etc/sane.d/saned.conf

4) Проверить запись sane-port, должен быть:

nano /etc/services

sane-port 6566/tcp

sane-port 6566/udp

5) сделать запись в xinetd.conf:

nano /etc/xinetd.conf

service sane-port

{

socket_type = stream

server = /usr/sbin/saned

protocol = tcp

user = [здесь указываем пользователя с правами на сканирование]

wait = no

disable = no

}

5) Перезапуск сервиса xinetd:

systemctl restart xinetd

Проверить статус - не должно быть ошибок:

systemctl status xinetd

Настройка клиента для сканирования по сети

1) установить sane, xsane:

dnf install sane-backends-daemon sane-frontends xsane

2) прописать IP-адрес сервера

nano /etc/sane.d/net.conf

3) запустить xsane и выбрать сервер сканирования (возможно понадобится перезапуск рабочей станции)

Если в сети много сканирующих устройств, то при запуске xsane они все отобразятся в программе, чтобы запустить программу xsane с конкретно назначенным сетевым сканером, то можно воспользоваться командой:

xsane kyocera_wc3:192.168.2.190

пример приведен для сканеров марки kyocera.

Но если необходимо, чтобы сканируемая программа не опрашивала сеть на

предмет поиска сетевых сканеров и не выдавала их список, то закомментируйте не нужные модели устройств в каталоге /etc/sane.d/dll.conf

Сетевые интерфейсы в РЕД ОС, принципы их наименования

В РЕД ОС существует 3 уровня адресов: Физический (MAC-адрес сетевого адаптера): назначаются производителями оборудования и управляются централизовано.

Пример:11-A0-17-3D-BC-01

Сетевой (IP адрес): назначается администратором и используется для доставки пакетов между сетями. Пример:192.168.10.1 (IPv4), 2001:DB8:AA10:1::FB (IPv6)

Символьный (DNS-имя): назначается администратором и ипользуется на прикладном уровне пользователями.

Пример: mail.yandex.ru

Для корректной работы сети необходимо выполнить основные сетевые настройки:

IP-адрес: задаётся вручную или автоматически через DHCP

Маска сети: определяет количество бит в IP-адресе, которые используются как адрес сети. Оставшиеся это адрес узла в сети

Шлюз (default gateway): на этот шлюз будут отправляться пакеты, для которых не заданы явно правила маршрутизации

Адрес DNS-сервера: он будет выполнять преобразование DNSадресов (используют пользователи) в IP-адреса (использует система)

Для подключения устройства к сети необходимо выбрать, какой сетевой интерфейс сконфигурировать. Имя сетевого интерфейса складывается из двух составляющих:

1) тип интерфейса

en — Ethernet

sl — serial line IP (slip)

wl — wlan

ww — wwan

2) тип имени

ps<bus> s<slot>— PCI location and USB port number

o <index>— on-board device index number

s <slot>— hotplug slot index number

x<MAC> — MAC address

a<vendor><model>i <instance>— Platform bus ACPI instance id Например: enp0s3 wlp3s0

Для того, чтобы переименовать сетевой интерфейс, например с "enp2s0" в "eth0", надо добавить файл в каталог "/etc/udev/rules.d/*.rules" Написать в файл строчку

SUBSYSTEM=="net", ACTION=="add", ATTR{address}=="XX:XX:XX:XX:XX:XX", NAME="eth0"

где, XX:XX:XX:XX:XX:XX - мак адрес сетевого интерфейса. После перезагрузки интерфейс поменяет своё название. Проверить это можно коммандой dmesg | grep 'renamed'

Ранее сетевые интерфейсы именовались как eth0, eth1 и т. д.

Сетевые настройки системы и клиентские сетевые службы

Для просмотра сетевых настроек выполните команду:

ifconfig

или воспользуйтесь утилитой ip с параметром addr:

ip addr

Тут можно увидеть параметры и название сетевой карты.

ip addr

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

inet 127.0.0.1/8 scope host lo

valid_lft forever preferred_lft forever

inet6 ::1/128 scope host

valid_lft forever preferred_lft forever

2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000

link/ether be:cd:c8:c8:7a:60 brd ff:ff:ff:ff:ff:ff

inet 10.10.10.83/24 brd 10.10.10.255 scope global dynamic noprefixroute enp0s3

valid_lft 435sec preferred_lft 435sec

inet6 fe80::f42e:4ae5:4dc0:ff68/64 scope link noprefixroute

valid_lft forever preferred_lft forever+

По умолчанию сетевой адаптер настроен для получения IP-адреса по **DHCP**. Настройки хранятся в файле **/etc/NetworkManager/system-connections/<имя_сетевого_интерфейса>.nmconnection**. Именуются они, например, как **enp0s3** или **enp0s4**.

В данном примере сетевой интерфейс имеет имя **enp0s3**. Используя команду cat, можно посмотреть содержимое файла настроек:

cat /etc/sysconfig/network-scripts/enp0s3.nmconnection

[connection]

id=enp0s3

uuid=97def639-603a-4167-9613-edd5bd902ef8

type=ethernet

interface-name=enp0s3

timestamp=1704763284

[ethernet]

[ipv4]

method=auto

[ipv6]

addr-gen-mode=eui64

method=auto

[proxy]

Для редактирования файла настроек можно использовать любой текстовый редактор (например **mcedit** или **vi**).

Для установки статического IP-адреса необходимо для параметра method указать значение manual в секции **[ipv4]**:

method=manual

Далее следует указать собственные конфигурации:

Игнорировать **DNS** из **DHCP**:

ignore-auto-dns=true

Указать статический **DNS**:

dns=8.8.8.8;

Указать **IP**, **маску** и **шлюз** по умолчанию:

address1=192.168.1.100/24,192.168.1.1

Для немедленного применения изменений перезапустите сеть:

systemctl restart NetworkManager

Таким образом, при статической настройке IP-адреса, секция **[ipv4]** будет иметь следующий вид:

[ipv4]
address1=192.168.1.100/24,192.168.1.1
dns=8.8.8.8;
ignore-auto-dns=true
method= manual

В файле настройки сетевой карты можно добавить столько DNS-серверов, сколько требуется. Например:

dns=8.8.8.8;8.8.4.4;192.168.1.1;

ВАЖНО!

В работе используются максимум 3 DNS-сервера (те, что указаны первыми).

Проверить шлюз, по умолчанию установленный в системе, можно с помощью команды:

netstat -nr

Kernel IP routing table

Destination Gateway Genmask Flags MSS Window irtt Iface

0.0.0.0 172.16.0.1 0.0.0.0 UG 0 0 0 eno16777736

172.16.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eno16777736

Строка Destination 0.0.0.0 определяет адрес шлюза. Если у вас ее нет, возможно, что для параметра Gateway установлен неверный шлюз, это можно исправить. Установите шлюз по умолчанию:

route add default gw 172.16.0.1

Если вы не сменили при установке ОС имя сервера или вы хотите его изменить, то сделать это можно следующим образом. Сначала проверьте, какое имя сервера (hostname) у вас установлено:

hostname

В примере имя сервера будет изменено на server.work с помощью команды:

hostnamectl set-hostname server.work

Для смены имени ПК перезагрузка не требуется.

Для записи статических DNS имён в ОС имеется отдельный файл /etc/hosts

Способы изменения настроек сети

Основной набор инструментов для работы с сетевыми возможностями Net-tools. Содержит большое количество пакетов. Посмотреть можно **dnf repoquery -l net-tools**

/usr/bin/**netstat**

/usr/sbin/**ether-wake**

/usr/sbin/**ipmaddr**

/usr/sbin/**mii-diag**

/usr/sbin/**nameif**

/usr/sbin/**route**

/usr/sbin/**arp**

/usr/sbin/**ifconfig**

/usr/sbin/**iptunnel**

/usr/sbin/**mii-tool**

/usr/sbin/**plipconfig**

/usr/sbin/**slattach**

Для просмотра состояния сетевых интерфейсов используется команда ifconfig [<опции>]

Запущенная без опций команда отобразит все активные сетевые

интерфейсы, а с ключом «**-a**» — все имеющиеся в системе, включая неактивные.

В качестве аргумента можно указать имя интерфейса, в таком случае дополнительными опциями можно произвести изменение его настроек, которые будут действовать до перезапуска сети или перезагрузки системы.

Команда arp

Чтобы увидеть таблицу ARP вашего сервера, вы можете использовать команду arp. Есть некоторые параметры, которые можно использовать с командой arp, но для просмотра таблицы ARP Linux по умолчанию вы можете использовать эту команду с параметром «-e».

Пример:

arp -e

Команда netstat

Для проверки сетевых подключений. Команда «netstat» очень полезна, особенно для устранения неполадок.

Без опций netstat предоставляет информацию об открытом сокете. Но есть много вариантов netstat. Например, если мы используем

«netstat -r», он дает нам информацию о таблице маршрутизации.

Команда route

Чтобы проверить таблицу IP-маршрутизации в Linux, мы используем команду «route». В этих таблицах вы можете увидеть все определенные и изученные маршруты к любому пункту назначения.

Таблица маршрутизации также может быть создана с дополнительными параметрами, такими как “add”, “delete”, “flush”. Например:

Чтобы добавить маршрут:

route add -net 192.168.10.1/24 gw 192.168.1.1

Чтобы удалить маршрут:

route del -net 192.168.17.1/24 gw 192.168.2.1

Сейчас происходит замена утилит пакета net-tools, в который, кроме ifconfig, входят route, arp, netstat, на утилиты ip и ss из пакета iproute2.

ip [опции] объект команда [параметры]

Опции - это глобальные настройки, которые сказываются на работе всей утилиты независимо от других аргументов, их указывать необязательно.

объект - это тип данных, с которым надо будет работать, например: адреса, устройства, таблица arp, таблица маршрутизации и так далее;

команды - какое-либо действие с объектом;

параметры - само собой, командам иногда нужно передавать параметры, они передаются в этом пункте.

ip [опции] объект команда [параметры]

объекты

address или **a** - сетевые адреса.

link или **l** - физическое сетевое устройство.

neighbour или **neigh** - просмотр и управление ARP.

route или **r** - управление маршрутизацией.

rule или **ru** - правила маршрутизации.

tunnel или **t** - настройка туннелирования в этом пункте.

это не все объекты которые поддерживает команда ip

ip [опции] объект команда [параметры]

команды

допустимые *команды* зависят от объекта, но обычно это:

- add — добавление
- delete — удаление
- show — отображение информации (по умолчанию)
- help — справка по допустимым командам

Поддерживаются сокращения (address - a, link - l, route - r и т.д.)

ip [опции] объект команда [параметры]

посмотреть все IP адреса

ip a или ip addr show

просмотра информации в кратком виде опция -br:

ip -br a show

посмотреть IP адреса только по определённому сетевому интерфейсу

ip a show eth0

Утилита ip также позволяет создать виртуальные сетевые интерфейсы. Виртуальные сети — VLAN — удобное решение для построения различных сегментов логических сетей на базе одной физической сети. Естественно, кроме поддержки со стороны ОС требуется реализация VLAN на сетевом оборудовании (коммутаторы, маршрутизаторы).

Пример:

ip link add link enp0s3 name vlan122 \ type vlan id 122

Здесь vlan122 — имя виртуального интерфейса, 122 — номер сети, задаваемой администратором.

Примеры использования

ip link show - отобразить состояние всех сетевых интерфейсов

ip l sh - то же самое

ip l - то же самое

ip link show eth0 - отобразить состояние eth0

ip link set eth1 up - включить eth1

ip link set eth1 down - выключить eth1

ip address show - показать все ip адреса и их интерфейсы

ip a l permanent - отобразить только статические ip адреса

ip a l dynamic - отобразить только динамические ip адреса

ip addr add 1.1.1.13/24 dev eth0 - установить ip адрес eth0

ip addr del 1.1.1.13/24 dev eth0 - удалить ip адрес инт-са eth0

ip r sh показать все маршруты в таблице маршрутизации

ip route get 10.10.20.0/24 - отобразить маршрут к этой сети

ip route add 10.10.20.0/24 via 192.168.50.100 - создать маршрут

ip route delete 10.10.20.0/24 - удалить маршрут.

ip neigh show dev eth0 - посмотреть все ARP записи для eth0

Изменить настройки сетевого интерфейса «на лету» (до следующей перезагрузки интерфейса или системы) можно с помощью утилиты ip:

ip address change 192.168.10.1/24 dev enp0s3

Этой же утилитой можно задать адрес (вместо change указать add) или удалить его (указать del)

Настройка IP-сети с помощью **nmtui**

(Network Manager текстовый пользовательский интерфейс)

nmtui — удобный инструмент настройки сетевых интерфейсов с помощью графического дисплея

Настройка Network Manager с помощью nmcli:

nmcli [опции] объект [команда]

где основные *объекты* это

device - управление сетевыми интерфейсами;

connection - управление соединениями;

networking - управление сетью в целом;

general - отображение состояния сети и Network Manager;

radio - управление сетевыми протоколами, wifi, ethernet и т д.

Допустимые *команды* зависят от объекта. Служба NetworkManager должна быть запущена.

nmcli general status

nmcli connection show

Примеры использования nmcli

nmcli general status — просмотр состояния Network Manager

nmcli device status — просмотр состояния интерфейсов

nmcli connection show — просмотр доступных подключений

nmcli connection show eth0 — просмотр подробной информации об eth0

nmcli connection up eth0 - активация подключения по eth0

nmcli connection add con-name "dhcp" type ethernet ifname ens33 — создание подключения с именем «dhcp» типа ethernet для устройства ens33

nmcli connection add con-name "static" ifname enp2s0 autoconnect no type ethernet ip4 192.168.0.210 gw4 192.168.0.1 — создание подключения со статическим IP-адресом

nmcli conn modify "static" ipv4.dns 8.8.8.8 — задание DNS-сервера nmcli radio wifi on — включение WiFi

nmcli device wifi connect "TP-Link" password 12345678 name "TP-Link Wifi" — подключение к сети WiFi

Команда ethtool

Чтобы проверить настройки вашей сетевой карты (NIC), можно использовать команду ethtool. Эта команда позволяет настраивать такие параметры, как скорость, дуплексный режим и т. Д.

ethtool eth5

Используется ethtool с различными параметрами. Например, можно использовать приведенную ниже команду, чтобы заставить этот интерфейс иметь скорость 100 и полный дуплекс.

ethtool –s speed 100 duplex full

Управление сетевой картой и драйвером с помощью ethtool

ethtool [параметры] имя-сетевого-интерфейса

Примеры использования

ethtool eth0 — получение общей информации о сетевом адаптере

ethtool -i eth0 — получение информации о драйвере сетевого адаптера

ethtool -S eth0 — статистика сетевого адаптера (если поддерживается драйвером)

ethtool -s eth0 duplex half — перевод адаптера в полудуплексный режим

ethtool -p eth0 — идентификация порта сетевой карты мигающим индикатором

ethtool -k eth0 — отображает возможные настройки адаптера

ethtool -K eth0 — меняет указанные настройки адаптера, если это возможно

ethtool -t eth0 online — выполняет тестирование адаптера в режиме online

Команда host

Команда host используется для сопоставления IP — Имя хоста . Вы можете получить результаты как для IPv4, так и для IPv6.

host www.google.com

www.google.com has address 173.194.38.180

www.google.com has address 173.194.38.176

www.google.com has address 173.194.38.177

www.google.com has address 173.194.38.178

www.google.com has address 173.194.38.179

www.google.com has IPv6 address 2404:6800:4003:802::1014

Команда nslookup

«nslookup» используется для DNS-запросов. Он предоставляет информацию о DNS-сервере.

$ nslookup www.google.com Server: 192.168.42.1

Address: 192.168.42.1

Non-authoritative answer: Name: www.google.com

Address: 2a00:1450:4017:809::2004 172.217.17.164

Команда dig

«dig» — это аббревиатура от Domain Information Groper. Он в основном используется для простого поиска DNS на DNS-сервере, таком как CName, MX-записи и т. Д.

dig google.com

Команда ifstat

Команда ifstat используется для отслеживания статистики сетевого интерфейса. Эта статистика может включать использование полосы пропускания, полученные кадры, отброшенные кадры, ошибки, коллизии и т. д.

ifstat eth0

Вы можете использовать параметр «-z», чтобы очистить статистику и начать заново.

ifstat –z eth0

Команда curl

Команда curl — одна из сетевых команд в Linux, которая используется для передачи файлов. Curl может использовать различные протоколы, такие как HTTP, HTTPS, FTP, FTPS, SFTP, SCP и т. д.

Загрузить abc.txt с xyz.com на свой локальный компьютер. Это набор библиотек, в которых реализуются базовые возможности работы с URL страницами и передачи файлов.

$ curl –O https://xyz.com/abc.txt

Определить внешний IP компьютера, используя внешний скрипт.

curl ifconfig.co

Получить и вывести в терминал прогноз погоды

curl -4 wttr.in/Murom

Команда wget

«Wget» используется для загрузки содержимого веб-серверов. Вы также можете загрузить определенный файл с веб-сервера. Ниже вы можете увидеть команду wget, которая загрузит abc.txt с xyz.com.

wget http://xyz.com/abc.txt

Команда ss

Команда ss дает подробную информацию о сокетах. Мы можем использовать опцию «-l» для вывода списка прослушивающих сокетов и опцию «-t» только для tcp-соединений.

ss -l ss -t

Команда sftp

Команда sftp — это один из протоколов передачи файлов, используемых в Linux. Это безопасный протокол передачи файлов. В SFTP FTP используется поверх SSH.

sftp ipcisco@192.168.5.1

Команда iftop

iftop одна из сетевых команд в Linux, которая используется для отображения текущего использования полосы пропускания на сетевых интерфейсах. Вы также можете использовать эту команду для определенного интерфейса с параметром “-i” .

sudo iftop sudo iftop -i eth0

Команда ifup

На ваших серверах есть сетевые интерфейсы. Вы можете административно включить или отключить эти интерфейсы. Ключевые слова для этих заданий — «ifup» и «ifdown».

ifup eth3 ifdown eth4

Команда ping

Команда «ping» — наиболее часто используемая команда в мире сетевых технологий. Ping используется в Linux аналогично другим платформам, таким как cisco, juniper и т. Д.

ping -c 4 192.168.1.1

Команда traceroute

«Traceroute» — это команда, которая используется для проверки узлов на пути к месту назначения из вашей системы. Другими словами, он проверяет переходы и их доступность. Команда «traceroute» широко используется в сетевом мире, и использование этой важной сетевой команды такое же в системах Linux.

Команда tracepath

Команда tracepath — это аналогичная команда Linux для отслеживания пути, такая как команда traceroute. Но вам не обязательно быть суперпользователем, чтобы использовать команду tracepath. С помощью tracepath вы можете перечислить серию хостов на пути к месту назначения.

tracepath www.google.com

Команда tcpdump

Команда Tcpdump — это наиболее часто используемая команда анализа и захвата в Linux. С помощью этого вывода команды вы можете видеть передачи TCP в своей сети.

sudo tcpdump -i eth0

Здесь мы проверили TCP-трафик интерфейса Ethernet по умолчанию.

Команда w

Команда w используется для проверки текущей активности системы. Это может быть действие пользователя или любой процесс, связанный с системой. Вы можете перечислить текущих пользователей на вашем Linux-компьютере с помощью команды w.

Команда whois

Команды whois проверяют базу данных whois и возвращают информацию об IP-адресе и домене.

whois google.com whois город.рф

И конечно же хочется отметить, что это не все сетевые возможности

Лабораторные работы

Задание 1

1. Войти в систему в графическом режиме обычным пользователем.
2. Запустить текстовый редактор, создать на рабочем столе файл test.txt со своим ФИО.
3. Изменить раскладку переключения клавиатуры на Ctrl+Shift.
4. Запустить файловый менеджер (Caja или Double Commander).
5. Создать в домашней папке каталог test.
6. Скопировать в него созданный ранее файл test.txt.
7. Запустить терминал, в нём запустить файловый менеджер mc.
8. Перейти в корневой каталог в левой панели. Клавишей F9 из меню mc для левой панели настроить сортировку по имени в обратном порядке.
9. Перейти в каталог /var/log в правой панели. Клавишей F9 из меню mc для правой панели в меню «Формат списка» включить укороченный формат.
10. Сделать скриншот окна mc клавишей PrintScreen, сохранить его в домашнем каталоге.
11. Запустить менеджер архивов Engrampa.
12. Создать в домашнем каталоге архив arc.tar.gz. Добавить в него файл test.txt и каталог test.
13. В конфигурационном файле /etc/default/grub изменить значение таймаута выбора пункта загрузки
14. Обновить конфигурацию grub2 командой grub2-mkconfig
15. Перезагрузить систему и убедиться, что изменения вступили в силу
16. С помощью systemctl вывести список всех служб (имеющих тип service)
17. Вывести список всех служб, которые не смогли запуститься (состояние failed)
18. Посмотреть состояние службы cups
19. Остановить службу cups и попытаться зайти в браузере на его страницу (localhost:631)
20. Остановить сокет cups и попытаться зайти в браузере на его страницу (localhost:631)
21. Перезапустите ОС. Проверьте состояние службы cups.

22. Уберите сервис из автозапуска командой systemctl. Перезапустите ОС и проверьте состояние службы cups.
23. Запустите сервис cups при помощи systemctl. Проверьте его статус.
24. Остановите сервис, удалите службу из автозапуска и замаскируйте его. Перезапустите ОС и попробуйте запустить сервис. Проверьте его статус.
25. Размаскируйте сервис и попробуйте его запустить. Проверьте его статус.
26. Установите таргет загрузки системы – в значение multi-user.target. Перезагрузите ОС.
27. Верните таргет в исходное значение (graphical.target) и перезагрузите ОС.

Задание 2

1. Выполните установку РЕД ОС на ВМ. Системные требования использовать минимальные

Задание 3

1. Определите имя исполняемого файла оболочки, запускаемой при входе в сеанс с вашим учетным именем.
2. Выполните команду `ls -l ~`, выводящую содержимое Вашего домашнего каталога в подробном формате.
3. Опишите структуру командной строки предыдущей команды: где в командной строке имя команды, опции и аргументы?
4. Посмотрите на приведенный выше пример использования команды `ls -d`. Проверьте, можно ли менять части командной строки местами.
5. Получите список оболочек, установленных на Вашей системе.
6. Смените оболочку, загружаемую по умолчанию на `tcsh`, выйдите из сеанса и снова зайдите в сеанс. Изменилось ли что-нибудь?
7. Установите оболочку `bash` в качестве оболочки, загружаемой при входе в сеанс. Временно запустите оболочку `tcsh` и выйдите из нее.
8. В большинстве современных дистрибутивов GNU/Linux в оболочке `tcsh` в качестве приглашения командной строки используется не `%`, а `$`. Как в таком случае определить, в какой оболочке вы работаете? Для ответа на этот вопрос воспользуйтесь командой `ps`, выводящей список

процессов, запущенных пользователем.

9. Получите список системных команд, файлы которых находятся в каталоге `/bin`. Есть ли среди них знакомые Вам команды?
10. Обычно помимо встроенной команды `pwd` имеется ее системный двойник в каталоге
11. `/bin`. Попробуйте вызвать встроенную и системную команды `pwd` с опцией `--help`. Есть ли разница в работе встроенной и системной команд?
12. Выполните последнюю введенную команду заново.
13. Пользуясь клавишами управления курсором, найдите в файле истории команду `echo` и выполните ее.
14. Вызовите последнюю введенную команду `echo` по первым двум буквам ее имени. Вызовите команду, содержащую подстроку `ho`.
15. Выведите в текстовый редактор последнюю исполненную команду. Выведите полный список команд в файле истории.
16. Введите команду `ls -ld /` , а затем получите список всех возможных продолжений командной строки. Подставьте все возможные продолжения в командную строку и выполните команду.
17. Получите значение переменной окружения `HISTFILESIZE`, пользуясь механизмом дополнения имен с автоматическим определением контекста дополнения.
18. Получите список всех возможных подстановок имен переменных окружения, начинающихся с символа `H`.
19. Введите команду `ping -c3` и первую букву имени хоста `localhost`. Получите список всех возможных продолжений имен хостов. Выберите из них `localhost` и выполните команду.
20. Проверьте, работает ли опция `--help` с командой `id`.
21. Попробуйте вместо `--help` использовать `-h` с командой `id`.
22. Подсчитайте количество строк в файле `/etc/hosts` с помощью команды `wc`, изучите базовую подсказку по этой команде и используйте нужную опцию для подсчета строк.
23. Является ли команда `cd` встроенной командой оболочки?
24. Получите помощь по команде `alias`.
25. Для чего нужен ключ `-P` команды `man`?
26. Выведите все имеющиеся данные о системе `man`.

27. Получите все страницы, называющиеся `groff`.
28. Найдите все файлы страниц `man`, касающиеся любых обектов, называющихся `exit`. Вы собираетесь опубликовать игру для X Window, называющуюся `xzombie`. Как следует назвать файл страницы `man` по этой игре?
29. В каком каталоге следует установить эту страницу, если она используется на данной системе локально и не связана с системой установки пакетов?
30. Получите помощь по команде `who`, пользуясь системой `info`.
31. Найдите раздел документации `info`, относящийся к теме переходов по гипертекстовым ссылкам, перейдите в родительский узел.
32. Перейдите в начало страницы, найдите строку `scroll` на этой странице. Выйдите из `info`.
33. Проверьте, какая документация для оболочки Bash имеется в вашей системе. Имеется ли дополнительная информация о системе `man`?

Задание 4

1. Создайте директорию test в домашнем каталоге и файл test.txt внутри неё
2. Заполните файл любым текстом
3. Создайте символическую ссылку на файл при помощи команды ln –s test.txt soft.link
4. Попробуйте посмотреть содержимое файла cat soft.link
5. Создайте жёсткую ссылку при помощи команды ln test.txt hard.link
6. Попробуйте посмотреть содержимое файла cat hard.link
7. Посмотрите тип файлов при помощи команды ls –l
8. Удалите файл test.txt
9. Посмотрите тип файлов при помощи команды ls –l
10. Попробуйте просмотреть содержимое файлов hard.link и soft.link
11. Командой fdisk отобразить разделы на дисках. Нужный ключ узнать из справки или man
12. Перенаправить вывод fdisk с информацией о дисках в файл parts.txt
13. Отобразить список смонтированных разделов командой mount
14. Отобразить смонтированные разделы только с файловой системой ext4

15. Командой mkdir создать каталог testdir в домашнем каталоге
16. Командой cp скопировать в этот каталог файлы из /var/log, начинающиеся на букву X
17. С помощью df определить свободное место на диске в мегабайтах. Результат перенаправить в файл free.txt в каталоге testdir
18. Командой du определить размер каталога testdir в килобайтах (не выводить размеры вложенных файлов)
19. Определить размер каталога testdir с помощью mc (командой меню)
20. Создать в каталоге testdir жесткую ссылку free.log на файл free.txt
21. Создать в каталоге testdir символическую ссылку home на домашний каталог
22. Командой ls вывести подробную информацию о каталоге testdir
23. Запустить DoubleCommander
24. Скопировать в нём в каталог testdir файлы из каталога /etc, начинающиеся на букву n Определить размер каталога testdir из DoubleCommander
25. В консоли скопировать каталог testdir в каталог toDelete
26. В консоли удалить каталог toDelete
27. Создать в виртуальной машине новый диск размером 1 ГБ. Если нет такой возможности — сделать в домашнем каталоге файл командой dd if=/dev/zero of=test.file bs=1M count=1000
28. С помощью fdisk просмотреть информацию о новом диске. Результат вывести в fdisk.log в домашнем каталоге пользователя.
29. Командой mount вывести смонтированные файловые системы, результат вывести в mount.log в домашнем каталоге пользователя.

Задание 5 -6

1. Создать пользователя testusr командой useradd
2. Создать группу testgrp
3. Добавить пользователя testusr в группу testgrp
4. Создать каталог rdir в домашнем каталоге пользователя от имени root
5. Попытаться создать обычным пользователем файл в каталоге rdir
6. Посмотреть командой ls права на каталог rdir
7. Командой chmod дать всем пользователям права на запись в каталог rdir

8. Создать обычным пользователем файл в каталоге rdir
9. Сменить владельца каталога rdir и всех вложенных файлов на testusr командой chown
10. Убрать права на запись для «остальных пользователей» командой chmod
11. С помощью битовой маски установить на rdir права на чтение/запись/выполнение для владельца, чтение/выполнение для группы владельца, отсутствие прав для остальных
12. Удалить пользователя testusr и группу testgrp.
13. Создать пользователя tuser с домашним каталогом /tmp/tuser и оболочкой /bin/sh **если ругается SELinux то переводим его в режим предупреждений командой setenforce 0**
14. Создать группу tgroup
15. Включить в неё пользователя tuser

Задание 7

1. Проверить установлен ли пакет binutils
2. Получить информацию об этом пакете
3. Вывести список файлов этого пакета в файл binutils.list
4. Определить какому пакету принадлежит файл /bin/mount
5. Найти установленные пакеты, начинающиеся на «ip»
6. Найти доступные пакеты по строке iptables
7. Получить информацию о пакете shorewall
8. Установить пакет shorewall с помощью yum или dnf
9. Удалить пакет shorewall с помощью rpm

Задание 8

1. Запустить в консоли команду cat /dev/zero > /dev/null
2. Во второй консоли командой top посмотреть её активность
3. Завершить команду в первой консоли с помощью Ctrl+C
4. Запустить в консоли команду cat /dev/zero > /dev/null в фоне
5. Найти её в расширенном выводе команды ps (можно с помощью grep)
6. Завершить команду cat с помощью kill
7. Посмотреть права доступа на файл /bin/passwd

8. Запустить команду passwd, но не менять пароль
9. Из второй консоли найти найти процесс passwd в расширенном выводе команды ps
10. Определить от какого пользователя запущена команда passwd

Задание 9

1. Удостовериться, что к виртуальной машине не подключены оптические диски (вкладка «устройства» окна виртульной машины).
2. Выполнить команду udevadm monitor. Подключите «образ диска дополнений гостевой ОС» на вкладке «устройства» окна виртуальной машины.
3. Определите тип файловой системы (параметр ID_FS_TYPE) оптического диска при помощи утилиты udevadm
4. Определите размер оптического диска с помощью утилиты udevadm (понадобятся дополнительные опции)
5. Проверить установлен ли пакет cups и если нет — установить его
6. Запустить службу cups с помощью systemctl
7. В браузере зайти на страницу web-интерфейса CUPS (localhost:631)
8. Если доступен сетевой принтер — попытаться добавить его либо с помощью CUPS, либо графической утилитой «Настройки принтера»
9. Командой ip вывести информацию о сетевых интерфейсах в системе (имя интерфейса, назначенный адрес, сетевая маска)
10. Командой ip вывести информацию о таблице маршрутизации, определить адрес шлюза по умолчанию.
11. Найти файл с настройками сетевого интерфейса и скопировать его в домашний каталог
12. Запустить графическую утилиту настройки сетевых соединений
13. Зайти в ней в настройки активного сетевого подключения, изучить возможные настройки
14. Установить во вкладке IPv4 настройки сети вручную. Они должны быть такими же, как были до этого
15. Убедиться, что сеть работает.
16. Сравнить новую конфигурацию сетевого интерфейса в конфигурационном файле со старой из домашнего каталога

17. Заменить конфигурацию на старую и перезапустить службу NetworkManager
18. Запустить графическое приложение Yum Extender или Dnfdragora.
19. Найти в нём в категории «Группы» пункт «Приложения».
20. В нём выбрать для установки inkscape
21. Установить выбранную группу.
22. Запустить установленную программу, сделать скриншот.
23. Поменять тему оформления, сделать скриншот
24. Добавить три приложения на выбор в избранное, сделать скриншот категории Избранное.
25. Установить пакет dconf-editor.
26. Создать для него ярлык на рабочем столе.
27. Запустить калькулятор MATE, выполнить в нём любую арифметическую операцию, запомнить результат.
28. Запустить dconf-editor, в нём найти настройки для калькулятора (org/mate/calc)
29. Найти настройку по отображению незначащих нулей (show-zeroes) и включить её (изменить на True)
30. Запустить калькулятор MATE, выполнить в нём ту же арифметическую операцию, сравнить результат